Application Server

2023 R2 P01

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, mechanical, photocopying, recording, or otherwise, without the prior written permission of AVEVA Group Limited. No liability is assumed with respect to the use of the information contained herein.

Although precaution has been taken in the preparation of this documentation, AVEVA assumes no responsibility for errors or omissions. The information in this documentation is subject to change without notice and does not represent a commitment on the part of AVEVA. The software described in this documentation is furnished under a license agreement. This software may be used or copied only in accordance with the terms of such license agreement. AVEVA, the AVEVA logo and logotype, OSIsoft, the OSIsoft logo and logotype, ArchestrA, Avantis, Citect, DYNSIM, eDNA, EYESIM, InBatch, InduSoft, InStep, IntelaTrac, InTouch, Managed PI, OASyS, OSIsoft Advanced Services, OSIsoft Cloud Services, OSIsoft Connected Services, OSIsoft EDS, PIPEPHASE, PI ACE, PI Advanced Computing Engine, PI AF SDK, PI API, PI Asset Framework, PI Audit Viewer, PI Builder, PI Cloud Connect, PI Connectors, PI Data Archive, PI DataLink, PI DataLink Server, PI Developers Club, PI Integrator for Business Analytics, PI Interfaces, PI JDBC Driver, PI Manual Logger, PI Notifications, PI ODBC Driver, PI OLEDB Enterprise, PI OLEDB Provider, PI OPC DA Server, PI OPC HDA Server, PI ProcessBook, PI SDK, PI Server, PI Square, PI System, PI System Access, PI Vision, PI Visualization Suite, PI Web API, PI WebParts, PI Web Services, PRiSM, PRO/II, PROVISION, ROMeo, RLINK, RtReports, SIM4ME, SimCentral, SimSci, Skelta, SmartGlance, Spiral Software, WindowMaker, WindowViewer, and Wonderware are trademarks of AVEVA and/or its subsidiaries. All other brands may be trademarks of their respective owners.

U.S. GOVERNMENT RIGHTS

Use, duplication or disclosure by the U.S. Government is subject to restrictions set forth in the license agreement with AVEVA Group Limited or its subsidiaries and as provided in DFARS 227.7202, DFARS 252.227-7013, FAR 12-212, FAR 52.227-19, or their successors, as applicable.

AVEVA Legal Resources: https://www.aveva.com/en/legal/

AVEVA Third Party Software Notices and Licenses: https://www.aveva.com/en/legal/third-party-software-license/

Contents

Welcome to Application Server 46

What's new in Application Server? 47
- **July 2024** 47
- **June 2024** 47
- **December 2023** 47
- **July 2022** 48
- **October 2021** 50
- **December 2020** 51
- **April 2020** 52

System Platform IDE 55
- **Galaxy overview** 55
- **Start the System Platform IDE** 56
  - IDE backstage configuration options 56
  - Install and activate your license 57
  - Manage security certificate verification 57
  - AVEVA CONNECT 58
  - IDE application views 58
    - Application object icons 58
    - Perform a search in a view 61
    - Model view 61
    - Deployment view 63
    - Flex licenses view 64
    - Derivation view 64
    - IO devices view 65
    - Mapping view 66
    - Operations view 67
  - Keyboard shortcuts 68
- **Customize your workspace** 69
  - Dock views 69
  - Float views 70
  - Hide views 70
  - Reset the workspace 70
  - Synchronize views 71
  - Configure user preferences 71
- **Create a Galaxy** 73
- **Connect to a Galaxy** 75
- **Change users** 77
- **View the Galaxy** 77
  - Template folder 79
  - Visualization folder 79

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Work with objects **79**
View object properties 79
Manage folders 80
Create folders 82
Create child folders 82
Delete folders 83
About using CONNECT 83
Log into CONNECT 83
Navigate between shared drives 84
Upload/download graphics to the cloud 84
Upload graphics to CONNECT 85
Download graphics to the local visualization folder 85
Upload/download support for various cloud content 86
Manage graphics in CONNECT 86
Manage graphics with multiple users 87
About cloud graphic versions 87
Overwrite graphic contents in the cloud 88
Cloud graphic versions - scenarios and examples 89
Manage galaxy style libraries 90
Import a galaxy style library 90
Export a galaxy style library 90
Reset a pre-defined or user-defined galaxy style library 91
Create derived templates 91
Derive templates from another derived template 92
Create contained templates 92
ApplicationObject containment 93
Use contained names 96
Containment examples 97
View containment relationships 99
Rename contained objects 100
Edit objects 101
Object editor user assistance 101
Help file structure 102
Object editor layout 102
Lock and unlock template attributes 103
Set object security 106
Group lock and security controls 107
Attributes page 108
Scripts page 112
Object information page 114
Customize help 115
Locate the help folders 115
Field attributes, UDAs, and extensions pages 115
View the Field Attributes Page 115
Determine whether to convert field attributes 116
Optimize performance if you use field attributes 116
Convert field attributes to attributes 116
Special considerations when converting field attributes 119
Limitations and exclusions when converting field attributes 119

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Update scripts to reference converted attributes . . . 120
Use the galaxy browser to reference objects . . . 120
Browse for attributes . . . 120
View attribute details in the galaxy browser . . . 122
Browse for graphics . . . 124
Browse for element properties . . . 126
Create a filter for the galaxy browser . . . 126
Change how information is shown in the galaxy browser . . . 128
Configure security . . . 129
About security . . . 129
Security model for galaxies . . . 130
Authentication modes . . . 131
Multiple accounts per user . . . 132
Using the local administrator account with single sign on . . . 133
Changing security settings . . . 133
About security groups . . . 133
About roles . . . 134
About users . . . 137
About named credentials . . . 138
Create a new credential . . . 138
Associate an OS group with a named credential . . . 139
Configure credentials . . . 141
About SQL Server security . . . 142
Configure roles and groups . . . 142
Assign roles to users . . . 147
Delete security groups . . . 149
Delete roles . . . 150
Delete users . . . 150
About OS group-based security . . . 150
Connecting to a remote node for the first time . . . 150
Cached data at log in . . . 151
Mixed or native domains . . . 151
Domain local groups . . . 151
Security and distribution domain configurations . . . 151
InTouch access levels security . . . 152
Secured and verified writes . . . 152
Configurable descriptions and comments for secured and verified writes . . . 153
Reason description . . . 153
Predefined comment list . . . 153
Editable comment box . . . 154
Logged information for secured and verified writes . . . 154

Design standards . . . 155
Galaxy options . . . 155
Auto-buffering . . . 155
Enable auto-buffering . . . 155
Advanced communications management . . . 156
Out-of-order data handling . . . 157

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Alarm latching . . . 157
Enable alarm latching . . . 158
About templates and instances . . . 159
Instances . . . 160
Templates . . . 161
Propagation . . . 161
About derived templates . . . 162
About base templates . . . 162
Application templates . . . 163
Device integration templates . . . 164
System templates . . . 164
WinPlatform object . . . 164
AppEngine object . . . 164
Area object . . . 164
ViewEngine object . . . 165
WebViewEngine object . . . 165
InTouchViewApp object . . . 165
ViewApp object . . . 166
About the galaxy style library . . . 166
Open the galaxy style library . . . 167
Manage galaxy style libraries . . . 167
Duplicate a galaxy style library . . . 168
Set the default galaxy style library . . . 168
Rename a galaxy style library . . . 168
Export a galaxy style library . . . 169
Export all galaxy style libraries . . . 170
Import a galaxy style library . . . 170
About element styles . . . 171
Element styles details . . . 172
Change the visual properties of an element style . . . 173
Override element style text properties . . . 173
Override element style fill properties . . . 174
Override element style line properties . . . 175
Override element style outline properties . . . 175
Preview an element style . . . 176
Reset an element style to its default values . . . 176
Change the visual properties of user-defined element styles . . . 177
Apply element styles to elements . . . 177
Element style list . . . 177
Properties grid . . . 178
Format painter . . . 179
Clear an element style . . . 179
Set an element style as a default for a canvas . . . 180
Apply element styles to groups of elements . . . 180
Set run-time behavior for a group to TreatAsIcon . . . 180
Element style behavior with a group of elements . . . 180
Use element styles to configure an animation . . . 180
Use element styles to configure a boolean animation . . . 181
Use element styles to configure a truth table animation . . . 181

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Delete a condition from an animation truth table . . . 182
Change the processing order of element styles in a truth table animation . . . 182
About the quality and status tab . . . 183
Show data status . . . 184
Show quality and status with the status element . . . 184
Show quality and status by overriding the appearance of an animation . . . 184
Quality and status styles . . . 184
Use status elements . . . 185
Draw the status element on the canvas . . . 185
Configure the status element . . . 185
Set the appearance of a status element . . . 185
Override element appearance depending on quality and status of its attributes . . . 186
Override the text appearance of elements to indicate non-good status or quality . . . 186
Override the fill appearance of elements to indicate non-good status or quality . . . 186
Override the line appearance of elements to indicate non-good status or quality . . . 186
Add outlines to elements to indicate non-good status or quality . . . 187
Preview all status appearances . . . 188
Reset an override appearance to its default . . . 188
Format Styles tab . . . 188
Configure a global numeric format . . . 189
Format list . . . 190
Special characters styles . . . 191
User-defined global number styles . . . 191
Rename user-defined global number styles . . . 191
Import and export user-defined global number styles . . . 192
Set number formats by regional locales . . . 192
Design time considerations for numeric formats . . . 193
Enter input numbers in U.S. format . . . 193
Set the regional locale of the computer hosting the HMI/SCADA application . . . 195
Runtime considerations for formatting numbers . . . 196
Numeric format restrictions by regional locale . . . 197
Numeric strings enclosed within quotation marks . . . 198
Numbers passed as script parameters . . . 198
Double-byte character languages . . . 198
Switch to another galaxy style library during runtime . . . 199

Build . . . 201
Configure objects . . . 201
Add attributes to an object . . . 201
Configure an attribute . . . 202
Data type definitions . . . 203
Attribute naming conventions . . . 205
Attributes and scripting . . . 206
Add features to an attribute . . . 207
Features inheritance . . . 207

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Enable I/O for an attribute . . . 209
Configure I/O as read-only . . . 211
Configure I/O as read/write . . . 212
When objects are on scan . . . 213
Use read/write I/O in scripts . . . 213
Configure I/O as write-only . . . 214
I/O outputs . . . 215
I/O data quality . . . 216
ATS-configure I/O for use with AVEVA Telemetry Server Communication Drivers . . . 216
Enable history for an attribute . . . 217
Enable limit alarms for a numeric attribute . . . 221
Enable rate of change (ROC) alarms for a numeric attribute . . . 223
Enable deviation alarms for a numeric attribute . . . 225
Enable a state alarm for a boolean attribute . . . 227
Configure an external alarm source for a boolean attribute . . . 229
Enable a bad value alarm for an attribute . . . 231
Enable statistics for a numeric attribute . . . 232
Enable log change for an attribute . . . 233
I/O auto assignment . . . 234
Assign areas and objects to scan groups . . . 235
Rename application, system, and device integration objects . . . 237
Rename scan groups . . . 237
Edit and validate I/O assignments . . . 237
How to use the mapping view for I/O auto assignment . . . 238
Validate I/O references . . . 240
Override I/O auto assignments . . . 240
Batch overrides of I/O assignments . . . 241
Upload runtime configuration changes for auto-assigned objects . . . 242
I/O auto-assignment workflow example . . . 242
I/O auto-assignment with OPC clients . . . 244
Override scenario 1 — Siemens S7 controllers with an OPCClient object . . . 244
Override scenario 2 — Allen-Bradley CIP controllers with an OPCClient object . . . 245
Write and edit scripts . . . 246
About object scripts . . . 247
Script execution . . . 248
Script locks and change propagation . . . 251
Create and work with content . . . 252
Add a graphic to an object . . . 253
Errors and warnings in an owned graphic . . . 253
Edit a graphic . . . 254
Set graphics properties with a galaxy style library . . . 254
Use element styles to modify graphics . . . 254
Use quality and status indicators to provide runtime feedback . . . 254
Use UI themes to set the appearance of WPF controls . . . 255
Symbol Wizards . . . 256
Create Symbol Wizards . . . 259
Embed a Symbol Wizard into an application . . . 259
Rename a graphic . . . 260
Delete a graphic . . . 260

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Rearrange Objects to Improve AVEVA OMI ViewApp Navigation . . . 261
Rearrange objects: drag and drop . . . 261
About rearranged objects . . . 261
Assign an alias name to an object . . . 262
Assign alias name to an object in Model view . . . 264
About alias names during operations . . . 265
Configure historization . . . 266
Application Server historization components . . . 266
Send historical data between Application Server and the historian . . . 267
Save object attribute data to the historian . . . 268
Save process values (PV) as historical data . . . 268
Save data quality (VTQ) as historical data . . . 268
Additional quality data saved to the historian . . . 269
Save the data timestamp as historical data . . . 270
Save alarms and events as historical data . . . 271
Deploy and undeploy attributes . . . 272
Save historical data during run time . . . 272
Advanced communication management . . . 272
Store-and-forward mode . . . 273
Buffered data behavior . . . 273
Configure common historical attributes . . . 273
Configure system objects to store historical data . . . 276
Configure the WinPlatform object to store historical data . . . 278
Configure an AppEngine object to store historical data . . . 280
Configure an Area object to save alarm counts as historical data . . . 281
Configure an application object to save historical data . . . 282
Alarms and events . . . 284
Event definition . . . 284
Types of events . . . 285
Alarm definition . . . 286
Alarm states . . . 287
Latched alarm attributes . . . 288
Types of alarms . . . 288
State alarms . . . 288
Limit alarms . . . 290
Target deviation alarms . . . 291
Rate of change alarms . . . 292
Statistical alarms . . . 292
Set alarm state with object attributes . . . 292
AlarmModeCmd attribute . . . 293
AlarmInhibit attribute . . . 293
AlarmMode attribute . . . 293
AlarmModeEnum attribute . . . 294
Alarms and buffered data . . . 294
Alarm detection . . . 294
Set the alarm state for individual alarms . . . 294
Enable, silence, and disable alarms . . . 295
Enable alarms . . . 296
Silence alarms . . . 296

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Disable alarms . . . 296
Shelve alarms . . . 296
Enable alarm shelving . . . 297
Shelve alarms at runtime . . . 298
Throttle alarms . . . 299
Propagate timestamps with alarms and events . . . 300
Alarms or events become active . . . 300
Alarms become disabled . . . 300
Alarms revert to normal . . . 300
Alarm acknowledgement . . . 301
Acknowledge alarms when signature is required . . . 301
Dismiss a latched alarms . . . 301
Configure alarms . . . 302
Configure alarms for system objects . . . 303
Configure WinPlatform object alarms . . . 304
Configure alarm priority for communication failure . . . 304
Register using "Galaxy_<GalaxyName>" instead of "Galaxy" option . . . 304
Configure alarms for an AppEngine object . . . 307
Configure alarms and events for application objects . . . 308
Set alarms on the attributes page . . . 311
Distribute alarms and events . . . 313
Subscribe to alarms and events from a client . . . 314
Using InTouch HMI as the Alarm and Event Client . . . 314
Syntax of alarm queries . . . 314
Alarm query syntax when 'Register Using Galaxy_<GalaxyName>' is enabled . . . 315
Alarm query examples . . . 315
Alarm requirements for InTouch client applications . . . 316
Alarms and event implemenation in application server and runtime applications . . . 316
Configure plant state-based alarms . . . 318
Mapping alarm modes to plant states . . . 319
Configure the state-based alarm mode on an area object . . . 320
View plant state-based alarms at runtime . . . 321
User access and security at runtime . . . 322
Configure priority ranges for alarm historization, mapping, and shelving . . . 322
Configure historization for alarms and events . . . 323
Monitor alarm severities at runtime . . . 324
Alarm ranking at runtime . . . 324
Aggregated alarm information . . . 325
Configure alarm state aggregation . . . 325
Aggregated alarm state information . . . 326
AlarmCntsBySeverity attribute . . . 328
AggregatedAlarmStatesIncluded attribute . . . 328
Local alarm display in the Object Viewer . . . 329
Aggregated alarm display in OMI ViewApp navigation . . . 331
Alarm adorner attributes . . . 333
Monitor alarm state information at runtime . . . 334
Object wizards . . . 335
Object Wizard features . . . 337
Create a basic Object Wizard . . . 337

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Object wizard configuration best practices . . . 339
Object wizard GUI panes . . . 340
Object wizard user interface details . . . 343
Object wizard components . . . 348
Add content for use with an object wizard . . . 349
Add a new object-owned symbol . . . 350
Link to shared content in the visualization folder . . . 351
Associate content with a choice or option . . . 352
Disassociate content from a choice or option . . . 353
Multiple content item associations . . . 353
Map attribute and graphic configurations to a choice or option . . . 353
Attribute and graphic overrides . . . 354
Delete content from an object . . . 354
Delete an attribute or content from a template . . . 355
Example1: Map a symbol wizard to an object wizard . . . 355
Example2: Configure a linked graphic associated with multiple choices . . . 359
Conditional visibility expressions for choice groups, choices, and options . . . 361
Add a basic visibility expression . . . 362
Add a visibility expression with logical operators . . . 362
Operators, rules, and behavior for visibility expressions . . . 363
Object wizard visibility settings in a derived template . . . 364
Example 1: Basic visibility expressions . . . 365
Example 2: Complex visibility expressions . . . 367
Example 3: Visibility of groups, choices, and options in a derived object wizard . . . 370
Object wizard trimming . . . 371
Enable trimming . . . 372
Example: Enable trimming for attributes and graphics . . . 373
Attribute and graphic overrides . . . 375
Enter overrides and set visibility for attribute and graphic values . . . 376
Attribute and graphic overrides . . . 378
Configurable settings for attribute and graphic values . . . 379
Object wizard derivation . . . 380
Derive a child template from a template with an object wizard . . . 380
Set visibility of choices and options in a derived template . . . 381
Reasons to derive a template from an object wizard . . . 382
Changes to object wizards in derived templates . . . 382
Behavior of object wizard components in a derived template . . . 383
Example: Set visibility of derived wizard choices and options . . . 383
Object wizard test mode . . . 384
Verify behavior of visibility expressions . . . 385
About object wizard behavior in test mode . . . 387
Example 1: Verify behavior of associated attributes . . . 387
Example 2: Verify behavior of associated content . . . 389
Propagation of object wizard changes to derived objects . . . 391
Object wizard behavior when a default choice is deleted . . . 392
Propagation limitations of object wizards . . . 392
Differences in propagation behavior of object wizard overrides between releases . . . 392
Extend an inherited object wizard . . . 394
Add a choice group to a derived template . . . 394

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Add an option to a derived template . . . 395
Add an Association to an Inherited Object Wizard . . . 396
Add a script to a derived template . . . 396
Add content to a derived template . . . 398
Link to a graphic in the visualization folder . . . 399
Graphics and Object Wizards . . . 400
Example 1: Add a graphic to the object . . . 400
Example 2: Link to a graphic in the visualization folder . . . 401
Configure instances . . . 401
Configure an instance with the IDE . . . 401
View and configure graphics . . . 402
Guidelines for configuring instances with an object wizard . . . 403
Override default settings . . . 403
Create a new instance graphically . . . 404
VisualBuild Object options . . . 407
VisualBuild graphic options . . . 408
Add a configured asset to an overview graphic . . . 409
Configure the OPCClient object for data integration . . . 410
Convert existing DINetwork object references . . . 411
Why you should use OPCClient objects instead of legacy DINetwork objects . . . 413
Configure the OPC UA server . . . 414
OPC UA configuration checklist . . . 414
Use an OPC UA client to read and write run-time data . . . 415
Configure and deploy the OPC UA service . . . 415
Client access rules and galaxy security . . . 419
Configure the firewall for the OPC UA service . . . 420
Configure the run-time node firewall . . . 420
Firewall test . . . 423
Configure server and client certificates for third-party OPC UA client applications . . . 424
Export the OPC UA server certificate to the OPC UA client node . . . 425
Import the OPC UA server certificate on the client computer . . . 425
Configure OPC UA client certificates on the OPC UA server . . . 427
OPC UA ports and port configuration . . . 427
Configure an OPC UA data source object . . . 427
Verify OPC UA certificate installation . . . 434
Use an OPC UA client to read and write run-time data . . . 437
Use a script to call an OPC UA server . . . 438
Work with the Industrial Graphic Editor . . . 438
Industrial Graphic Editor workflows for use with HMI/SCADA applications . . . 439
InTouch managed applications . . . 439
AVEVA OMI ViewApps . . . 440
Create a graphic in the IDE . . . 440
Create graphics in the visualization folder . . . 440
Create graphics in AutomationObject templates . . . 441
Create graphics in AutomationObject instances . . . 441
Add an existing graphic to your graphic . . . 442
Add an Industrial Graphic to an AVEVA OMI pane . . . 444
About graphic scaling . . . 444
Additional information about Industrial Graphics . . . 445

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Element styles . . . 445
Style libraries . . . 445
Visual properties defined by element styles . . . 445
Element styles in animations . . . 446
Property style order of precedence . . . 446
Update element styles at application run time . . . 447
Manage element styles . . . 447
Import and export element style libraries . . . 447
Change the visual properties of an element style . . . 447
Override element style text properties . . . 448
Override element style fill properties . . . 448
Override element style line properties . . . 449
Override element style outline properties . . . 449
Preview an element style . . . 450
Reset an element style to its default values . . . 451
Change the visual properties of user-defined element styles . . . 451
Apply element styles to elements . . . 451
Element style list . . . 451
Properties grid . . . 452
Format painter . . . 453
Clear an element style . . . 453
Set an element style as a default for a canvas . . . 454
Apply element styles to groups of elements . . . 454
Set run-time behavior for a group to TreatAsIcon . . . 454
Element style behavior with a group of elements . . . 454
Use element styles to configure an animation . . . 454
Use element styles to configure a boolean animation . . . 455
Use element styles to configure a truth table animation . . . 455
Delete a condition from an animation truth table . . . 456
Change the processing order of element styles in a truth table animation . . . 456
Using custom properties with Application Server . . . 457
Custom property examples . . . 457
Use custom properties to show historical summary data . . . 457
Analog statistical summary data . . . 457
State statistical summary data . . . 458
Historical summary period . . . 459
Show statistical summary data . . . 460
Use binding in custom properties . . . 462
Change the expression or reference of a custom property at runtime . . . 464
Working with the SignedWrite() function for secured and verified writes . . . 465
SignedWrite() runtime behavior . . . 465
SignedWrite() script execution sequence at runtime . . . 466
SignedWrite() scripting tips . . . 466
Examples of using the attribute parameter in the SignedWrite() function . . . 467
Configure the SignedWrite() script function . . . 469
Show/Hide content script functions . . . 470
Configure the Show/Hide Content script functions . . . 470
About the best pane match algorithm . . . 472
Content display rules . . . 474

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Languages and localization **475**
Define and configure galaxy languages 475
Graphics language switching 475
Alarm language switching 476
InternationalizedString attributes 476
Configure languages for a Galaxy 477
Add a language to a Galaxy 477
Remove a language from a Galaxy 478
Modify the font for a language 479
Change the default language for a Galaxy 479
Types of language dictionary files 480
Export files for offline translation 480
About localization workflow 481
Export language data for internationalized strings 482
Export language data for all Industrial Graphics in a Galaxy 483
Export language data for specific objects 485
Export graphic language data for a managed InTouch application 486
Export graphic language data for a published InTouch application 487
Export text to an existing dictionary file 488
Export alarm comments for offline translation 488
Guidelines and recommendations 488
Organize your export 488
Translation file formatting and editing 488
Reimport 489
About the alarm comments language file 489
Export alarm comments from very large galaxies 489
Export all galaxy alarm comments 490
Export alarm comments by area 490
About file names 491
Export objects not assigned to an area 491
Translate exported graphic language files 491
Translate exported graphic text dictionary files 492
Translate exported alarm comment language files 493
Import translated language files 496
Importing Translated International Strings 496
Import translated graphic dictionary files 498
Examples of graphic or object mismatch handling during language imports 500
Language data handling for galaxy operations 502
Import translated alarm comment language files 503
Re-export alarm comments 504
Export new untranslated alarm comments 504
Export modified existing alarm comments 504
Test the language switching functionality at run time 504
Sequencer Object: Configuration and Use **504**
Sequencer object structure 505
Step program 505
Steps 505
Outputs 506
Literal values 506

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Aliases . . . 506
Aliases . . . 506
Settings . . . 507
Step structure . . . 509
Step entry, step body and step exit . . . 509
Minimal steps . . . 509
Step condition and jump condition . . . 510
Triggers . . . 512
Timers . . . 512
Simple timer . . . 512
Cyclic timers . . . 513
Combined triggers and timers . . . 513
Delay timer . . . 513
Always true and always false conditions . . . 513
Write on exit . . . 513
Jump to . . . 513
Steps and aliases naming conventions . . . 513
Limitations . . . 514
Define steps . . . 514
Add steps . . . 514
Configure steps . . . 515
Rename a step . . . 519
Append steps . . . 519
Insert a step . . . 520
Reorder steps . . . 520
Delete steps . . . 521
Define minimal steps . . . 521
Define the initial and final steps . . . 522
Name the step program . . . 523
Validate the step program . . . 523
Add comments to Sequencer objects . . . 524
Lock the step program . . . 524
Define step and jump conditions . . . 525
Define steps with trigger-based conditions . . . 525
Define steps with timers . . . 526
Define steps with a cyclic timer . . . 527
Define steps with a monthly timer . . . 528
Define steps with a weekly timer . . . 528
Define steps with a daily timer . . . 529
Define steps with an hourly timer . . . 529
Define steps with a minute-based timer . . . 530
Define steps with always true or always false conditions . . . 531
Define trigger and timer combinations . . . 532
Use operators to connect triggers . . . 532
Retentive and non-retentive timers . . . 532
Configure steps with combined conditions . . . 533
Overview of trigger and timer combinations . . . 533
Define outputs for a step . . . 539
Create aliases . . . 539

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Assign an output value to a step . . . 540
Assign a literal output value . . . 540
Assign a literal output value in the output matrix . . . 541
Assign a literal output value in the edit outputs dialog box . . . 541
Assign an alias value as the output value . . . 543
Assign an alias value in the output matrix . . . 543
Assign an alias in the edit outputs dialog box . . . 544
Remove an output value . . . 545
Remove values from the output matrix . . . 545
Remove values from the edit outputs dialog box . . . 545
Rename an alias . . . 546
Delete an alias . . . 546
Alias definitions . . . 547
Create an alias . . . 547
Assign an attribute reference to an alias . . . 547
Rename an alias . . . 548
Change the appearance order of aliases . . . 548
Remove aliases . . . 548
Delete an alias . . . 548
Clear aliases . . . 549
Lock the alias definition list . . . 550
View alias references . . . 550
View Alias Crossreferences . . . 551
Define Sequencer program settings . . . 551
Configure the initial command . . . 552
Configure errors and alarms . . . 553
Configure the initialization timeout . . . 553
Halt program execution on error . . . 554
Use alarms to report errors . . . 554
Configure security . . . 555
Modify access rights to the program execution command . . . 556
Modify access rights to step settings . . . 556
Modify access rights to the active step command settings . . . 556
Modify access rights to the initial step command setting . . . 557
Modify access rights to the final step command setting . . . 557
Modify access rights to the sequence configuration command . . . 558
Modify access rights to the output value arrays of a selected step . . . 558
Modify access rights to the timer presets of a selected step . . . 559
Modify access rights to the jump destination setting of a selected step . . . 560
Clear settings . . . 561
Import and export the Sequencer program configuration . . . 561
Export the step program . . . 561
Export the alias definitions . . . 562
Export program settings . . . 563
Import a step program . . . 563
Import alias definitions . . . 564
Import program settings . . . 565
Run the Sequencer . . . 566
Deploy the Sequencer object . . . 566

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Restart and failover conditions . . . 566
What are restart/failover conditions? . . . 566
How does the running sequence program react to a restart/failover event? . . . 566
Detect restart/failover events . . . 567
Configure alarms to detect if program execution halts . . . 567
Use attributes to detect if program execution halts . . . 567
Resume operation after a restart/failover event . . . 568
Automatically resume execution after a restart/failover event . . . 568
What happens to the execution state after the sequence program resumes? . . . 568
Sequencer program commands and states . . . 569
View and monitor the current active step . . . 569
View the configuration of the current active step . . . 569
Monitor execution of the current active step . . . 570
Control the program flow at run time . . . 571
Start or stop program execution . . . 572
Start program execution . . . 572
Stop program execution . . . 573
Reset program execution . . . 574
Reset program execution . . . 574
Hold or resume program execution . . . 575
Hold program execution . . . 575
Resume program execution . . . 576
Advance a step . . . 577
Advance a step . . . 577
Run in single step mode . . . 579
Initiate single step mode . . . 579
Confirm a transition when running in single step mode . . . 580
Jump to a specific step . . . 580
Jump to a specific step . . . 580
Set initial command at run time . . . 582
Modify the Sequencer program at run time . . . 582
View and change the configuration of a selected step . . . 582
View configuration of a selected step . . . 582
Change configuration of a selected step . . . 584
View the execution order of aliases . . . 585
Change step program and alias configuration at run time . . . 585
Load and save Sequencer configuration . . . 586
Save Sequencer configuration at run time . . . 586
Load Sequencer configuration at run time . . . 587
Upload run-time changes . . . 588
Set initial and final steps . . . 589
Detect errors at run time . . . 589
Detect if program execution halts . . . 590
Detect condition trigger failure . . . 590
Detect OnEntry output failure . . . 591
Detect OnExit output failure . . . 591
Detect errors during Sequencer configuration change . . . 592
Redundancy and the Sequencer object . . . 592
Sequencer object help . . . 592

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Configuration object attributes . . . 593
Step program . . . 593
Aliases . . . 594
Settings . . . 594
Settings . . . 595
Alarms . . . 596
Locking and security . . . 597
Run-time object attributes . . . 599
Execution attributes . . . 599
Current attributes . . . 605
Program attributes . . . 608
Selected attributes . . . 613
Alarm attributes . . . 617
History attributes . . . 619
Sequencer program XML schema . . . 620
General information . . . 620
XML roots and sub-roots . . . 620
About steps . . . 620
Step/jump condition syntax . . . 621
Condition type . . . 621
OnExit output flag . . . 622
Timer preset . . . 622
Trigger expression . . . 622
Examples of conditions . . . 622
Alias configuration . . . 622
Settings . . . 623
Example XML . . . 623
Sequencer state transition tables . . . 624
Terminology . . . 624
Current execution state: Running . . . 625
Current execution state: RunningSingleStep . . . 625
Current execution state: SingleStepTransitionReady . . . 626
Current execution state: RunningHeld . . . 627
Current execution state: Stopped . . . 628
Current execution state: StoppedComplete . . . 629
Current execution state: StoppedError . . . 629
Error codes for run-time updates . . . 630
SQLDataGrid Graphic . . . 632
The SQLDataGrid graphic . . . 632
Configure authentication modes . . . 633
Windows authentication . . . 633
SQL Server . . . 634
Configure the database connection . . . 634
Configure SQLDataGrid graphic custom properties . . . 634
Scripts with the SQLDataGrid . . . 636
Associate grid control actions with InTouch scripts and animations . . . 636
Access data from a selected row in the SQLDataGrid . . . 637
The SQLDataGrid in runtime . . . 637
View data . . . 638

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Populate the Table, View, or Query list . . . 638
Hide and show window issues . . . 638
Move columns . . . 639
Grouping Columns . . . 639
View aggregate values . . . 641
Progressive column filtering . . . 642
Discard changes . . . 645
View errors . . . 645
Write changes to the database . . . 645
Views and updates . . . 645
Database connection management . . . 647
Pool and shares connections . . . 647
Custom properties for the SQLDataGrid graphic . . . 648

Deploy . . . 656
Deployment overview . . . 656
Deploy objects from the IDE to run time . . . 657
Allocate run-time resources . . . 658
Allocate galaxy repository node memory . . . 658
Object, area, engine, and platform assignments . . . 659
Allocate AppEngines to platforms . . . 660
Data Integration Objects . . . 662
Auto-build . . . 662
Determine galaxy status . . . 662
Configure advanced communication management . . . 663
Select advanced communication management . . . 664
Configure scan modes . . . 664
Simulation server . . . 666
Simulation server and the OPCClient object . . . 667
I/O auto assignment in the simulation server . . . 667
Configure the simulation server . . . 668
View simulated data . . . 669
Deploy objects . . . 670
Deployment error messages . . . 674
Publish managed InTouch applications . . . 674
Redeploy objects . . . 675
Redeploy the WinPlatform object . . . 675
Retentive attribute properties and behavior . . . 677
Undeploy objects . . . 678
Upload run-time configuration changes . . . 678
Upload limitations . . . 679
Undeployment situations . . . 680
Associate all galaxy graphics with an InTouchViewApp . . . 680
Publish and deploy behavior when all galaxy graphics are associated with an InTouchViewApp . . . 681
Configure the associate galaxy graphics option . . . 682

Run . . . 683

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Process late data at run time . . . 683
Object Viewer overview . . . 683
Object viewer availability . . . 684
Start object viewer . . . 684
Navigate in object viewer . . . 684
Find objects . . . 685
Close object viewer . . . 686
View attributes . . . 687
View object attribute data . . . 687
Quality . . . 687
Status . . . 687
Has buffer . . . 688
Failed requests and errors . . . 689
Communication errors . . . 690
Configuration errors . . . 690
Operational errors . . . 691
Sort data . . . 692
View attributes in a watch window . . . 693
Aveva OPC UA client service . . . 693
Add a watch window . . . 693
Save a watch window . . . 694
Remove attributes from a watch window . . . 694
Modify attribute values . . . 694
Attribute properties . . . 695
Category property . . . 695
Value property . . . 695
Timestamp property . . . 696
Modify an attribute value . . . 696
Modify an attribute from the watch window . . . 696
View changes dynamically . . . 697
Set an object OnScan . . . 697
Set an object OffScan . . . 698

Manage . . . 699

Manage product licensing and user authorization . . . 699
Connected experience and operations control . . . 699
Configure operations control and connected experience . . . 699
Authentication and entitlement . . . 705
Authenticate with the AVEVA Identity Manager . . . 706
Behavior if connection is lost . . . 710
Single Sign-On (SSO) . . . 711
License mode . . . 712
Start and run the IDE . . . 712
Connect to a galaxy . . . 713
Connect to a galaxy with no security configured . . . 713
Connect to a galaxy not using Operations Control connected experience (configuration mismatch) . . . 714
Sign out of the IDE . . . 714

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Start Application Manager . . . 715
Launch a ViewApp using the connected experience . . . 715
Configure user permissions for connected experience . . . 716
Launch ViewApp in the OMI Web Client using connected experience . . . 717
Sign out from an OMI ViewApp . . . 718
Switch OMI ViewApp users . . . 719
Secured and verified writes . . . 719
Deploy scenarios . . . 719
Change Operations Control connected experience mode . . . 720
Using the OCMC with connected experience . . . 721
Galaxy migration to support connected experience . . . 721
Manage objects . . . 721
Check out objects . . . 722
Check in objects . . . 722
Validate objects . . . 722
Validate scripts and other external components . . . 723
Validate an object manually . . . 724
Create an instance . . . 724
Object names . . . 725
Rename an Object . . . 726
Delete an object . . . 726
Export objects . . . 727
Protect objects on export . . . 728
About protecting objects on export . . . 728
Export objects as protected . . . 729
Export objects with I/O auto assignment . . . 729
Export script function libraries . . . 729
Exporting Objects with Linked Content . . . 730
Import objects . . . 730
Import protected objects . . . 732
Import objects with I/O auto assignment . . . 733
Import objects with linked content . . . 734
Import a SQLData object . . . 734
Import script function libraries . . . 734
Import client controls . . . 735
Import an app from the AVEVA digital exchange . . . 736
Import AVEVA apps and client controls (ASUG) . . . 736
Add a control from an AVEVA app to a ViewApp . . . 737
Configurable properties in AVEVA apps . . . 737
AVEVA app development guidelines . . . 739
AVEVA app limitations and restrictions . . . 740
Troubleshoot AVEVA or WPF apps that fail to import . . . 741
Import HTML5 widgets . . . 741
Carousel widget . . . 742
QR code scanner . . . 743
Web browser widget . . . 743
After you import . . . 744
The multi-galaxy environment . . . 744
About the multi-galaxy environment . . . 745

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Set up multi-galaxy communication . . . 745
Galaxy names in a multi-galaxy environment . . . 748
Unpair galaxies . . . 748
Rename a galaxy . . . 749
Rename paired nodes . . . 749
Access multiple galaxies . . . 749
Use the galaxy browser with multiple galaxies . . . 750
Use Object Viewer with multiple galaxies . . . 750
Use InTouch with multiple galaxies . . . 751
Use PCS with multiple galaxies . . . 752
Guidelines to access to multiple galaxies . . . 752
Security in a multi-galaxy environment . . . 753
Pair galaxies with multiple network interface cards . . . 753
Prevent PCS services disruptions . . . 753
Work with IDE extensions on a GR node . . . 753
Alarms in a multi-galaxy environment . . . 754
AVEVA Alarm Control function and results . . . 754
Alarm Viewer Control function and results . . . 755
Alarm DB View Control function and results . . . 755
Enhance multi-galaxy performance . . . 755
Fix performance issues after the fact . . . 755
Scale the multi-galaxy environment for optimum performance . . . 756
Sett remote galaxy references in the owning object property . . . 756
Platform common services . . . 757
About platform common services (PCS) . . . 758
Define service discovery . . . 758
Define user-configurable PCS services . . . 758
Configure and deploy PCS services . . . 759
Create an instance of a service . . . 759
Assign the service instance to a node . . . 760
Deploy the instance . . . 760
Configure service TCP ports . . . 760
Configure the ASBGRBrowsing service . . . 761
Configure the ASBMxDataProvider service . . . 762
Configure the ASBAuthentication service . . . 763
IOMBLSService . . . 763
Configure and deploy the OPC UA service . . . 763
Client access rules and galaxy security . . . 766
Manage galaxies in a multi-galaxy environment . . . 768
Uninstall/reinstall Application Server . . . 768
Back up a galaxy . . . 768
Delete a galaxy . . . 769
Restore a galaxy . . . 769
Restore to an existing galaxy . . . 769
Restore o a new galaxy . . . 770
Upgrade a galaxy . . . 770
Migrate a galaxy . . . 770
Upgrade SQL Server . . . 770
Use the OS Configuration utility to fix communication issues . . . 770

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Enable buffered data **771**
Components that use buffered data 772
Configure buffered data for an attribute 773
Use object viewer with buffered data 774
Use scripts to process buffered data 776
Script basic functions 776
Sample script and output 776
Scripting tips and best practices 777
Use asynchronous scripts for buffered data 777
Buffered data script syntax and configuration 777
BindTo calls for buffered data 778
Buffered data run-time behavior 779
About buffered data and history 780
About buffered data and alarms and events 781
AppEngine reconnect configuration for objects linked to the telemetry server 781
Alarm functions and buffered data 782
Alarm and event types and buffered data 784
Work with references **786**
Message exchange and attributes 787
Reference strings 787
Relative references 788
Property references 788
Handle time zones with the time property 789
Preserve time stamps from the publishing source 790
Arrays 790
Format reference strings 790
Literals 791
Notes 792
View object attributes in the IDE 792
View references and cross references 794
Find objects 795
Use galaxy references in InTouch 797
Telemetry server data references 800
Telemetry server dynamic tags 801
Dynamic data references vs. static data references 801
Dynamic tag prerequisites 802
Naming rules 802
Configuration errors 802
DNP3 data references 803
DNP3 analog input and output points 804
DNP3 binary input and output points 805
DNP3 double-bit binary input points 806
DNP3 counter input points 807
DNP3 string input points 807
DNP3 Pulse Output Points 808
DNP3 pulse (trip/close) output points 808
IEC 60870-5 dynamic data references 809
IEC60870 analog input and output points 810
IEC60870 digital input and output points 811

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Modbus data references . . . 812
Modbus analog input and output points . . . 813
Modbus digital input and output points . . . 814
Modbus string points . . . 815
Telemetry server static data references . . . 816
Manage galaxies . . . 817
Back up and restore galaxies . . . 817
Change galaxies . . . 817
Delete a galaxy . . . 818
Galaxy object components synchronization . . . 819
Definitions . . . 819
Typical out-of-sync scenarios . . . 819
Use the synchronization feature . . . 819
Exportg instances to a CSV file . . . 820
Edit the exported instance data . . . 820
About the CSV file structure . . . 820
Host attributes . . . 821
About quotation marks and carriage returns . . . 821
Time formats in Excel . . . 821
Enable I/O auto assignment in the exported CSV file . . . 822
Import objects from a CSV file . . . 823
Synchronizing time across a galaxy . . . 824
Time synchronization in Windows domains . . . 824
Synchronization schedule . . . 824
Configure the time master . . . 824
Host multiple galaxies in a galaxy repository . . . 825
Disk space requirements . . . 825
Manage communication between galaxy nodes . . . 826
Map network drives with user account control enabled . . . 826
About the System Platform network account . . . 826
Multiple network interface cards . . . 828
Define the NIC order . . . 828
Configure IP address and DNS settings . . . 828
Configure multiple NICs . . . 828
Redundancy . . . 831
Redundancy overview . . . 831
Configure AppEngine redundancy . . . 832
Redundancy during run time . . . 833
Engine restart after failure . . . 833
CPU load balancing . . . 834
AppEngine Redundancy . . . 835
Configure the redundancy message channel . . . 835
Enable AppEngine redundancy . . . 836
Configure redundancy in the AppEngine template . . . 838
Delete redundant AppEngines . . . 838
Deploy AppEngine objects . . . 839
Configuration requirements . . . 839
Undeploy AppEngine objects . . . 840
Redundant AppEngines during deployment . . . 840

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Objects at run time . . . 841
Redundant AppEngines during run time . . . 841
AppEngine redundancy states . . . 841
Troubleshoot redundant AppEngines . . . 842
Generate alarms . . . 843
Generate history . . . 844
Data acquisition redundancy . . . 845
Configure data acquisition redundancy . . . 845
Deploy redundant DIObjects . . . 845
Data acquisition redundancy during run time . . . 846
RedundantDIObject and PLC connectivity . . . 846
Change network account utility . . . 846
Node-to-node communications . . . 846
Use the change network account utility . . . 846
Change the password of the current account . . . 848
Create a new local user account . . . 848
Use an existing local machine or network domain account . . . 848
Error messages . . . 849
Select a language . . . 852
Determine the change network account utility version . . . 852
Galaxy Database Manager . . . 852
Operations control management console . . . 852
Start the galaxy database manager . . . 853
Navigate in the galaxy manager . . . 853
Console tree . . . 853
Details pane . . . 854
Use the galaxy database manager . . . 854
Back up a galaxy . . . 854
Restore a galaxy . . . 855
Determine the galaxy database manager version . . . 858
Common Platform Services . . . 858
License Mode . . . 858
System Management Server . . . 860
System Management Server overview . . . 860
Install System Management Server . . . 860
Redundant SSO server . . . 861
Work with Configurator . . . 862
Connect a machine to a System Management Server . . . 862
Configure the System Management Server . . . 865
Run products without a System Management Server . . . 867
Advanced Configuration . . . 867
Certificates tab . . . 868
Ports tab . . . 870
Communications tab . . . 871
Authentication tab . . . 874
Federated Identity Provider . . . 874
Troubleshooting connection problems . . . 875
Configure a multi-user development environment . . . 876
Development system architecture . . . 876

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Local GR nodes plus shared GR (recommended) . . . 877
Create a multi-user development system . . . 879
Development process . . . 880
Production galaxy updates . . . 880
Benefits of a distributed multi-user development system . . . 881
Shared GR with local IDE nodes (alternative architecture) . . . 881
Shared GR and IDE node (alternative architecture) . . . 882
Managing Security for Application Server . . . 883
General considerations for security . . . 884
Secure the host . . . 884
General guidelines for securing the host . . . 885
Windows updates . . . 885
Configuring Automatic Updates . . . 886
Scanning the Host . . . 886
Protecting the applications and content on the host . . . 887
Secure the network . . . 888
Segment the ICS network . . . 889
Manage network services and ports . . . 890
Secure communication between the client and server . . . 891
Secure systems through authentication and authorization . . . 892
Manage users and groups through ICS software . . . 892
Manage users and groups through windows . . . 893
Contingency planning . . . 893
Business continuity planning . . . 894
Disaster recovery planning . . . 894
Security configuration for InTouch HMI . . . 895
Security configuration for AVEVA Application Server . . . 896
Data protection in Application Server and System Platform . . . 897

Diagnose . . . 899
Platform Manager . . . 899
Platform Manager . . . 899
About Platform Manager . . . 900
Start Platform Manager . . . 900
Navigate in the Platform Manager . . . 900
Security . . . 901
Galaxy authentication . . . 901
OS user-based authentication . . . 902
OS group-based authentication . . . 902
Authentication providers mode . . . 902
Change users . . . 902
About Object Viewer . . . 902
Use Platform Manager . . . 903
Platform and engine status . . . 903
Object status description . . . 903
On scan/off scan states . . . 904
Checkpointed state . . . 904
View platform status . . . 905

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

View engine status . . . 906
Export a list . . . 907
Manage platforms . . . 907
Sort platforms . . . 908
Set a platform on scan . . . 908
Set a platform off scan . . . 908
Start a platform . . . 908
Shut down a platform . . . 909
Remove a local platform . . . 910
Restart redundant platforms . . . 910
Manage engines . . . 912
Engine status icons . . . 912
Redundancy state descriptions . . . 913
Set an engine on scan . . . 915
Set an engine off scan . . . 915
Start an engine . . . 915
Shut down an engine . . . 916
Galaxy Repository diagnostic generation . . . 916
aaGR diagnostic utility (aadr.EXE) . . . 917
Troubleshoot a remote galaxy connection . . . 917
Connect to a remote GR from the IDE . . . 924
Object Viewer advanced diagnostic information . . . 925

Scripting Reference . . . 928

Common script environment . . . 928
Script editing styles and syntax . . . 928
Required syntax for expressions and scripts . . . 928
Simple scripts . . . 929
Script execution types . . . 929
Startup scripts . . . 932
OnScan scripts . . . 937
Execute scripts . . . 939
OffScan scripts . . . 942
Shutdown scripts . . . 944
Deployment scripts . . . 948
QuickScript editor features . . . 948
Color indicators for script elements . . . 948
Autocomplete . . . 949
Accept autocomplete suggestions . . . 953
Multi-level undo and redo . . . 953
Dynamic reference considerations . . . 953
Run-time client script behavior . . . 956
Open a client application window . . . 956
Close a client application window . . . 956
Minimize a client application window . . . 956
Maximize or restore a client application window . . . 957
Visual indication of script errors . . . 957
Line numbers . . . 957

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Log functions . . . 957
QuickScript .NET functions . . . 958
Script functions . . . 958
aaMethods . . . 958
Graphic client functions . . . 958
EmbedContent() . . . 958
GetCPQuality() . . . 964
GetCPTimeStamp() . . . 965
GetReferences() . . . 965
SetCustomPropertyValue() . . . 967
HideContent() . . . 969
HideGraphic() . . . 974
HideSelf() . . . 975
Logoff() . . . 975
RemoveEmbeddedContent() . . . 976
ShowContent() . . . 979
ShowGraphic() . . . 985
ShowLoginDialog() . . . 995
InTouch functions . . . 996
AddPermission() function . . . 996
AttemptInvisibleLogon() function . . . 998
ChangePassword() function . . . 999
EnableDisableKeys() function . . . 1000
FileCopy() function . . . 1001
FileDelete() function . . . 1002
FileMove() function . . . 1003
FileReadFields() function . . . 1004
FileReadMessage() function . . . 1005
FileWriteFields() function . . . 1005
FileWriteMessage() function . . . 1006
GetAccessToken() Function . . . 1007
GetAccountStatus() function . . . 1008
GetNodeName() function . . . 1009
GetSecureAccessToken() Function . . . 1010
GetTokenConnectionStatus() function . . . 1011
InfoAppTitle() function . . . 1012
InfoDisk() function . . . 1012
InfoFile() function . . . 1013
InfoInTouchAppDir() function . . . 1014
InTouchVersion() function . . . 1014
InvisibleVerifyCredentials() function . . . 1015
IsAssignedRole() function . . . 1016
LaunchTagViewer() function . . . 1017
LogonCurrentUser() function . . . 1017
PlaySound() Function . . . 1018
PostLogonDialog() function . . . 1019
PrintScreen() Function . . . 1020
QueryGroupMembership() function . . . 1020
ShowHome() function . . . 1021

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Start a Windows application . . . 1021
SwitchDisplayLanguage() function . . . 1022
TseGetClientId() function . . . 1023
TseGetClientNodeName() function . . . 1023
TseQueryRunningOnClient() function . . . 1023
TseQueryRunningOnConsole() function . . . 1024
Math functions . . . 1024
Abs() . . . 1025
ArcCos() . . . 1025
ArcSin() . . . 1026
ArcTan() . . . 1027
Cos() . . . 1028
Exp() . . . 1028
Int() . . . 1029
Log() . . . 1029
Log10() . . . 1030
LogN() . . . 1031
Pi() . . . 1031
Round() . . . 1032
Sgn() . . . 1033
Sin() . . . 1033
Sqrt() . . . 1034
Tan() . . . 1034
Trunc() . . . 1035
Miscellaneous functions . . . 1036
ActivateApp() . . . 1036
Filtering events . . . 1037
DateTimeGMT() . . . 1038
IsBad() . . . 1038
IsGood() . . . 1039
IsInitializing() . . . 1040
IsUncertain() . . . 1040
IsUsable() . . . 1041
LogCustom() . . . 1042
LogDataChangeEvent() . . . 1043
LogError() . . . 1044
LogMessage() . . . 1044
LogTrace() . . . 1045
LogWarning() . . . 1046
SendKeys() . . . 1047
SetAttributeVT() . . . 1049
SetAttributeVT2() . . . 1049
SetBad() . . . 1050
SetGood() . . . 1051
SetInitializing() . . . 1052
SetUncertain() . . . 1052
SignedAckAll() . . . 1053
SignedAlarmAck() . . . 1057
SignedWrite() . . . 1062

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

WriteStatus() . . . 1066
WWControl() . . . 1067
OPC UA Methods . . . 1068
Introduction to OPC UA scripts . . . 1068
Function . . . 1069
Client class . . . 1069
GenericStruct class . . . 1070
GenericField class . . . 1071
Value class . . . 1071
ValueType enumerations . . . 1072
MethodArgument class . . . 1073
MethodCallStatus class . . . 1073
MethodCallState enumeration . . . 1074
Sample Application Server script . . . 1075
CallMethodAsync example . . . 1076
How the script library handles MethodArgument values . . . 1077
Use complex (structured) data . . . 1077
Limitations . . . 1079
String functions . . . 1080
DText() . . . 1080
StringASCII() . . . 1080
StringChar() . . . 1081
StringCompare() . . . 1082
StringCompareNoCase() . . . 1083
StringFromGMTTimeToLocal() . . . 1085
StringFromIntg() . . . 1086
StringFromReal() . . . 1087
StringFromTime() . . . 1089
StringFromTimeLocal() . . . 1090
StringInString() . . . 1092
StringLeft() . . . 1093
StringLen() . . . 1094
StringLower() . . . 1095
StringMid() . . . 1096
StringReplace() . . . 1098
StringRight() . . . 1099
StringSpace() . . . 1101
StringTest() . . . 1102
StringToIntg() . . . 1103
StringToReal() . . . 1104
StringTrim() . . . 1105
StringUpper() . . . 1107
Text() . . . 1108
WWStringFromTime() . . . 1109
Security . . . 1110
GetNamedCredentials() . . . 1110
System functions . . . 1111
CreateObject() . . . 1111
Now() . . . 1111

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

WWDDE functions . . . 1112
WWExecute() . . . 1112
WWPoke() . . . 1113
WWRequest() . . . 1114
QuickScript .NET operators . . . 1115
Parentheses ( ) . . . 1117
Negation (-) . . . 1117
Complement (~) . . . 1118
Power (**) . . . 1118
Multiplication (*), division (/), addition (+),subtraction (-) . . . 1118
Modulo (MOD) . . . 1118
Shift left (SHL), shift right (SHR) . . . 1118
Bitwise AND (&) . . . 1119
Exclusive OR (^) and inclusive OR (|) . . . 1119
Assignment (=) . . . 1120
Comparisons (<, >, <=, >=, ==, <>) . . . 1120
AND, OR, and NOT . . . 1120
QuickScript .NET variables . . . 1120
Numbers and strings . . . 1122
QuickScript .NET control structures . . . 1123
IF … THEN … ELSEIF … ELSE … ENDIF . . . 1123
IF … THEN … ELSEIF … ELSE … ENDIF and attribute quality . . . 1125
FOR … TO … STEP … NEXT loop . . . 1126
FOR EACH … IN … NEXT . . . 1126
TRY ... CATCH . . . 1127
WHILE loop . . . 1128
Sample QuickScript .NET scripts . . . 1128
Use an imported type library to access an Excel spreadsheet . . . 1129
Use CreateObject to access an Excel spreadsheet . . . 1129
Call a web service to get the temperature in a Zip code . . . 1130
Call a web service to send an e-mail . . . 1130
Create a look-up table and do a look-up on it . . . 1130
Create an XML document and save it to disk . . . 1130
Execute a SQL parameterized INSERT command . . . 1131
Fill and use a string array . . . 1132
Fill and use a two-dimensional integer array . . . 1132
Format a Number with a .NET format 'Picture' . . . 1132
Format a time with a .NET format 'Picture' . . . 1132
Get the directories under the C drive . . . 1133
Load an XML Document from disk and do look-ups on it . . . 1133
Query a SQL Server database . . . 1133
Read a performance counter . . . 1133
Read a text file from disk . . . 1134
Share a SQL connection or other .NET object . . . 1134
Use DDE to access an Excel spreadsheet . . . 1134
Use Microsoft Exchange to send an e-mail message . . . 1134
Use screen-scraping to get the temperature for a city . . . 1135
Use SMTP to send an e-mail message . . . 1135
Write a text file to disk . . . 1135

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Dynamically bind an indirect variable to a reference . . . 1136
Bind to off-engine attributes . . . 1137
SQLData Script Library . . . 1138
Import and access the SQLData Script Library . . . 1138
SQLData Script Library interface . . . 1139
SQLData Script Library architecture . . . 1139
SQLData Script Library work flow . . . 1141
Create a connection object with a command . . . 1141
Create a connection object with a transaction . . . 1141
Specify connection strings . . . 1142
Connect to a SQL Server data source . . . 1142
Connect to an Oracle data source . . . 1142
Connect to Microsoft Access through OLEDB . . . 1142
Connect to Microsoft Excel through OLEDB . . . 1142
Example scripts . . . 1143
Overview of sample scripts . . . 1143
Detailed description of sample scripts . . . 1143
Asynchronous command script . . . 1144
Query script configuration . . . 1144
Process script configuration . . . 1144
Query script code . . . 1144
Process script code . . . 1145
Synchronous transaction script . . . 1146
QueryandProcess script code . . . 1146
aaDBAccess object . . . 1147
Create a reusable connection object . . . 1147
Create a unique connection object . . . 1148
Differences between CreateConnection() and GetConnection() . . . 1148
Work with Connections . . . 1148
Windows integrated security . . . 1148
Windows account . . . 1149
SQL Server authentication . . . 1149
Methods . . . 1149
CreateConnection() . . . 1149
Connecting to databases other than SQL Server . . . 1150
GetCommand() . . . 1150
GetConnection() . . . 1151
Authentication . . . 1152
Connecting to databases other than SQL Server . . . 1152
GetDiagnostics() . . . 1152
GetTransaction() . . . 1153
LogDiagnostics() . . . 1153
RemoveCommand() . . . 1153
RemoveTransaction() . . . 1154
ResetDiagnostics . . . 1154
Shutdown() . . . 1155
aaDBCommand object . . . 1155
Methods . . . 1155
AddRow() . . . 1156

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

DeleteCurrentRow() . . . 1156
Dispose() . . . 1156
ExecuteAsync() . . . 1157
ExecuteAsyncCancel() . . . 1157
ExecuteSync() . . . 1157
GetCurrentRowColumnByIndex() . . . 1158
GetCurrentRowColumnByName() . . . 1158
GetDataSet() . . . 1159
GetId() . . . 1159
GetParameterByIndex() . . . 1160
GetParameterByName() . . . 1160
GetRow() . . . 1161
SaveChangesAsync() . . . 1161
SaveChangesSync() . . . 1162
SelectRow() . . . 1162
SelectTable() . . . 1162
SetCurrentRow() . . . 1163
SetCurrentRowColumnByIndex() . . . 1164
SetCurrentRowColumnByName() . . . 1165
SetParam type methods for SQL Server and Oracle . . . 1165
Output parameters . . . 1166
SetBitParameterByName() . . . 1166
SetCharParameterByName() . . . 1167
SetDecimalParameterByName() . . . 1168
SetDateTimeParameterByName() . . . 1169
SetDoubleParameterByName() . . . 1170
SetFloatParameterByName() . . . 1171
SetIntParameterByName() . . . 1171
SetLongParameterByName() . . . 1172
SetParam type methods for OLEDB . . . 1173
SetBitParameterByIndex() . . . 1173
SetCharParameterByIndex() . . . 1174
SetDateTimeParameterByIndex() . . . 1175
SetDecimalParameterByIndex() . . . 1176
SetDoubleParameterByIndex() . . . 1177
SetFloatParameterByIndex() . . . 1178
SetIntParameterByIndex() . . . 1178
SetLongParameterByIndex() . . . 1179
Properties . . . 1180
CommandTimeout . . . 1180
CurrentRowNumber . . . 1180
CurrentTableNumber . . . 1180
Disposed . . . 1181
ExecutionState . . . 1181
LastExecutionError . . . 1181
RowCount . . . 1181
Public enumerations . . . 1182
aaDBCommandState . . . 1182
Created . . . 1182

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Queued . . . 1182
Failed . . . 1182
Completed . . . 1182
Canceled . . . 1182
Disposed . . . 1182
aaDBCommandType . . . 1183
sqlStatement . . . 1183
storedProcedure . . . 1183
aaDBParameterDirection . . . 1183
Input . . . 1183
InputOutput . . . 1183
Output . . . 1183
ReturnValue . . . 1183
aaDBConnection object . . . 1183
Connection pooling . . . 1184
Methods . . . 1184
CreateCommand() . . . 1184
CreateCommand for Oracle . . . 1185
CreateCommand for OLEDB . . . 1185
CreateTransaction() . . . 1185
Dispose() . . . 1186
GetDiagnostics() . . . 1186
LogDiagnostics() . . . 1186
ResetDiagnostics . . . 1187
Properties . . . 1187
ConnectionName . . . 1187
ConnectionState . . . 1187
Disposed . . . 1188
LastError . . . 1188
Public enumerations . . . 1188
aaDBConnectionState . . . 1188
Disconnected . . . 1188
Connecting . . . 1188
Connected . . . 1189
Created . . . 1189
Disposed . . . 1189
aaDBConnectionType . . . 1189
Sql . . . 1189
OleDb . . . 1189
Oracle . . . 1189
aaDBRow object . . . 1189
Methods . . . 1190
GetColumnName() . . . 1190
GetColumnValue() . . . 1190
Properties . . . 1190
ColumnCount . . . 1191
ColumnNames . . . 1191
ColumnValues . . . 1191
aaDBTransaction object . . . 1191

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Methods . . . 1192
CreateCommand() . . . 1192
Dispose() . . . 1192
ExecuteAsync() . . . 1192
ExecuteAsyncCancel() . . . 1193
ExecuteSync() . . . 1193
GetID() . . . 1194
Properties . . . 1194
Disposed . . . 1195
ExecutionState . . . 1195
FailedCommandID . . . 1195
LastExecutionError . . . 1195
Public enumeration . . . 1195
aaDBTransactionState . . . 1196
Created . . . 1196
Queued . . . 1196
Failed . . . 1196
Completed . . . 1196
Canceled . . . 1196
Disposed . . . 1196
Error codes . . . 1196

API and extensions reference . . . 1200
GRAccess Toolkit . . . 1200
How to use the GRAccess Toolkit . . . 1200
Target clients . . . 1200
Microsoft Visual Studio . . . 1200
Configuration files . . . 1200
GRAccess object model . . . 1201
GRAccess methods to set license mode . . . 1203
API reference . . . 1203
COM classes . . . 1203
GRAccess . . . 1203
Interfaces implemented . . . 1203
GRAccessApp . . . 1204
Interfaces implemented . . . 1204
MxValue . . . 1204
Interfaces implemented . . . 1204
Interfaces . . . 1204
IAssociatedFileAccess . . . 1205
IAttribute . . . 1205
IAttribute class members . . . 1205
CommandResult property . . . 1206
RtSethandler property . . . 1206
CfgSethandler property . . . 1207
SecurityClassification property . . . 1207
Name property . . . 1208
DataType property . . . 1208

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

SetLocked method . . . 1209
UpperBoundDim1 property . . . 1209
SetSecurityClassification method . . . 1210
Locked property . . . 1210
SetValue method . . . 1211
Value property . . . 1211
AttributeCategory property . . . 1212
HasBuffer property . . . 1212
SetHasBuffer method . . . 1213
IAttributes . . . 1213
IAttributes class members . . . 1213
Item property . . . 1214
Count property . . . 1215
ShortDescription property . . . 1215
ExecutionOrder property . . . 1216
SecurityGroup property . . . 1216
ExecutionRelatedObject property . . . 1216
ICommandResult . . . 1217
ICommandResult Class Members . . . 1217
Successful property . . . 1218
Text property . . . 1218
ID property . . . 1218
CustomMessage property . . . 1219
ICommandResults . . . 1219
ICommandResults class members . . . 1219
Count property . . . 1219
Item method . . . 1220
CompletelySuccessful property . . . 1220
ICondition . . . 1221
ICondition class members . . . 1221
Kind property . . . 1221
Value property . . . 1222
Negation property . . . 1222
IConditions . . . 1222
IConditions class members . . . 1222
CommandResult property . . . 1223
Remove method . . . 1223
Add method . . . 1224
Count property . . . 1225
Item property . . . 1225
Join method . . . 1225
IGalaxy . . . 1226
IGalaxy class members . . . 1226
CommandResult property . . . 1228
QueryObjectsMultiCondition method . . . 1229
QueryScriptLibraries method . . . 1229
Name property . . . 1230
Backup method . . . 1230
VersionString property . . . 1231

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Login method . . . 1232
LoginEx method . . . 1233
LoginWithAccessMode method . . . 1234
SynchronizeClient method . . . 1234
GetTimeMasterSettings method . . . 1235
CreategObjectCollection method . . . 1236
ImportObjects method . . . 1236
ImportObjectsEx method . . . 1237
ImportObjectsEx2 method . . . 1237
ImportArchestrAApp method . . . 1238
QueryToolsets method . . . 1239
QueryObjectsByName method . . . 1239
GetUserDefaults method . . . 1240
VersionNumber property . . . 1241
SetUserDefaults method . . . 1241
Restore method . . . 1242
Logout method . . . 1243
UpgradeRequired property . . . 1243
MigrateGalaxy method . . . 1244
CdiVersionString property . . . 1245
GetSecuritySettings method . . . 1245
CreateConditionsObject method . . . 1246
QueryObjects method . . . 1246
ImportScriptLib method . . . 1247
ImportScriptLibEx method . . . 1248
GetLocaleSettings method . . . 1248
ExportAll method . . . 1249
GetReadOnlySecurity method . . . 1250
CommandResults property . . . 1250
GRLoad method . . . 1251
GetFlexLicensesInformation method . . . 1251
IGalaxyQueryToolsetsEx method . . . 1253
IGalaxies . . . 1253
IGalaxies class members . . . 1253
Item property . . . 1254
Count property . . . 1254
IGalaxyRole . . . 1255
IGalaxyRole class members . . . 1255
RoleName property . . . 1255
AccessLevel property . . . 1256
Permissions property . . . 1256
OperationalPermissions property . . . 1256
IGalaxyRoles . . . 1257
IGalaxyRoles class members . . . 1257
Item property . . . 1257
Count property . . . 1258
IGalaxySecurity . . . 1258
IGalaxySecurity class members . . . 1258
AuthenticationMode property . . . 1259

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

SecurityGroupsAvailable property . . . 1259
RolesAvailable property . . . 1260
UsersAvailable property . . . 1260
LoginTime property . . . 1261
RoleUpdateInterval property . . . 1261
IGalaxyUser . . . 1262
IGalaxyUser class members . . . 1262
UserName property . . . 1262
FullName property . . . 1263
AssociatedRoles property . . . 1263
IGalaxyUsers . . . 1264
IGalaxyUsers class members . . . 1264
Item property . . . 1264
Count property . . . 1264
IGraphicAccess . . . 1265
About programmatic graphic export and import . . . 1265
The IGraphicAccess API . . . 1266
GraphicAccess interface methods . . . 1267
About programmatic string and reference export and import . . . 1268
The IGraphicAccess2 API . . . 1268
GraphicAccess2 interface methods . . . 1269
Programmatic GPI export . . . 1270
The IGraphicAccess3 API . . . 1270
GraphicAccess3 interface methods . . . 1271
About programmatic import from a folder location . . . 1271
The IGraphicAccess4 API . . . 1271
GraphicAccess4 interface methods . . . 1272
IgObject . . . 1273
IgObject class members . . . 1273
Tagname property . . . 1275
ContainedName property . . . 1276
HierarchicalName Property . . . 1276
Attributes property . . . 1277
ConfigurableAttributes property . . . 1277
ValidationStatus property . . . 1278
Errors property . . . 1278
Warnings property . . . 1279
CheckedOutBy property . . . 1279
CheckoutStatus property . . . 1280
ConfigVersion Property . . . 1280
Category property . . . 1281
DerivedFrom property . . . 1281
BasedOn property . . . 1282
Container property . . . 1282
Area property . . . 1283
Host property . . . 1284
GetObjectHelpURL method . . . 1284
Save method . . . 1285
CheckIn method . . . 1285

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

CheckOut method . . . 1286
UndoCheckOut method . . . 1286
Unload method . . . 1286
EditStatus property . . . 1287
AddExtensionPrimitive method . . . 1287
DeleteExtensionPrimitive method . . . 1288
RenameExtensionPrimitive method . . . 1289
AddUDA method . . . 1289
DeleteUDA method . . . 1290
RenameUDA method . . . 1291
UpdateUDA method . . . 1291
CommandResult property . . . 1292
CategoryGUID property . . . 1293
GetExtendedAttributes method . . . 1293
PlatformLicenseType property . . . 1294
IgObjects . . . 1295
IgObjects class members . . . 1295
CommandResults property . . . 1296
Area property . . . 1297
Host property . . . 1297
AddFromCollection method . . . 1298
Upload method . . . 1298
CheckIn method . . . 1299
Add method . . . 1300
Container property . . . 1300
ExportObjects method . . . 1301
ExportObjectsAsProtected method . . . 1301
DeleteAllObjects method . . . 1302
CheckOut method . . . 1302
SecurityGroup property . . . 1303
Undeploy method . . . 1303
Deploy Method . . . 1304
Item property . . . 1305
Count property . . . 1305
UndoCheckOut method . . . 1306
DeployEx method . . . 1306
UndeployEx method . . . 1307
UploadEx method . . . 1307
IGRAccess . . . 1308
IGRAccess class members . . . 1308
CommandResult property . . . 1309
CreateGalaxy method . . . 1309
DeleteGalaxy method . . . 1310
QueryGalaxies method . . . 1311
QueryGalaxiesEx method . . . 1311
CreateGalaxyFromTemplate method . . . 1312
IInstance . . . 1313
IInstance class members . . . 1313
AddExtensionPrimitive method . . . 1315

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

AddUDA Method . . . 1316
Area property . . . 1316
Attributes property . . . 1317
BasedOn property . . . 1317
Category property . . . 1318
CategoryGUID property . . . 1318
CheckedOutBy property . . . 1319
CheckIn method . . . 1319
CheckOut method . . . 1320
CheckoutStatus property . . . 1320
CommandResult property . . . 1320
ConfigurableAttributes property . . . 1321
ConfigVersion property . . . 1321
ContainedName property . . . 1322
Container property . . . 1322
DeleteExtensionPrimitive method . . . 1323
DeleteInstance method . . . 1323
DeleteUDA method . . . 1324
Deploy method . . . 1324
DeployedVersion property . . . 1325
DeployEx method . . . 1326
DeploymentStatus property . . . 1327
DerivedFrom property . . . 1327
EditStatus property . . . 1328
Errors property . . . 1328
GetExtendedAttributes method . . . 1328
GetObjectHelpURL method . . . 1329
HierarchicalName property . . . 1330
Host property . . . 1330
RenameExtensionPrimitive method . . . 1330
RenameUDA method . . . 1331
Save method . . . 1331
Tagname property . . . 1332
Undeploy method . . . 1332
UndeployEx method . . . 1333
UndoCheckOut method . . . 1334
Unload method . . . 1334
UpdateUDA method . . . 1335
Upload method . . . 1335
UploadEx method . . . 1336
ValidationStatus property . . . 1337
Warnings property . . . 1337
IMxValue . . . 1338
IMxValue class members . . . 1338
Clone method . . . 1340
Empty method . . . 1341
PutBoolean method . . . 1341
PutInteger method . . . 1342
PutFloat method . . . 1342

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

PutDouble method . . . 1343
PutString method . . . 1344
PutTime method . . . 1344
PutElapsedTime method . . . 1345
PutMxReference method . . . 1346
PutMxStatus method . . . 1346
PutMxDataType method . . . 1347
PutMxSecurityClassification method . . . 1347
PutMxDataQuality method . . . 1348
PutCustomStruct method . . . 1349
PutCustomEnum method . . . 1349
GetDataType method . . . 1350
GetBoolean method . . . 1351
GetInteger method . . . 1352
GetFloat method . . . 1352
GetDouble method . . . 1353
GetString method . . . 1354
GetTime method . . . 1355
GetElapsedTime method . . . 1356
GetMxReference method . . . 1356
GetMxStatus method . . . 1357
GetMxDataType method . . . 1358
GetMxSecurityClassification method . . . 1358
GetMxDataQuality method . . . 1359
GetCustomStruct method . . . 1360
GetCustomEnum method . . . 1361
GetDimensionCount method . . . 1362
PutElement method . . . 1362
GetElement method . . . 1363
GetDimensionSize method . . . 1364
PutCustomStructVB method . . . 1364
GetCustomStructVB method . . . 1365
PutInternationalStrings method . . . 1365
PutInternationalStringsVB method . . . 1366
GetInternationalStrings method . . . 1366
GetInternationalStringsVB method . . . 1367
GetInternationalString method . . . 1368
PutInternationalString method . . . 1369
IPermission . . . 1369
IPermission Class members . . . 1370
PermissionName property . . . 1371
PermissionParentName property . . . 1371
IsConfigured property . . . 1372
HasChildren property . . . 1372
ChildPermissions property . . . 1373
IsSecurityGroup property . . . 1373
SecurityGroup property . . . 1373
IPermissions . . . 1374
IPermissions class members . . . 1374

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Item property . . . 1374
Count property . . . 1375
IScriptLibrary . . . 1375
IScriptLibrary class members . . . 1375
CommandResult property . . . 1375
Export method . . . 1376
Name property . . . 1376
IScriptLibraries . . . 1377
IScriptLibraries class members . . . 1377
CommandResult property . . . 1377
Item property . . . 1378
Count property . . . 1378
Add method . . . 1379
ISecurityGroup . . . 1379
ISecurityGroup class members . . . 1379
GroupName property . . . 1380
gObjects property . . . 1380
ISecurityGroups . . . 1381
ISecurityGroups class members . . . 1381
Item property . . . 1381
Count property . . . 1382
ISettings . . . 1382
ISettings class members . . . 1382
CommandResult property . . . 1383
Cancel method . . . 1383
Instance property . . . 1383
Close method . . . 1384
ITemplate . . . 1384
ITemplate class members . . . 1384
AddExtensionPrimitive method . . . 1387
AddUDA method . . . 1387
Area property . . . 1388
Attributes property . . . 1389
BasedOn property . . . 1389
Category property . . . 1389
CategoryGUID property . . . 1390
CheckIn method . . . 1390
CheckOut method . . . 1391
CheckedOutBy property . . . 1391
CheckoutStatus property . . . 1392
CommandResult property . . . 1392
CommandResults property . . . 1392
ConfigurableAttributes property . . . 1393
ConfigVersion property . . . 1393
ContainedName property . . . 1394
Container property . . . 1394
CreateInstance method . . . 1395
CreateInstances method . . . 1396
CreateTemplate method . . . 1396

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

DeleteExtensionPrimitive method . . . 1397
DeleteTemplate method . . . 1398
DeleteUDA method . . . 1398
DerivedFrom property . . . 1399
EditStatus property . . . 1399
Errors property . . . 1400
GetExtendedAttributes method . . . 1400
GetObjectHelpURL method . . . 1401
HierarchicalName property . . . 1401
Host property . . . 1402
RenameExtensionPrimitive method . . . 1402
RenameUDA method . . . 1403
Save method . . . 1403
Tagname property . . . 1404
Toolset property . . . 1404
UndoCheckOut method . . . 1405
Unload method . . . 1405
UpdateUDA method . . . 1406
ValidationStatus property . . . 1407
Warnings property . . . 1407
IToolset . . . 1407
IToolset class members . . . 1408
CommandResult property . . . 1408
GetChildToolsets method . . . 1408
MoveToToolset method . . . 1409
Name property . . . 1409
Rename method . . . 1410
IToolsets . . . 1410
IToolsets class members . . . 1410
Add method . . . 1411
AddToolSet method . . . 1411
DeleteToolSet method . . . 1412
Item property . . . 1412
Count property . . . 1413
Type definitions . . . 1413
EATTRIBUTEPROPERTY . . . 1414
EActionForCurrentlyDeployedObjects . . . 1417
EAuthenticationMode . . . 1418
EAutomaticallyUndocheckout . . . 1418
EBASERUNTIMEOBJECT . . . 1418
ECATEGORY . . . 1419
ECOMMONATTRIBUTES . . . 1420
ECOMMONPRIMITIVE . . . 1420
ECascade . . . 1422
ECheckoutStatus . . . 1423
EConditionType . . . 1423
EDeployOnScan . . . 1425
EDeploymentStatus . . . 1426
EEXECUTIONGROUP . . . 1426

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

EEditStatus . . . 1427
EExecutionOrder . . . 1428
EExportType . . . 1428
EFileType . . . 1429
EForceDeleteInstanceOption . . . 1429
EForceDeleteTemplateOption . . . 1430
EForceOffScan . . . 1430
EGRCommandResult . . . 1431
EMatch . . . 1433
EPACKAGESTATUS . . . 1434
ERESERVEDPRIMITIVEIDS . . . 1434
ESkipIfCurrentlyUndeployed . . . 1434
ESkipObjectsWithPendingUpdates . . . 1435
ESkipOtherUsersCheckedOutObjects . . . 1435
ETestOnly . . . 1436
EUserDefault . . . 1436
EgObjectIsTemplateOrInstance . . . 1437
MxAttributeCategory . . . 1437
MxPropertyLockedEnum . . . 1439
GRAccess code examples . . . 1439
Programming tips . . . 1440
Example: define the entry point for a console application . . . 1440
Example: create and populate a galaxy . . . 1443
IDE Extensions development guide . . . 1445
Introduction . . . 1445
Organization . . . 1446
Terms and definitions . . . 1446
Overview . . . 1447
Get started with IDE Extensions . . . 1448
IDE Extensions manifest . . . 1449
Discover IDE Extensions . . . 1450
Load IDE Extensions . . . 1450
Instantiate IDE Extensions . . . 1450
Create an IDE Extension . . . 1451
Create a command extension . . . 1452
Create a view extension . . . 1452
IDE object type system . . . 1453
Validate, package, and deploy IDE Extensions . . . 1454
Validate the extension manifest . . . 1454
Package IDE Extensions . . . 1455
Deploy the IDE Extension assemblies . . . 1455
Test IDE Extensions . . . 1455
Extend IDE specifics . . . 1455
Command extensibility . . . 1455
Add a new command . . . 1456
Extend the IDE user interface . . . 1456
Extend IDE menus and menu items . . . 1457
Extend context menus . . . 1459
Extend the IDE toolbar . . . 1459

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Add command accelerators . . . 1460
Extend the IDE status bar . . . 1460
Override a command . . . 1461
View extensibility . . . 1461
IDE services . . . 1461
ArchestrA/AVEVA configuration services . . . 1462
Galaxy configuration . . . 1463
Galaxy object information . . . 1463
User interface services . . . 1464
Register and invoke commands . . . 1464
Invoke/update other commands . . . 1465
Status bar service . . . 1465
General UI service . . . 1466
Progress notification service . . . 1466
Object icon service . . . 1466
Show context menus . . . 1467
Enumerate and activate IDE views . . . 1467
Tips for IDE Extension development . . . 1468
Override predefined IDE commands . . . 1469
Commands that can be overridden . . . 1470
Commands that cannot be overriden . . . 1471

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Welcome to Application Server

This document describes Application Server and the System Platform IDE, which comprises the configuration-time portion of Application Server. It is intended for developers who are familiar with object-oriented programming and others involved in the process of building System Platform applications.

Application Server is a central component of any System Platform installation, and includes a run-time element as well as the IDE used for configuration. Most of this document describes how to use the IDE to build System Platform applications.

With Application Server, you can:

- Create, configure, and maintain the objects that comprise your application.
- Implement security for the IDE and your application.
- Collaborate with other developers to build AVEVA OMI or managed InTouch applications.
- Deploy and run your applications.
- Monitor and troubleshoot your application in real time.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

What's new in Application Server?

These release notes describe the new and enhanced functionality available in Application Server and provide an overview of the most significant changes.

July 2024
June 2024
December 2023
July 2022
October 2021
December 2020
April 2020

July 2024

This update includes formatting changes to document headings and images, and some minor editorial changes to improve accessibility and readability.

June 2024

For the System Platform 2023 R2 P01 patch release, Application Server received a number of hot fixes and miscellaneous bug fixes.

December 2023

With the release of System Platform 2023 R2, Application Server received the following new features and enhancements:

Improved User Interface

The System Platform IDE has an improved user interface that provides object status at a glance.

New OMI Web Client

The web client for use with OMI ViewApps is completely new. The new client offers better performance, a look and feel more like the desktop client, and an easier way to make ViewApps available from a browser: you just assign them to an instance of the new WebViewEngine object type. The OMI web client is supported with the latest versions of the Google Chrome and Microsoft Edge browsers.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

OPC UA Scripting

You can now use Application Server object scripts and/or Industrial Graphics scripts to call OPC UA servers. Method calls are executed in the OPC UA server through the OPC UA client, and are supported through the OPC client in the OI Gateway Communication Driver.

Alarm Latching and Alarm Dismiss

Latching is a new alarm state that can be enabled for a Galaxy. When latching is enabled, an acknowledged alarm that has returned to a normal value will continue to be displayed in the LATCHED alarm state. Alarms go to the LATCHED state if you ACK an alarm from UNACK_RTN state or return an alarm from ACK state. LATCHED alarms continue to be displayed to show that the alarms did occur. A dismiss command will return alarms that are in the LATCHED state to normal.

July 2022

With the release of System Platform 2023, Application Server received the following new features and enhancements:

New look for Application Server

The System Platform IDE, the design environment for Application Server, now has a new look and improved user interface. Menus and icons have been redesigned for a better user experience. Some of the key highlights of the System Platform IDE user interface include a new backstage, menu items, and groups. All special operations that you need to access frequently are now in the backstage, on the Galaxy menu. Other commands have been regrouped in different menu items based on their functionality and ease of accessibility.

User authentication and single sign-on via Azure Active Directory

Azure Active Directory (AD) can be used for identity authentication and single sign-on. After setting Galaxy security to use an external identity provider, Azure AD can be used to authenticate users. System Platform users can then use their Microsoft-managed identities to log in to the System Platform IDE, Operations Control Management Console, AVEVA OMI ViewApps, and to perform secured and verified writes.

Integrated web help system

Help for Application Server is now available as web help, and can be viewed and searched in your default browser. Help for other System Platform products has also been migrated to web help, allowing you to more easily view and search across all System Platform products.

Alarm translation enhancements

You can now localize alarm and event message, and alarm groups.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Preview a ViewApp

System Platform includes the capability for developers to preview an OMI ViewApp from the ViewApp editor without deploying any associated assets. After selecting the Preview Web button, the ViewApp appears in the Web Client.

Object wizard performance

The time needed to propagate changes has been significantly improved, thus greatly increasing performance during development.

Symbol wizard integration

Symbol Wizard support has been improved in web graphics that appear in OMI web ViewApps. In the case of SAL symbols, Wizard Options are supported to provide different configurations of the same symbol.

Display a web ViewApp on a mobile device

The AVEVA Mobile Operations application has been updated to show OMI web ViewApps on a mobile device.

Enhanced management of cloud graphics

AVEVA Connect users are prevented from accessing or overwriting any cloud graphics that were created with a higher version of the Graphic Editor than their local editor. Since multiple AVEVA products use the Industrial Graphic Editor, this allows organizations to use these different products, which may incorporate different versions of the editor, while avoiding any issues that may result from attempting to edit a graphic with new extensions that do not exist in their local version of the editor.

Embed web widgets within OMI Industrial Graphics

Place one or more web widgets directly in an Industrial Graphic for use in the OMI desktop as well as the OMI web client.

GRAccess as an API

The GRAccess Toolkit is now delivered as an API with System Platform to allow users to programmatically interact with Application Server.

Security identity event logging

Log-on and log-off events, both successful and unsuccessful, are tracked for security and audit purposes.

Updates for historical playback capability

Source non-historized values during historical playback to avoid missing information. These values can optionally

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

be flagged to make the viewer aware of the alternative data source.

New BigString data type

The BigString data type supports a virtually unlimited string size to allow users to store and transmit huge strings.

Dynamic images in Industrial graphics

A new property, ImageURI, has been added to Industrial Graphics. This allows the user to modify the property at runtime to dynamically change the image.

AVEVA Flex subscription enhancements

Multiple Flex single engine licenses can now by assigned to a single platform.

Workspace feature for the AVEVA Operations Management Interface Web Client

Use the Workspace feature to create customizable dashboards for casual users across the enterprise. Workspace displays all graphics available in the Workspaces toolset of Graphic Toolbox in the selected application. This allows viewing an Industrial Graphic with operations data by binding attributes from AVEVA System Platform Enterprise. You can create, modify, and save your dashboards, and even share them with others.

Cybersecurity enhancements

Application Server and AVEVA OMI incorporate ongoing cybersecurity enhancements and hardening.

Extensions

- New Grid View Widget
- Content Presenter OMI App Update
- Map App Update

October 2021

With the release of System Platform 2020 R2 SP1, Application Server received the following new features and enhancements:

Warm Failover

Redundant AppEngines can be configured for warm failover. This greatly reduces the time and number of scan cycles needed before the backup AppEngine becomes functional, and thus greatly reduces any data lost during failover.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Alarm Message Translation

Alarm comment language switching (message translation) is now supported in the AVEVA OMI AlarmApp.

December 2020

With the release of System Platform 2020 R2, Application Server received the following new features and enhancements:

Alias Names

Alias Names are now first class citizens in the system.

Graphic Repeater App

The Graphic Repeater app allows a single graphic to be shown multiple times in an AVEVA OMI layout pane, with each occurrence using different values. Dynamic sorting of the graphic, based on the current run-time values, is supported.

Web Widgets

Web Widgets, previously supported only in InTouch HMI windows, are now supported in AVEVA layout panes.

Navigation Node Security

Navigation Node Security can now use roles.

Galaxy Style Libraries

Galaxy Style Libraries are supported and the Galaxy Style Library can be dynamically changed at runtime.

Support for Expressions in External Content Objects

External Content objects, which previously could use only static strings, now allow dynamic strings. These dynamic strings can be any combination of static strings, references, and complex expressions. The behavior of the dynamic content is determined at run time, and is the same as provided by the Industrial Graphic Editor.

Flex License Mode Configuration

Flex License mode configuration has been enhanced.

Layout Background Image

A background image can be set for an entire AVEVA OMI layout.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

GR Access Toolkit

The GR Access Toolkit has been updated to support Flex licensing.

Linked Layouts

Layouts can be used as content and linked to templates and assets.

Licensing

The Galaxy Repository process periodically retries obtaining a license if it cannot obtain one, or if it loses an acquired license.

April 2020

With the release of System Platform 2020, Application Server received the following new features and enhancements:

Performance Improvements

A number of enhancements to improve performance have been implemented for this release of System Platform.

- Propagation times for changes to templates and creating instances have been shortened, especially in large systems.
- Deployment times for InTouchViewApp instances have been reduced.
- The startup time for the aaGR process has been improved.

Credential Management for OMI ViewApps

OMI ViewApps can use security credentials created in Application Server to gain access to third-party data where Windows OS credentials are not recognized, and/or to provide a method to automatically load a particular data set for specified users.

Support for Additional .NET Controls OMI ViewApps

OMI now lets you import, configure, and use .NET WinForm controls in a ViewApp (.NET WPF controls were already supported).

Responsive Layouts for OMI ViewApps

The Layout Editor has a new option for creating responsive layouts. This allows the ViewApp to dynamically move and resize content in response to the screen size on which it is being viewed.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Script Support for OMI Layouts

Using the same foundation as scripting for Industrial Graphics, layout scripting lets you customize run time behaviors for layout content as well as the layout container by adding scripts. As part of layout support, graphic custom properties can be overridden within the layout. A new namespace, "MyContent," is available for addressing content within a single ViewApp, in addition to the MyViewApp global namespace.

Event Scripting for OMI Layouts

You can add event-based scripts to OMI layouts when the layout include apps that expose public properties, for example "MouseDown."

HideContent() Scripting Function and API Call

The HideContent() scripting function and API call can be used in conjunction with ShowContent() lets you place symbols or layouts into specific panes within an OMI layout, either through scripting or programmatically, and now HideContent() provides the additional functionality for more complete control of the content you wish to make available to users at run time.

OMI App Enhancements

The MapApp and Content Presenter App have been enhanced with new features and improved functionality.

- MapApp: Credential management has been improved, with a built-in menu for selecting a named credential.
- ContentPresenterApp: The ContentPresenter app shows graphics from one or more selected nodes of a ViewApp navigation model. The pane in which the ContentPresenter app has been placed shows a set of graphic thumbnails that match the ContentPresenter app filter criteria. Graphics shown by the ContentPresenter app can be filtered by content type, name, and location within the ViewApp navigation model.

Browse Application Server Run-Time Data with OPC UA Clients

AVEVA Application Server now includes OPC UA server functionality. This lets you browse the namespace in a deployed galaxy, and connect to, subscribe to, and retrieve galaxy data, without the need for a gateway or other protocol translation mechanism. See "Configuring and Using the OPC UA Server" in the Application Server User Guide for more information.

AVEVA Connect for Sharing Graphics

AVEVA Connect is a cloud-based resource that lets you easily create and share graphics between locations, galaxies, and applications. See "AVEVA Connect" in the Application Server User Guide for more information.

AVEVA Telemetry Server Enhancements

The Telemetry Server now supports handling data that arrives out-of-sequence via multiple channels. See "Telemetry Server Data References" in the Application Server User Guide for more information.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Visualize Externally-Detected Alarms

Alarm conditions that have been detected within a PLC can now be visualized in, for example, the Historical Trend App or the Alarm App. To accomplish this, the ability to configure a Boolean attribute within the Object Editor as an alarm input for a State Alarm has been added. This allows the trend tools to properly visualize alarms that have been detected externally. See "Using the State Alarm Feature" in the Application Server User Guide for more information.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

System Platform IDE

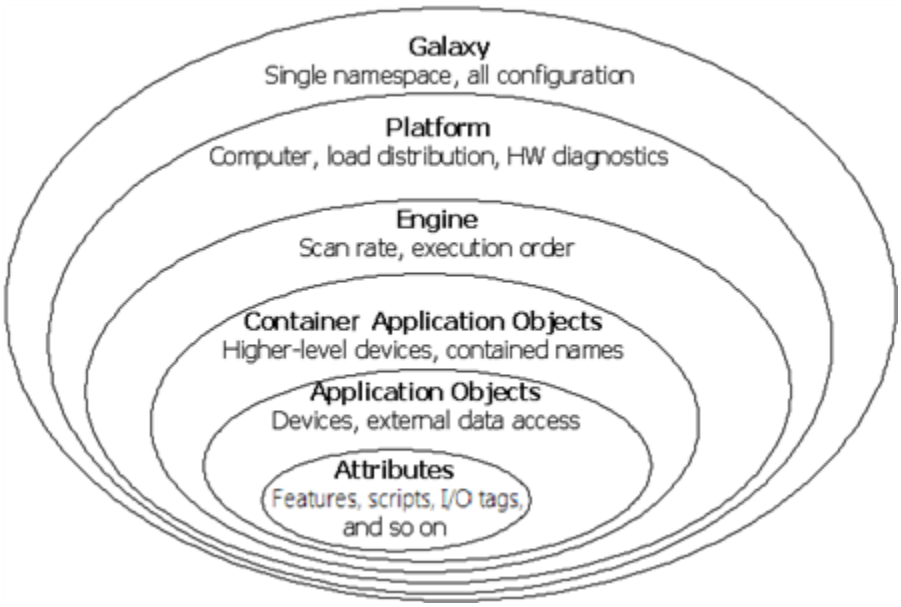
The System Platform IDE is the integrated design and development tool for the AVEVA™ Application Server.

The IDE lets you create, configure, and maintain the objects that comprise your application and the underlying infrastructure that supports your application. In Application Server, your application is referred to as a Galaxy and is stored in a database called the Galaxy Repository.

From the IDE, you can import new types of objects in to the Galaxy Repository, configure new ones, and deploy them to computers in your network. Multiple users can work concurrently on different sets of objects from different nodes running unique instances of the IDE.

Galaxy overview

A Galaxy is a collection of platforms, engines, templates, instances, and attributes you define as the parts of your specific application. Persistent information about this collection of objects is stored in a Galaxy database. Within Application Server, both templates and instances are collectively referred to as objects. For more information about what templates and instances are, see About templates and instances. A Galaxy consists of various objects that represents your entire production environment, including all computers and components that run your application.



A Galaxy Repository (GR) is the name of the single computer where the Galaxy database is located. A Galaxy database resides on a single network computer (the Galaxy Repository) and contains configuration details, including security and licensing information, for all assets that will be deployed at run time. A Galaxy database can reside on any computer on your network with SQL Server, Bootstrap, and Galaxy Repository software installed. A single Galaxy database cannot be split up among multiple computers. However, you can create multiple GR nodes and allow Galaxies to communicate with each other. For more information, see The multi-galaxy environment .

You can deploy Galaxy components, such as platforms and engines, on multiple computers to share the work

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

load while applications are running. For more information, see Deploy.

A Galaxy's namespace is the set of unique object and attribute identifiers. The namespace and the values of each of its identifiers define an Application Server application, and can be accessed by clients of the configuration system as well as the Application Server Message Exchange in a deployed system.

A key benefit of the Application Server namespace is that it allows Application Server objects and process data to be referenced by scripts and animation links from any computer in the Galaxy without the reference needing to specify the object's location.

Galaxies also include security, which is turned off by default. Using security allows you to limit what users can do. You can add more users, security roles, and security groups later if you want. For more information, see Configure security.

Start the System Platform IDE

Once the IDE has started, you can open an existing Galaxy or create a new Galaxy from one of four Galaxy templates, called Cab files. For more information, see Create a Galaxy.

If your installation is configured to use Operations Control connected experience as the license mode, you may need to log in when you start the IDE.

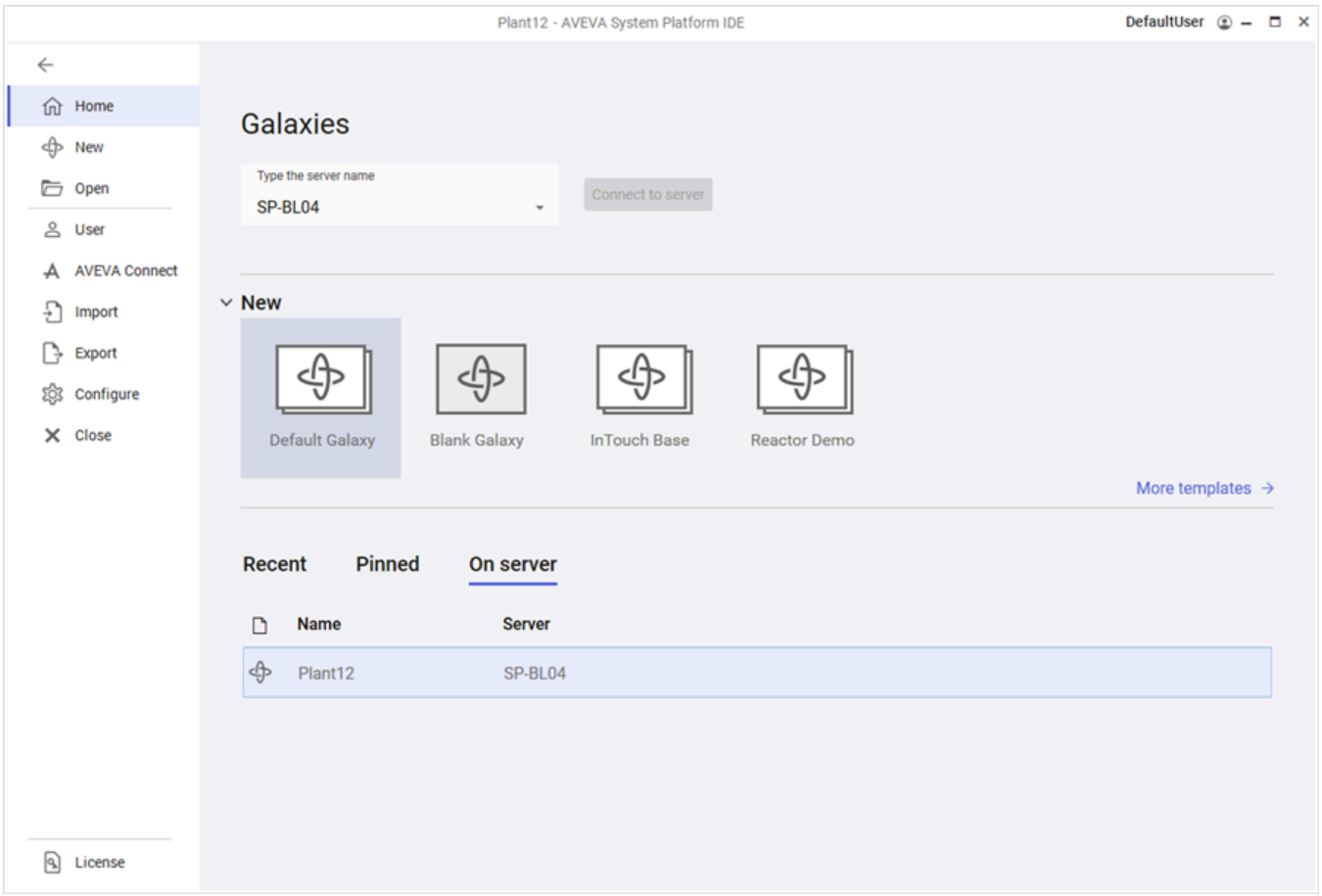
For more information about configuring the license mode, see the Configuration section of the *System Platform Installation Guide*.

For more information about logging in to the IDE, see Using the Conn Exp and AIM.

IDE backstage configuration options

The System Platform IDE contains a set of backstage options that lets you configure security, communications, alarms and events, and other parameters on a Galaxy-wide basis. In addition, the backstage lets you create or connect to a Galaxy, sign into CONNECT, import and export objects, change or view the Galaxy user, and set user preferences. Details about these functions and others are described within the procedures that use the various backstage functions. See Galaxy options for additional information.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.



Install and activate your license

Before you can open the IDE, you must have a valid license. For information about licensing, see "License Installation and Activation" in the *AVEVA System Platform Installation Guide*. If you need additional information, see the *AVEVA Enterprise Licensing Guide*.

If the Galaxy Repository process (aaGR) cannot obtain a license for any reason, it will send requests periodically to the license server. These attempts to acquire a license will continue until either a license is acquired successfully, or the aaGR process is stopped. This retry process also applies if access to a license is lost after it was acquired.

System Platform also supports AVEVA Operations Control and the connected experience. For more information, see Manage product licensing and user authorization.

Manage security certificate verification

The System Platform IDE may take 30 seconds to open on a system that is not connected to the Internet. When the IDE starts, the operating system attempts to verify the digital certificates for internal components against a Certificate Revocation List (CRL) located on a public website. If your system cannot access the public site within 30 seconds, the IDE startup process resumes and completes.

To maintain a better security profile by checking components against a CRL, keep your internet connection enabled and allow the verification to proceed.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

To avoid the potential delay, in **Internet Explorer** or through **Control Panel**, open **Internet Options**, **Advanced Options**, and uncheck the **Security** option to **Check for publisher's certificate revocation**.

AVEVA CONNECT

AVEVA Connect can be configured within the Visualization folder to provide centralized storage and access for reusable Industrial Graphics. You can use these graphics as templates to simplify building applications. For more information, see About using CONNECT.

IDE application views

Base templates and non-base templates are included with Application Server. The templates are automatically imported into the IDE from the Cab file you use to create a Galaxy.

Base templates appear in the Template folder and in the Derivation view with a $ as the first character of their name. You cannot directly modify base templates but you can use them to create your own objects or derived templates.

When you move from one view to another, the selected object in the first view is selected in the second view, if the object exists in that view. For example, templates are not shown in the Deployment view, Model view and IO Devices view.

You can open views from the **View** menu or by using keyboard shortcuts.

View	Shortcut
Model View	Ctrl+Shift+M
Deployment View	Ctrl+Shift+D
Derivation View	Ctrl+Shift+R
Template Folder	Ctrl+Shift+T
Visualization Folder	Ctrl+Shift+V
Operations View	Ctrl+Shift+O
IO Devices View	Ctrl+Shift+I
Mapping view	Ctrl+Shift+G

Application object icons

All IDE views (Templates, Visualization, Model, Deployment, Derivation, and IO Devices) allow you to choose whether to show objects in tree (hierarchical) view or in a list (flat) view. By default, objects are shown in a tree. To change to the view selection, click the three vertical dots at the top right of the view window. Default objects are shown in bold.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

For objects shown in any view, icons to the left of the object name help to identify the object type.

System object icons	
	Galaxy (project)
	GRPlatform
	WinPlatform
	Cloud Platform
	AppEngine
	Primary (1) Redundant AppEngine
	Backup (2) Redundant AppEngine
	Area
	Application Object (asset)
View engine icons	
	View Engine
	Webview Engine
ViewApp icons	
	OMI ViewApp
	InTouch ViewApp
Device integration icons	
	DDE SuiteLink Client InTouch Proxy OPC Client
	Redundant DI
	Scan group
Visualization icons	
	Screen profile

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

	Layout
	Symbol
	External content
	Namespace
	Client control
	OMI app
	Widget

Status Icons

Status icons are shown to the right of the object names. These icons are arranged in three columns are used to show validation state (first column), editing state (middle column), and deployment state (right column) of the objects.

Each status icon can also show a tooltip, identifying the icon. Hover your cursor over the icon to view the tooltip.

Validation Status	Editing Status	Deployment Status
No error	Checked in	Deployed
Good	Checked out by me	Not deployed
Warning	Checked out by other	Pending update
Error	Protected	Pending software update

Redundant AppEngine Validation Status

See AppEngine Redundancy for more information about working with redundant AppEngines.

	Both redundant AppEngines are not deployed
	This AppEngine is not deployed, but its redundant partner is deployed.
	This AppEngine is deployed, but its redundant partner is not deployed.
	Deployed, but the redundant AppEngines have a

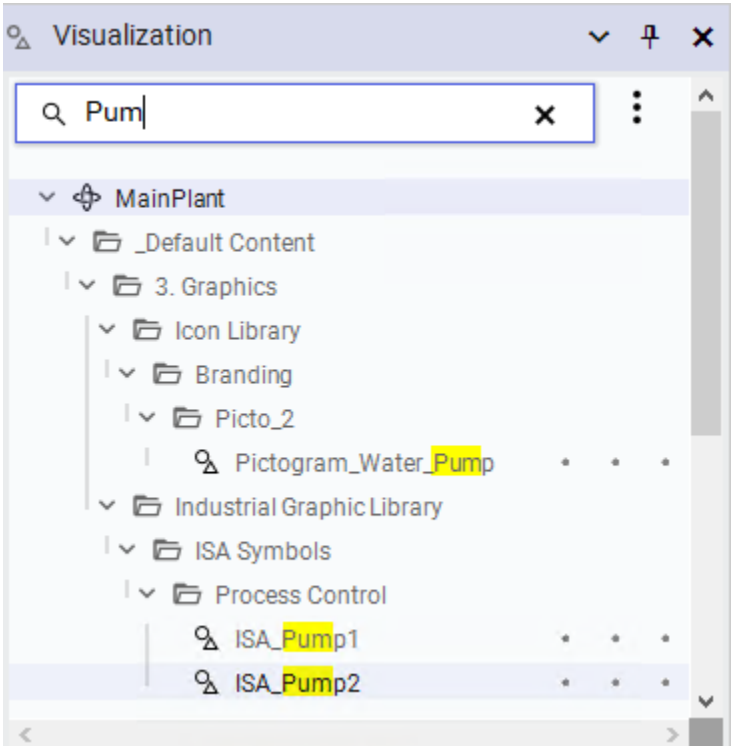
© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

	pending configuration update.
	This AppEngine has a pending configuration update, but its redundant partner is up-to-date.
	Both redundant AppEngines have pending software updates.
	This AppEngine has a pending software update, but its redundant partner is up-to-date.
	This AppEngine is up-todate, but its redundant partner has a pending software update.

Perform a search in a view

You can search in all IDE application views except for Mapping view and Operation view.

To search, just type the string to search for. As you type each character, the view expands the tree as needed to show items that match your entry, and highlights the matches.



Note: In Model view, the search includes either object tagnames or aliases, whichever is currently shown.

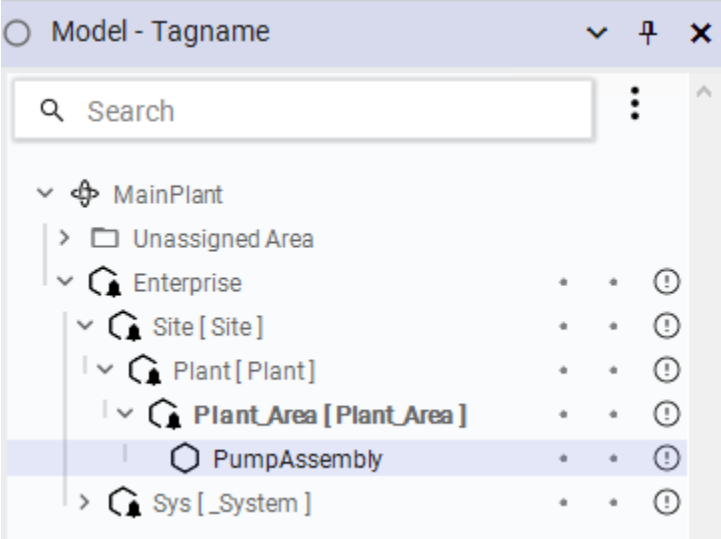
Model view

The Model view organizes your Galaxy by Area object assignment. It shows objects in terms of their physical or containment relationships, which can be organized through a folder structure. This Model view most accurately represents an application perspective of the processes that users are emulating: for instance, specific process

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

areas, tanks, valves, pumps and their relationships based on containment. For more information about containment, see Create contained templates.

Note: You must undeploy an object before reassigning it to another object.



The tree structure acts like a standard Microsoft Windows Explorer tree. Initially, it shows only the Galaxy name and the Unassigned Area folder.

In the Model view, all objects are grouped by areas and by containment relationship. The Model view shows these relationships in the following ways:

- The top of the tree is the Galaxy.
- Top-level Areas are shown under the Galaxy.
- Within each Area, contained Areas are listed. Areas support hierarchical composition; that is, they support sub-Area construction. Areas can be nested only 10 levels (after the sub-area is 10-levels deep, you cannot add another sub-level).
- Objects that belong to an Area are listed under the Area.
- Objects contained by other objects are listed under their respective containers. Multiple levels are allowed. For more information about containment, see Create contained templates.

  Note: Contained objects belong to the same Area as the object that contains them.

  Some objects' hierarchical, or contained, names are truncated if you have multiple levels shown. To view the entire hierarchical name, select the object and select **Properties** on the **Home** ribbon. The entire hierarchical name is shown in the **Properties** dialog box. For more information about hierarchical names, see Use contained names.
- Objects that currently do not belong to an Area are listed under Unassigned Area. Containment relationships between parent and child objects are shown there.
- In each branch of the tree, objects are listed in alphabetical order. Default objects are shown in bold.

⃠	To assign an object to another, drag it onto the host object. If the object is an inappropriate assignment match, the international Not symbol appears. To unassign an object, drag it to the **Unassigned Host**

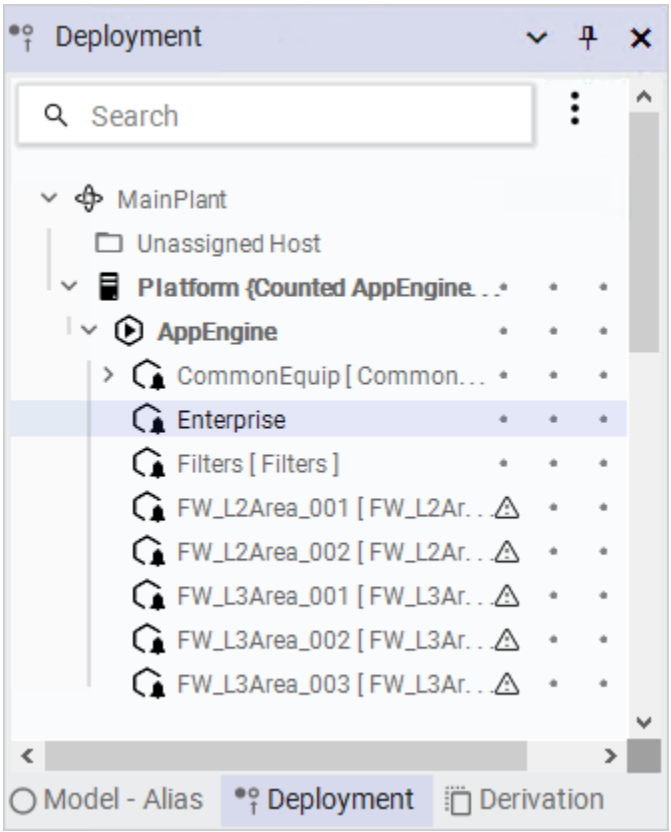
© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

folder.

Model view shows either the tagname or the alias name for each object. To change which is shown, right-click any object in the view and select **Alias** from the context menu.

Deployment view

The Deployment view shows which object instances reside on which computer (platforms), and AppEngine assignments to platforms. In the System Platform environment, the physical location of object instances is not required to approximate the real-world environment it models. The Deployment view does not need to reflect your physical plant environment. If you are using Flex licensing, your available licenses will help you determine how to assign AppEngines to platforms. Deployment view shows instances only in terms of their assignment relationships. This view enables you to organize those objects through a folder structure.



The tree structure acts like a standard Windows Explorer tree. It is initially divided into two levels: <Galaxy Name> and the Unassigned Host folder.

In the Deployment view, objects appear in a tree according to their distribution relationships in a multi-node system in the following ways:

- The top of the tree is the Galaxy.
- WinPlatforms are shown under the Galaxy.
- Under each WinPlatform, assigned AppEngines are listed.
- Under each AppEngine, assigned Areas and objects configured to communicate with PLCs, RTUs, etc. such as OPCClient objects, are listed.

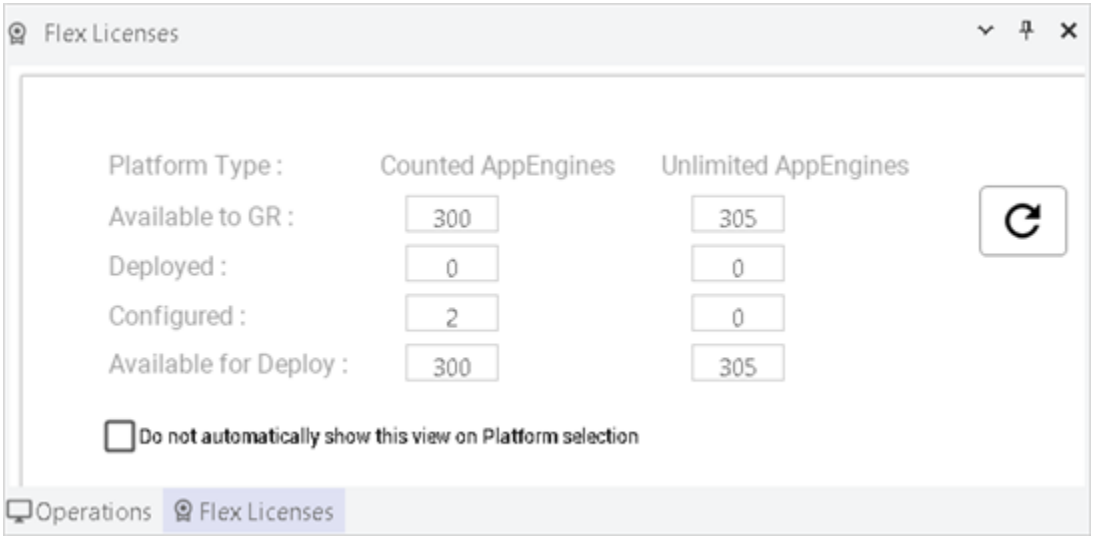
© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Note: DINetwork Objects, such as ABCIP and SIDIRECT, have been deprecated. Use an OPCClient object instead.

- Under each Area, assigned ApplicationObjects are listed.
- Under each ApplicationObject, contained ApplicationObjects are listed. Multiple levels are allowed.
- Unassigned objects are grouped together in the **Unassigned Host** folder. Area and containment relationships are maintained in this view.

Flex licenses view

Flex Licenses view is available only if your server is configured to use Flex licenses. Flex licenses are subscription-based licenses that can be used in place of traditional, perpetual licenses. When used with Application Server, Flex licenses are configured for Platform objects; each license allows one AppEngine to run on that Platform.. There are three choices when configuring Flex licenses for a Platform object: none, counted, and unlimited. See Allocate AppEngines to platforms for more information.



The **Flex Licenses** view shows how many licenses of each type (counted and unlimited) are available, configured, and deployed. The count is automatically updated before and after deployment or undeployment, upon connection to the galaxy, and whenever platform assignments are changed.

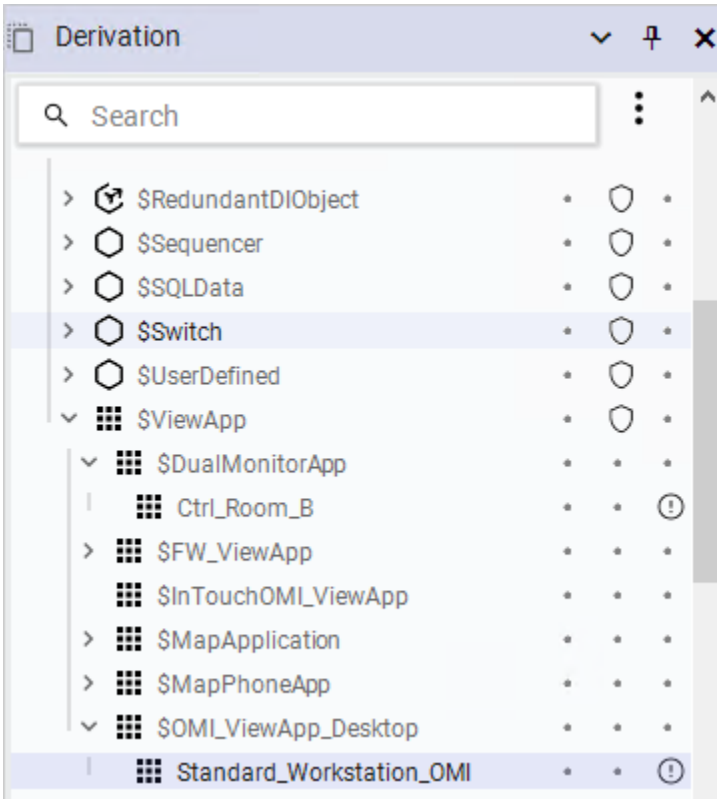
- **Available to GR**: The total number of counted AppEngine and unlimited AppEngine licenses available in the galaxy.
- **Deployed**: Licenses that have been deployed. You cannot deploy an AppEngine that does not have a license.
- **Configured**: The number of AppEngine licenses that have been assigned to a platform. Note that you can configure an AppEngine without a license, but you cannot deploy it. The platform will show a configuration error if there are not enough licenses.
- **Available for Deploy**: The number of licenses available to the galaxy minus the number of licenses configured on deployed platforms.

Derivation view

Derivation view shows objects and templates in terms of their parent/child relationship. An object derived from

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

another object appears in a hierarchy level under it.



The tree structure acts like a standard Windows Explorer tree, and initially is divided into three levels: <Galaxy Name>, <Used Base Templates>, and the **Unused Base Templates** folder.

In the Derivation view, objects appear according to their parent-child relationship in the following ways:

- The top of the tree is the Galaxy.
- Base templates with associated child objects, either derived templates or instances, are shown under the Galaxy.
- Under each base template, derived templates and instances created from the base template are listed. Multiple levels are allowed. Instances created from derived templates are listed under their parents.
- Templates with no associated derived templates or instances are grouped together in the **Unused Base Templates** folder.

Objects with names that start with a "$" are templates, either base or derived. Under each branch of the tree, child objects are listed in alphabetical order.

As in other views, dragging one object onto another in the **Derivation view** associates the two objects based on the predefined rules of the object types. For example, you can drag ApplicationObjects onto other ApplicationObjects but you cannot drag ApplicationObjects to an Engine.

IO devices view

The **IO Devices** view is used for I/O auto assignment and shows the relationship of system objects and application objects to scan groups and Device Integration objects. Before using this view, system and application objects must be configured to use I/O auto assignment by adding one or more attributes to the object and

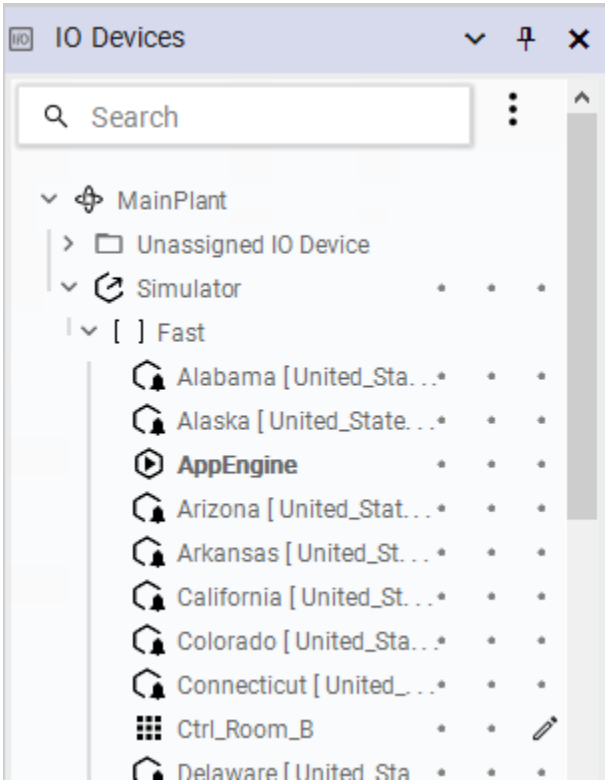
© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

activating the I/O feature. See Edit objects and I/O auto assignment for additional information.

You will also use the Object Editor to add scan groups to DI objects (OPCClient, DDESuiteLinkClient, or RedundantDIObject) before using the **IO Devices** view.

Once you have configured system and application objects for I/O auto assignment, and added scan groups to at least one DI object, you will assign the system and application objects to scan groups in the **IO Devices** view. This links auto-binds the system and application objects to PLCs, through the DI objects.

To open **IO Devices** view, select **IO Devices** on the **View** ribbon.



System and application objects appear as child objects to scan groups, and scan groups appear as child objects to DI objects.

Note: Only instances, and not templates, appear in the **IO Devices** view.

Device linkage, which is established by assigning a system or application object to a scan group, ensures that all of the object's attributes that have been prepared for automatic I/O assignment will match up with a corresponding input source or output destination in the scan group. The I/O references are then referred to as auto-bound.

Important: Objects must be undeployed before they can be assigned to a scan group.

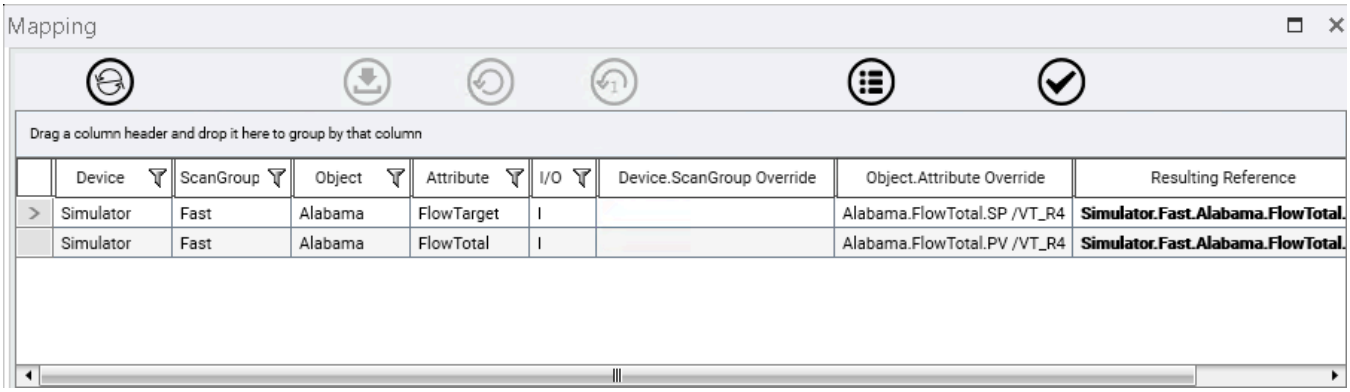
System and application objects that are not yet assigned to a scan group are placed under the folder "Unassigned IO Devices." To assign objects to a scan group, and thus allow I/O automatic assignment to occur, simply select objects in the Unassigned area, and drag and drop them to the applicable scan group. Once an object has been assigned to a scan group, the I/O reference for each I/O attribute is automatically configured.

Mapping view

The **Mapping** view shows I/O references for attributes that have been configured for I/O auto assignment, and that have been assigned to a scan group through the **IO Devices** view. To open this view, select **Mapping** on the

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

View ribbon. The **Mapping** view will also open if you select a DI object, scan group, or object assigned to a scan group in the **IO Devices** view.



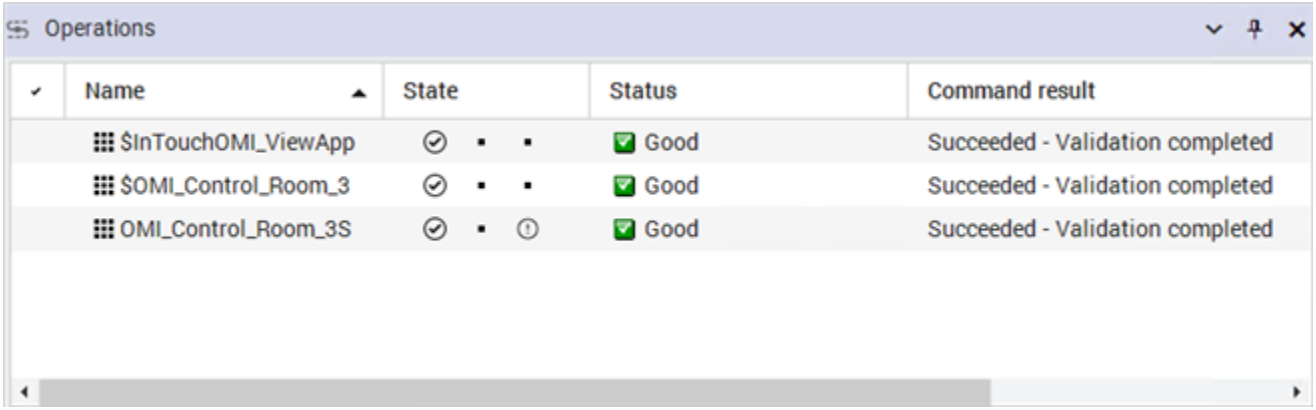
The **Mapping** view shows I/O auto assigned attributes with resolved I/O references for objects that have been assigned to scan groups. The **Mapping** view lets you enter override values to change the I/O reference. You can also validate the I/O references.

You select the references that are displayed in the **Mapping** view by selecting objects in the **IO Devices** view. Manually configured references are not displayed in the **Mapping** view, nor are attributes associated with objects that are not assigned to a scan group. Only I/O references for the object or group of objects currently selected in the **IO Devices** view are displayed. You can select DI objects, scan groups, and application and system objects in the **IO Devices** view. If only the top level item in a device hierarchy is selected, all subordinate items are automatically selected. If, however, you select a subordinate item along with the top level item, only I/O references for the selected items are displayed. You can select multiple items at different hierarchical levels.

Note: Selecting a subordinate object will exclude non-selected objects within the device hierarchy, even if the parent object is selected.

Operations view

The **Operations** view shows the results of validating the configuration of objects. You may need to open it before you see it. On the **View** ribbon, select **Operations**.



During the validation of an object, its icon and name appear with the state and status of the object and the status of the operation.

Important: You can validate both templates and instances if they are checked in.

- The icons in the **State** column show the current state of the object. See Application Object Icons for the

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

meaning of each icon.

- The status of the object (**Status** column) is shown with an icon and a descriptive word or phrase.

When validation is complete, the **Command Result** column shows a "Succeeded" or "Failed" message and additional information about the validation results. For more information about validating objects, see Validate objects.

Note: You can validate all objects in the Galaxy by running the Validate operation on the Galaxy object. In that case, Command Result messages are shown after all objects in the Galaxy are validated.

The **Operations view**, like the **Template folder** and **Applications views**, is also updated as the status and conditions of objects in the Galaxy change.

Keyboard shortcuts

The following keyboard shortcuts can be used, depending on what is currently selected in the IDE. Some shortcuts require that you have already selected one or more items from either the Template or Visualization folder. Other shortcuts exist only in more limited circumstances. For example, the Ctrl+N shortcut to create a copy of a service requires that the **Configure Services** dialog is open.

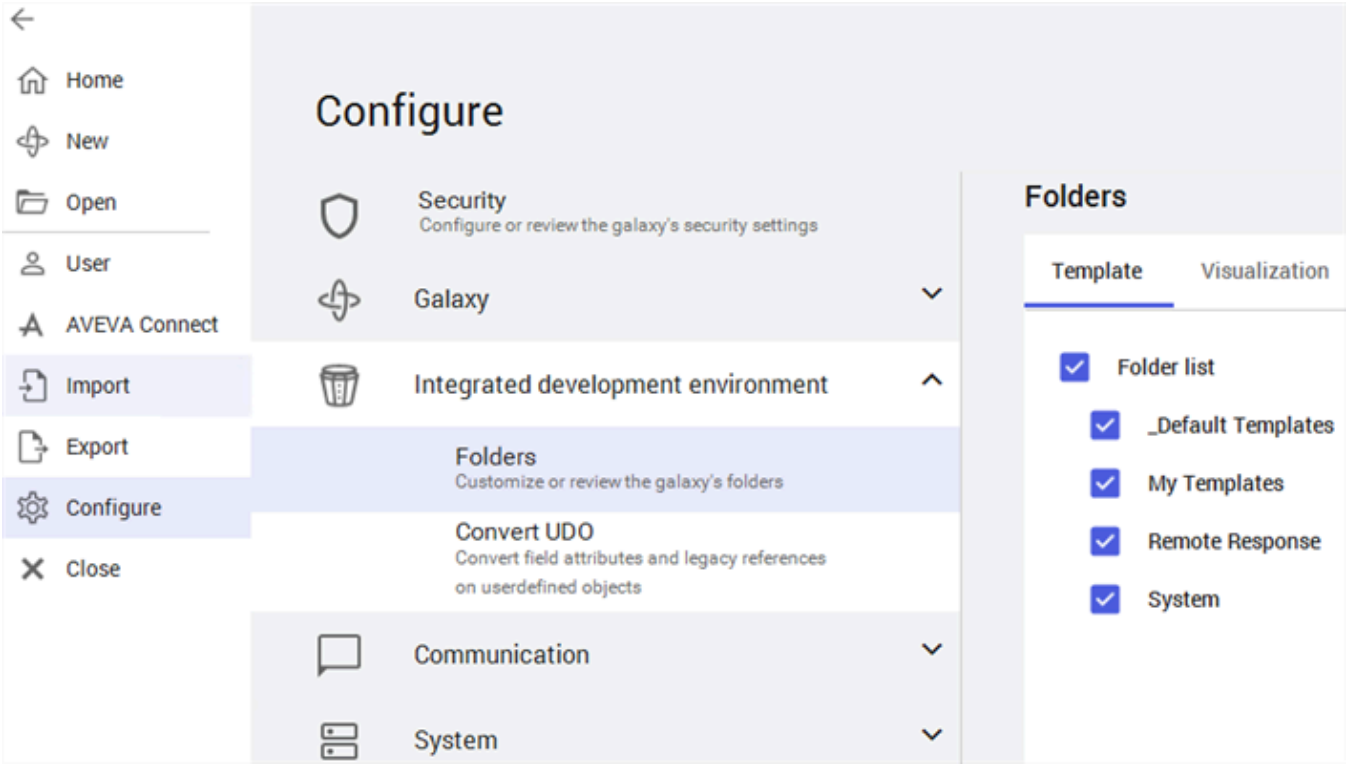
Keyboard Shortcut	Action
F1	Open the IDE help
F2	Rename the selected object
Alt+Enter	View the Properties of the selected object
Ctrl+F1	Open help for the selected object
Ctrl+D	Template Folder: Deploy the selected objects Visualization Folder: Duplicate the selected graphic
Ctrl+N	Create a copy of the selected service
Ctrl+Shift+C	Create a new External Content object
Ctrl+Shift+D	Open the IDE Deployment View
Ctrl+Shift+E	Create a new Screen Profile
Ctrl+Shift+G	Open the IDE Mapping View
Ctrl+Shift+H	Create a new Graphic object
Ctrl+Shift+I	Open the IDE IO Devices View
Ctrl+Shift+N	Create a new derived template
Ctrl+Shift+O	Open the IDE Operations View
Ctrl+Shift+V	Open the IDE Visualization Folder

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Ctrl+Shift+R	Open the IDE Derivation View
Ctrl+Shift+T	Open the IDE Template Folder
Ctrl+Shift+Z	Synchronize the IDE Views

Customize your workspace

You can dock, float or hide the different IDE views and folders to customize your workspace. You can also select which folders will be shown by default for the Galaxy by selecting **Galaxy** from the menu, then select **Configure**, and then navigate to **Folders** under **Integrated development environment**.



Dock views

To dock a view to a new location:

1. Place the cursor on the title bar of the view (or folder) and hold down the left mouse button.
2. As you drag the view from its position, blue directional arrows appear on the screen. While continuing to hold down the mouse button, position the cursor on one of the arrows. The position on the screen that correlates with the arrow will be highlighted.
3. Release the button when the area where you want to dock the view is highlighted. The view will snap to highlighted area.

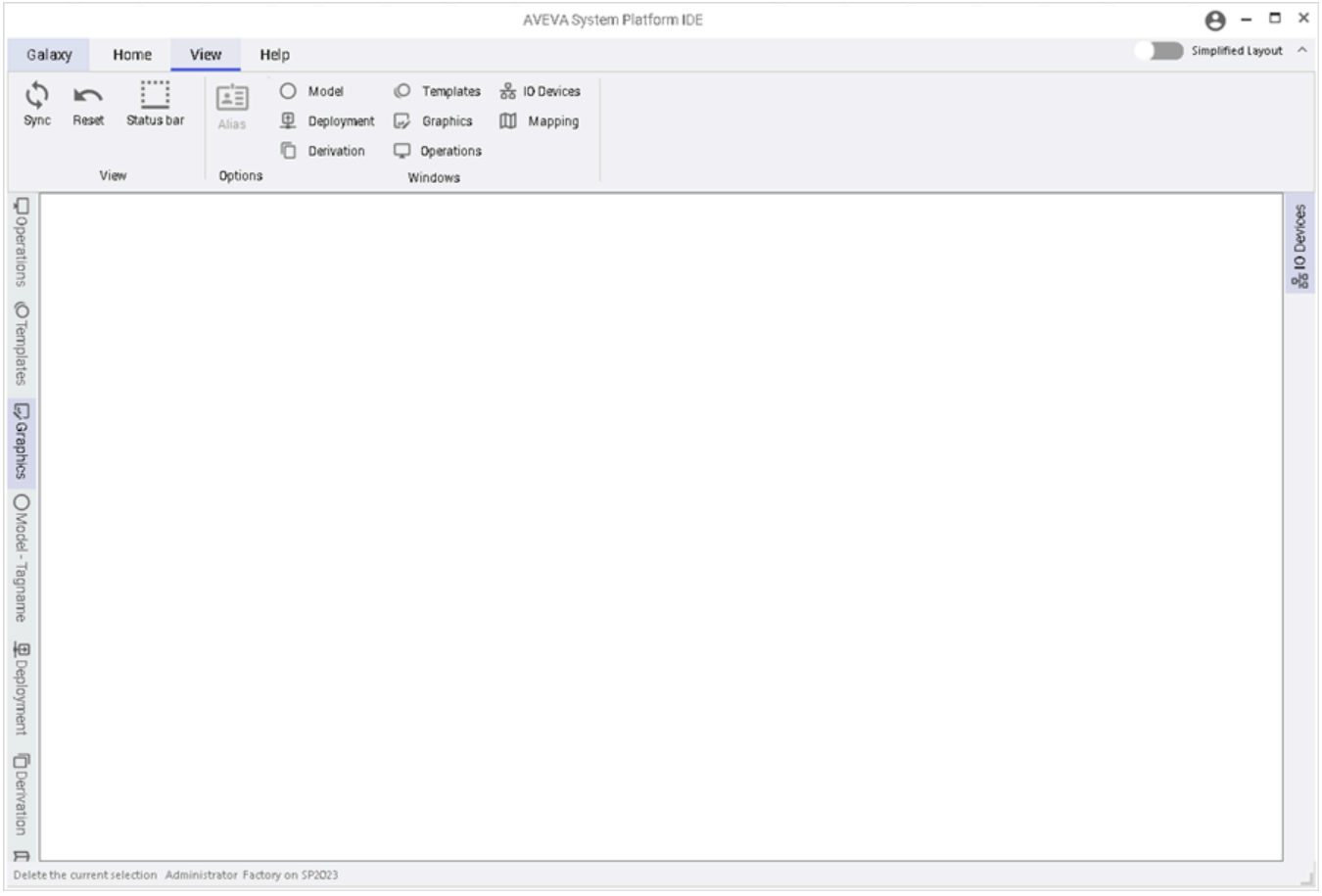
© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Float views

You can also float a view. With your cursor over the view you want to float, click the arrow. On the menu that appears, click **Floating**.

Hide views

	To hide a view, click the **Pin** icon. The views "hide" as tabs on the side of the window. The Operations and Mapping views hide at the bottom of the window, and the IO Devices view hides at the right side. The Visualization and Template folders, and Model, Derivation, and Deployment views are at the left side of the window.



When you move the mouse over the tab, the view expands into the workspace.

Reset the workspace

You can easily reset the IDE workspace and restore views back to their default locations.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

To return the views to the default docking

- On the **View** ribbon, select **Reset**.

Synchronize views

You can specify that a selected object stay selected as you move through the views. In any of the views, select the object you want to synchronize. On the **View** ribbon, select **Sync**. Now as you move from one view to another, that object stays selected.

You can also configure an option in your user information to automatically keep the same object selected as you switch between the **Model**, **Deployment**, and **Derivation** views. See Configure user preferences for details.

Configure user preferences

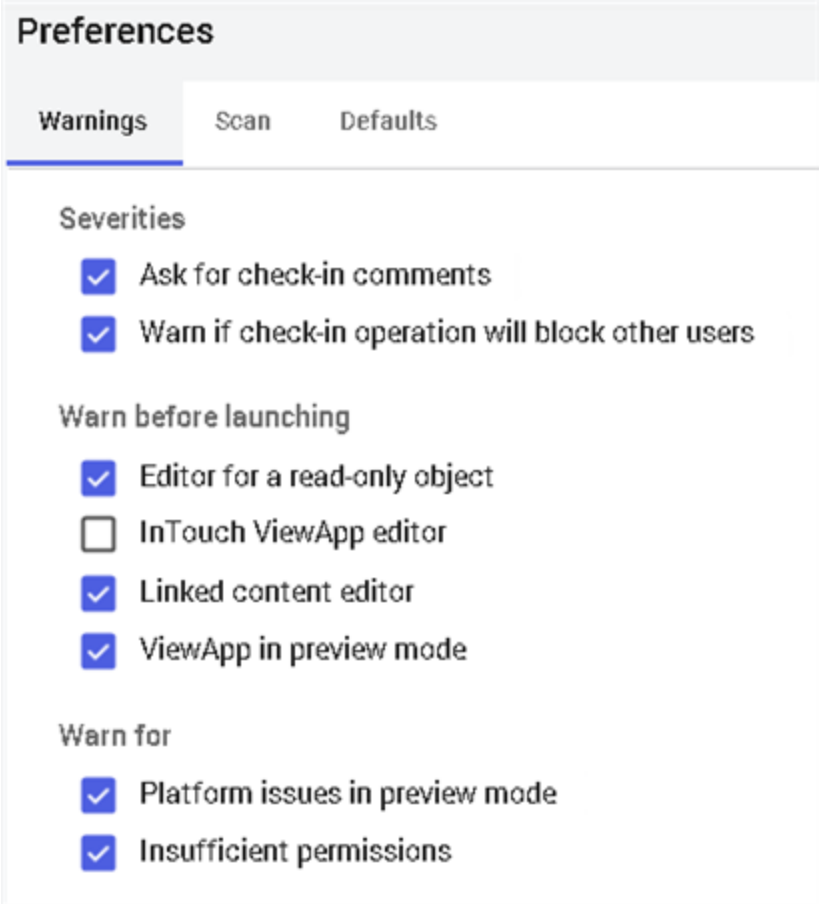
You can configure options for a Galaxy, including prompts for check-in comments and user defaults. These user options apply to the currently open Galaxy and do not change options for other Galaxies.

If you specify a security group, that security group must already exist. For more information about security and security groups, see Work with security.

To configure user information

1. From the ribbon, select **Galaxy**, then **User**, then **Preferences**. The **Preferences** pane appears. The pane has three tab pages.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.



2. On the **Warnings** page, select the check box for each option you want:
   - **Ask for check-in comments** to be prompted to type comments when checking in templates and objects.
   - **Warn if check-in operation will block other users** if you want to be warned if this occurs.
   - **Editor for a read-only object** to see a prompt that informs you if you are opening an instance or template as read-only. The prompt warns you if you open an instance or template while someone else is working on it. If someone else is working in the instance or template, you cannot make changes.
   - **InTouchViewApp editor** to see a prompt each time you attempt to edit an InTouchView application instance. You can edit the associated template or cancel the operation. If you do not select this check box, the request to edit the InTouchViewApp instance is automatically redirected to the associated template.
   - **Linked content editor** to see a prompt each time you attempt to edit a Visualization folder item, such as a graphic or layout, that is linked to an object. You can edit the linked content or cancel the operation. If you do not select this check box, the linked content opens in its editor (graphics in the Graphic Editor, layouts in the Layout Editor, etc.) without a warning, and you will be able to make changes to the content. If you then save the content, the item in the Visualization folder is updated with your edits, and the changes propagate to all objects that link to the content.
   - **ViewApp in preview mode** to restore the first warning message that appears after requesting a preview of a ViewApp. You need to select this prompt to display the message again after you select the check box on the warning message to hide the message.
   - **Platform issues in preview mode** to restore the warning message that appears after requesting a preview without the WinPlatform deployed. You need to select this prompt to display the message again

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

after you select the check box on the warning message to hide the message.

- Select the **Warn for insufficient permissions** to see a prompt that tells you if you have permission to create or modify InTouchView applications. These permissions authorize or prevent you from creating and modifying InTouchView Applications.

3. On the **Scan** page:
   - Select the **Initial scan state for deployed objects**. You can select **On Scan** or **Off Scan**. You can change this setting on an individual basis in the **Deploy** dialog box when you deploy.
   - Select the **Scan state defaults** when undeploying or redeploying instances. **Force Off Scan** will attempt to take the target object offscan when an already deployed object is redeployed. **Don't Force Off Scan** does not force the target to go off scan when you deploy.

     Note: Redeployment of objects that are currently deployed on-scan will be canceled unless this option is selected.

   - Enable or disable **Auto context selection**. This setting is enabled by default:
     - When **Auto context select** is enabled (checked), an object selected in the **Model**, **Derivation**, or **Deployment** view remains selected when you switch between views.
     - When **Auto context select** is disabled (unchecked), the **Model**, **Derivation**, and **Deployment** views do not keep the same object selected as you switch between views, so a different object can be retained in each view.
   - In the **Field Attributes** section, select the **Show Field Attributes** check box to display the **Field Attributes** tab in the User Defined Object editor. By default, the **Field Attributes** tab is displayed only if field attributes are already defined for the object. You can override the default by selecting this option. For more information about working with legacy field attributes, see Field attributes, UDAs, and extensions pages in the Application Server help file
4. On the **Defaults** page, type the object names of the Framework objects you want to be the defaults with respect to assignment relationships.
   - **Platform** name.
   - **Application engine** name.
   - **Area** name.
   - **View engine** name.
   - Your **Security group** name, if any.

   Leave a field blank if you do not want a default for that type of object.

   Your changes are saved automatically.
5. When you are done, click **OK**.

Create a Galaxy

Creating a new Galaxy requires you to specify a Galaxy Repository (GR) node name and the name of the Galaxy. The Galaxy database is created and is ready for you to connect to and use.

Guidelines for creating a new galaxy

- Activate a valid development license before opening the IDE.

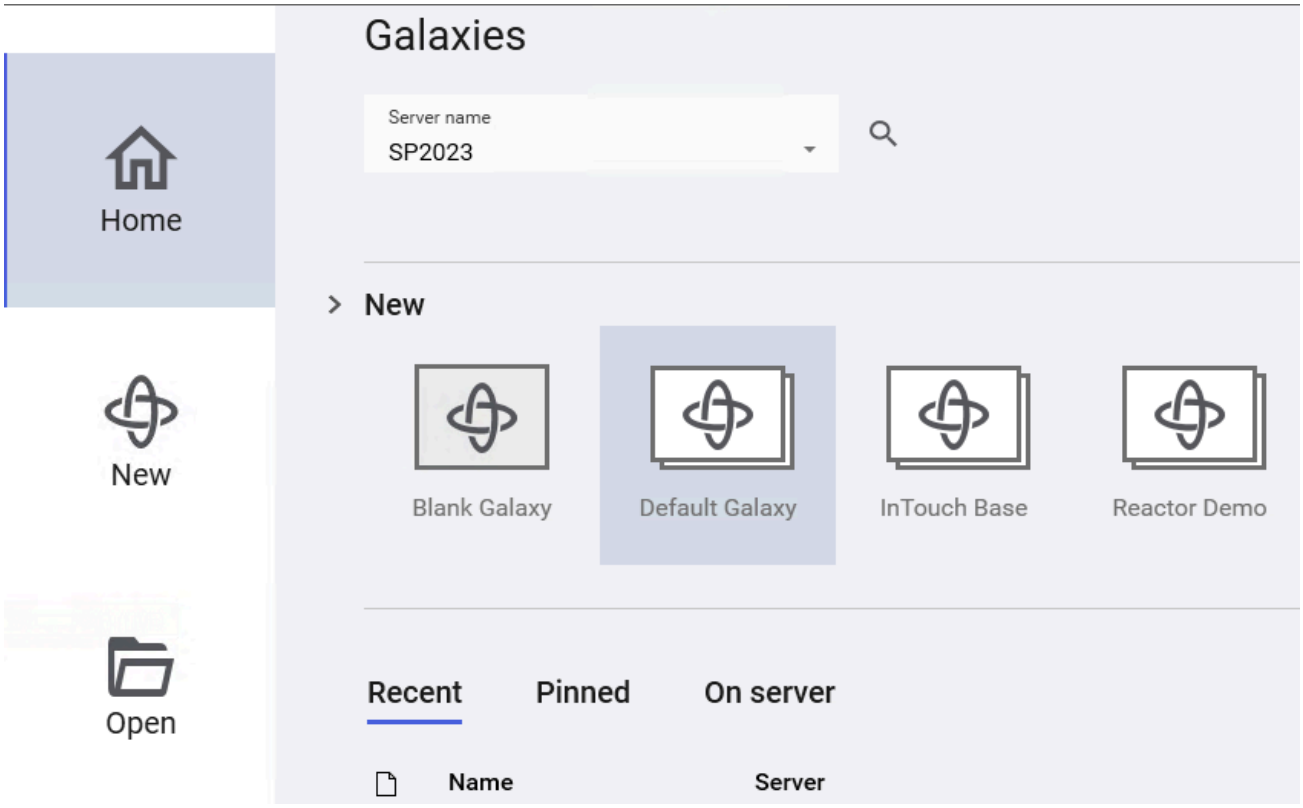
© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

- You can only create a new Galaxy on a computer with the Bootstrap and the Galaxy Repository software installed.
- To be able to create a new Galaxy, TCP/IP must be enabled on the computer hosting the SQL Server database.

  The TCP/IP protocol setting can be verified with **SQL Server Configuration Manager**. See the *System Platform Installation Guide* for instructions on enabling the TCP/IP protocol for SQL Server.
- New Galaxies are created without security. To learn more about setting security for your Galaxy, see Work with security.
- A computer name, as well as the name of the domain on which the computer exists, should include only English language characters, alphabetic and numeric. Use of characters from other language character sets that are not also included in the English language character set can result in a failure to create the new Galaxy.

To create a new galaxy

1. From the **Start** menu, go to the **AVEVA System Platform** folder, and then select **System Platform IDE**. The **Galaxies** dialog box appears.



2. If the **Server name** shown is not the Galaxy Repository server where you want to create the galaxy, either type the correct server name or click the down arrow and select it from the list.
3. In the **New** area, select the template to use for the new galaxy:
   - **Blank Galaxy**: Creates a baseline Galaxy that contains only base template objects. This is intended primarily as a recovery mechanism for recreating a damaged galaxy. After creating the baseline galaxy, you can import existing objects into it from aaPKG files.
   - **Default Galaxy**: This is the *recommended* starting point for a new Galaxy that will use AVEVA OMI

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

runtime applications (ViewApps). It creates default screen profiles, templates, a basic equipment model, and an example AVEVA OMI ViewApp that provides an introduction to key features.

- **InTouch Base**: Creates an InTouch HMI Galaxy. This is mainly for use if you are already using InTouch HMI applications, and can be used to convert legacy InTouch applications to managed applications.
- **Reactor Demo**: This is the same as the **InTouch Base** option, but includes a demonstration runtime application, "$ReactDemo," and all the configuration objects and graphic files used to create the demo. $ReactDemo is derived from the InTouchViewApp object.

If you have other galaxy templates available, click **More Templates** and select the one you want.

The **New Galaxy** dialog box opens.

4. Type the name for the new Galaxy. The name can be up to 32 alphanumeric characters, including _ (underscore), $, and #. The first character must be a letter. It cannot contain a blank space

   Note: You cannot use the following reserved names as Galaxy names: Me, MyContainer, MyArea, MyHost, MyPlatform, MyEngine and System. You cannot use a name that conflicts with an existing object in the template.

5. To open the galaxy after it is created, make sure **Connect to this Galaxy** is selected.
6. Click **Create**. The **Create Galaxy** dialog box opens and shows the progress of the Galaxy database being created.

   Note: If you use a backup file that was created in a previous version of Application Server as the galaxy template, and if Enhanced Security mode is in effect, you will be prompted to enter SQL SysAdmin credentials before proceeding.

7. After the Galaxy is created, click **Close**. If **Connect to this Galaxy** was checked, the Galaxy opens in the IDE. If it was not, you can open the Galaxy and begin creating your application. You must be a member of the aaConfigTools OS group to connect to a Galaxy, or you must be running the IDE as administrator. For more information about opening a Galaxy, see Connect to a Galaxy.

Connect to a Galaxy

You must connect to a Galaxy before you can perform work in it. To connect to a Galaxy, you must be a member of the **aaConfigTools** OS group *on the GR node*. To add a user to the **aaConfigTools** group, use the Windows Control Panel. If you are not a member of **aaConfigTools**, you can instead start the IDE as administrator and then connect to the Galaxy.

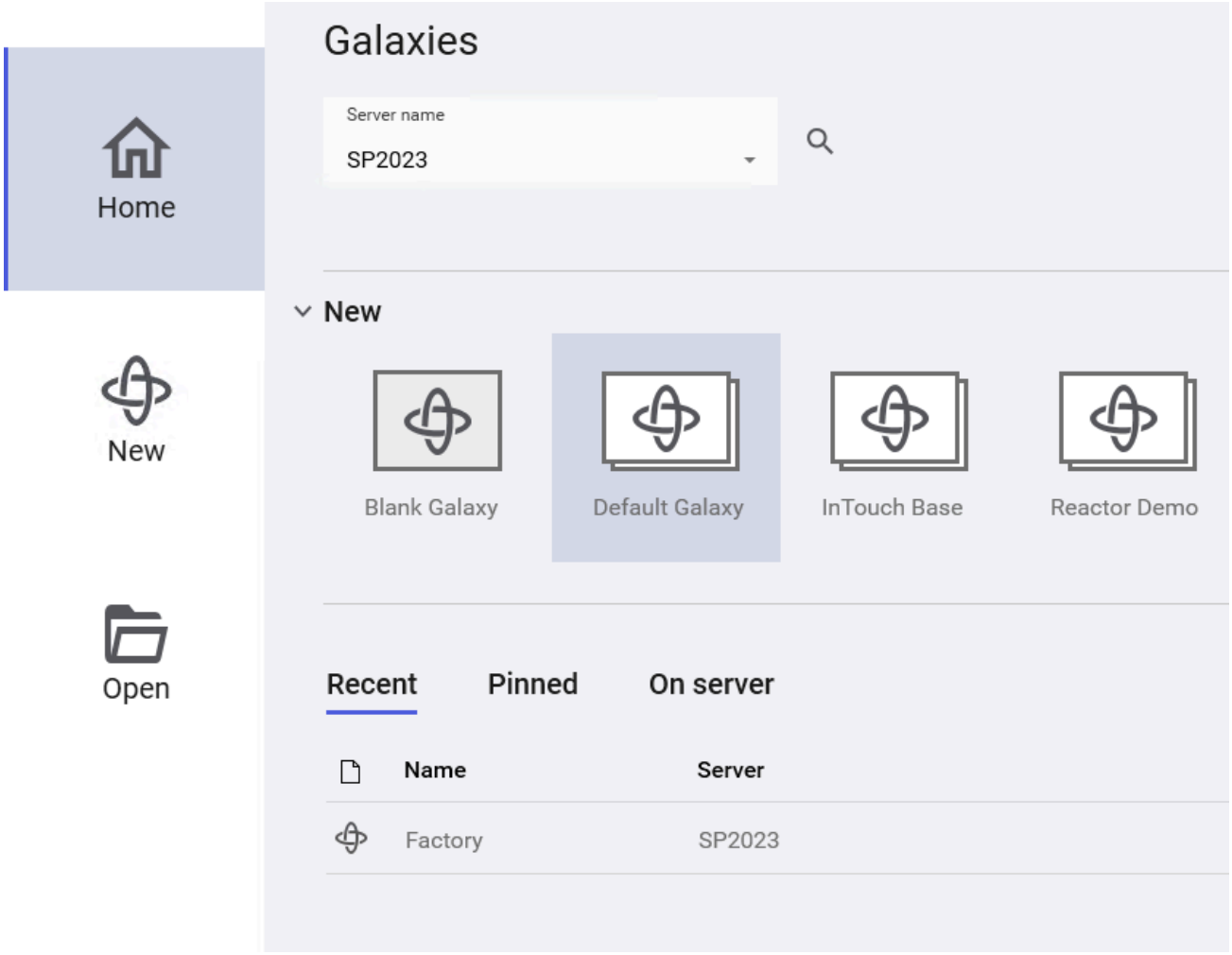
Important: You will not be able to connect to the Galaxy if you are not a member of the **aaConfigTools** group on the GR node. In a domain-based network, the **aaConfigTools** OS group must exist on the GR node.

If security is enabled for an existing Galaxy, you cannot open it without logging in, even if you are a system administrator or member of the **aaConfigTools** group. If you do not have log on rights to a Galaxy, you cannot log in to that Galaxy. For more information about Galaxy security, see Configure security.

To connect to a galaxy

1. From the **Start** menu, go to the **AVEVA System Platform** folder, then select **System Platform IDE**. The **Galaxies** page appears.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.



2. If the galaxy you want to connect to is shown, select it, then select the folder icon that appears to its right.
3. If the galaxy is not shown, but is hosted on the computer you are on, or on a network server that you have defined to System Platform (see below):
   a. Select **On server**.
   b. If necessary, in **Server name**, click the down arrow and select the server that hosts the galaxy.
   c. Select the galaxy, then select the folder icon that appears to its right.
4. If the galaxy is on a server you have not yet defined:
   a. Click **Open**. On the **Open** page:
   b. Select **Add a server**.
   c. In **Add a network server**, type the server name. Select **Add**.
   d. Under **Network Servers**, select the new server name.
   e. Select the galaxy from the list that appears on the right side of the page, then select the folder icon that appears to its right.

   If security is not defined for the galaxy, the IDE opens and you can begin working.

   If the galaxy does have security defined, you must log in.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

To log in to a galaxy

1. On the **Sign in to <galaxy>** dialog box, type your user name and select **Next**. If you need to include your domain,enter your name in the format *<Domain>**<User>*.
2. Type your password and select **Sign in**.

   Once the Galaxy opens in the IDE you can start working with your Galaxy. The IDE provides different ways to view the structure of the Galaxy and how it will be deployed to runtime. See IDE application views for more information.

Change users

You can change users in a Galaxy at any time. Any security restrictions associated with a user change when the user logs on or logs off from the Galaxy. For more information about users and security, see Work with security.

If the Galaxy has not been configured to enable security, you see a message. All users in an open-security environment are treated as the DefaultUser by the Galaxy. In an open-security environment, all users have full access to everything.

To change users

1. From the ribbon, select **Galaxy,** then select **User**.
2. Select **Change User**.
3. The **Sign in to <Galaxy>** dialog box appears.
4. Type your **User name** and select **Next**. If you need to include your domain,enter your name in the format *<Domain>**<User>*.
5. Type your **Password** and select **Sign in.**

View the Galaxy

Once you connect to a Galaxy, the IDE opens. When the IDE opens, it shows two windows that contain tools that you use to build and modify a galaxy, and three application views that let you see the structure of your Galaxy. The tool windows are:

- **Template folder:** this contains different object templates that you use to build a digital twin of your physical plant. The digital twin is a set of objects that interfaces with PLCs, RTUs, or other controllers. Expand the top level folders to see the different templates in the folders. For more information, see Template folder. For a complete discussion of template and instance objects, see About templates and instances.
- **Visualization folder:** this contains different graphic models that you use to construct the runtime graphical representation of your physical plant, including gauges, equipment, and valves. The graphic models in the folder include Industrial Graphics and Situational Awareness Library graphics. For more information, see Visualization folder.

When the Galaxy opens, the application views are shown by default in the lower left pane. See IDE application views for more information.

- **Model view:** this shows the object relationship to the automation scheme layout. The objects are organized

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

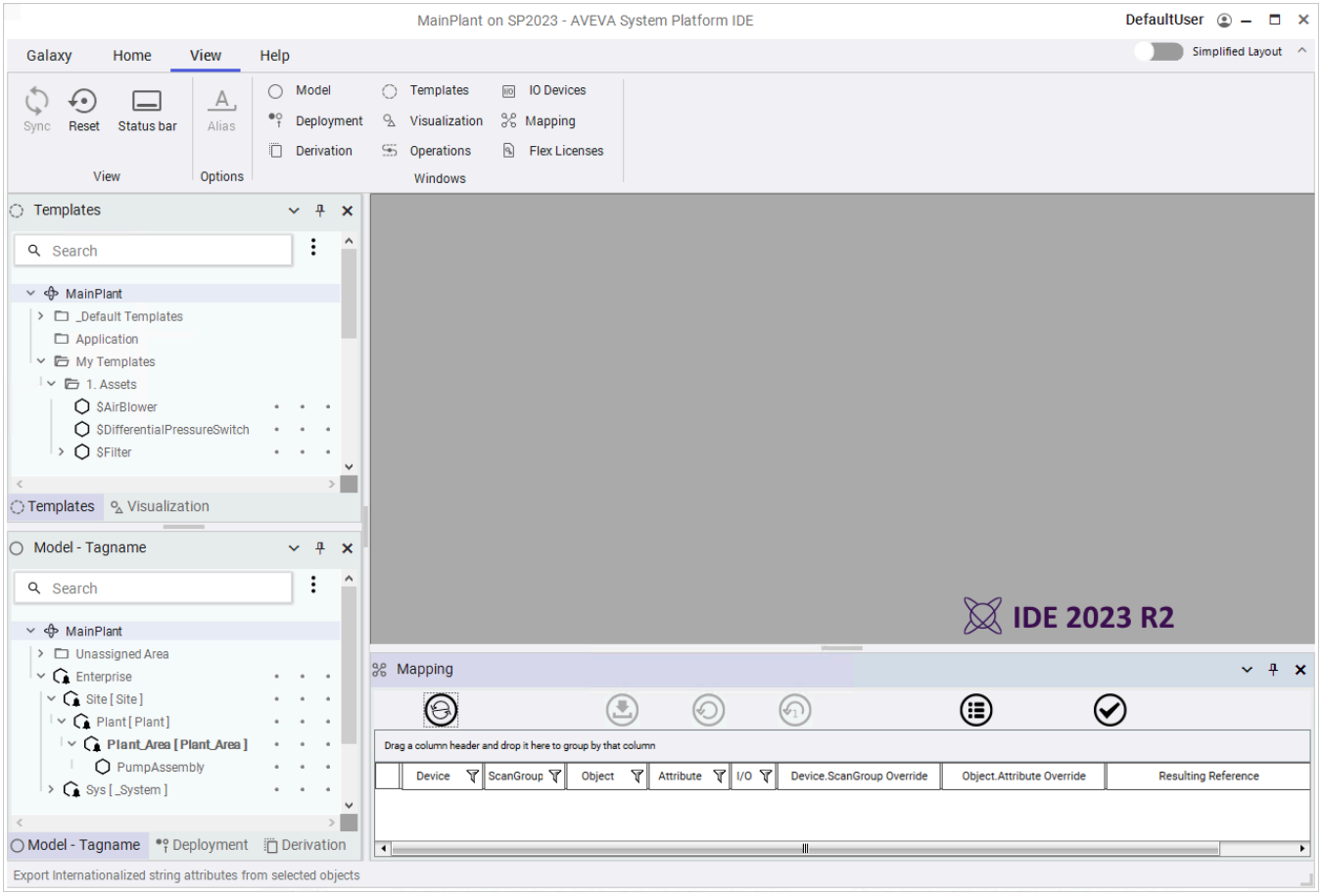
into Areas that typically represent the physical plant layout.

- **Deployment view:** this shows the object relationship to the computers that comprise the deployed system that the objects run on.
- **Derivation view:** this shows the derivation path from base template to the instances. This view allows a user to see all object instances that were based on a given template. All templates and instances appear in this view.

You can open additional application views:

- **IO Devices view:** this shows the object relationship to scan groups and DI objects. This view lets you assign objects that were configured for I/O automatic assignment in the Object Editor to scan groups, and assign scan groups to Device Integration (DI) objects. The IDE then builds the I/O references for each object.
- **Mapping view:** this shows the relationships of objects and attributes to scan groups and DI objects. This view lets you see, validate, and edit the I/O references of object attributes that have been automatically configured.
- **Operations view:** this shows the results of validating the configuration of objects.

A status bar at the bottom of the IDE window shows messages, user name, Galaxy name and node, and license information. You can turn off the Status bar by clicking **Status bar** on the **View** ribbon.



© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Template folder

The Template folder lists template folders, which contain object templates. The Template folder shows a tree view of template categories in the Galaxy. Expand the folders to see their contents.

A new Galaxy is automatically populated with the content defined by the cab file (template) that was used to create it. At a minimum, the newly-created Galaxy will include base templates.

- The Default Galaxy includes first level derived templates in addition to base templates, to simplify the process of building your Galaxy, and a sample AVEVA OMI ViewApp.
- The Blank Galaxy includes only base templates. See About base templates for additional information.
- The InTouch Base and Reactor Demo cab files include only the base templates needed to support an InTouch HMI application.

Visualization folder

The Visualization folder lists graphic-related content. The Visualization Folder shows a tree view of the graphic folders. As with the Template Folder, a new Galaxy is populated with the content defined by the cab file used to create it. Expand the folders to see their contents.

- The Default Galaxy includes several graphic libraries, including the Industrial Graphics Library, the Situational Awareness Library, and an icon library. The Default Galaxy also includes various Screen Profiles, Layouts, Apps and Controls.
- The Blank galaxy includes only a basic set of Apps and Controls.

  The InTouch Base and Reactor Demo include the Industrial Graphics and Situational Awareness Libraries. The Reactor Demo also includes additional graphics used for the InTouch HMI reactor demo app.

Work with objects

The System Platform Integrated Development Environment is the tool that you will use to work with Application Server objects. Both templates and instances are collectively referred to as objects. For more information about what templates and instances are, see About templates and instances.

View object properties

You can view the properties of an object by right-clicking and clicking **Properties**. Object properties vary, depending on the type of selected object and whether it is a base template, a derived template, or an instance.

The **Properties** window contains the tabs that allow you to examine different object properties. Only tabs applicable to the object are displayed.

Tab Name	Displayed for Object Type:
General	All objects including GalaxyObject
Attributes	All objects except GalaxyObject

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Tab Name	Displayed for Object Type:
References	Instances only
Cross References	Instances only
Change Log	All objects including GalaxyObject
Operational Limits	All objects including GalaxyObject
Errors/Warnings	All objects except GalaxyObject

The **General** tab contains basic information for the object, such as the codebase (major version), what template it is derived from, and any current errors and warnings.

The **Attributes** tab lists the object's attributes and values. In addition to attributes that have been added to the object, attributes for Errors and InAlarm are also listed, as are attributes for CodeBase (major version of the object), MinorVersion ("dot" version), and ConfigVersion. The ConfigVersion will increment each time the object has been checked in after an edit.

The **References** tab lists I/O references for the object.

The **Cross References** tab lists any attributes that are cross referenced for the object.

The **Change Log** tab is a historical list of changes, including deploy/undeploy operations, that have been made to the object.

The **Operational Limits** tab lists prohibited actions and the reason for the prohibition. For example, Check In is not allowed for objects that are already checked in.

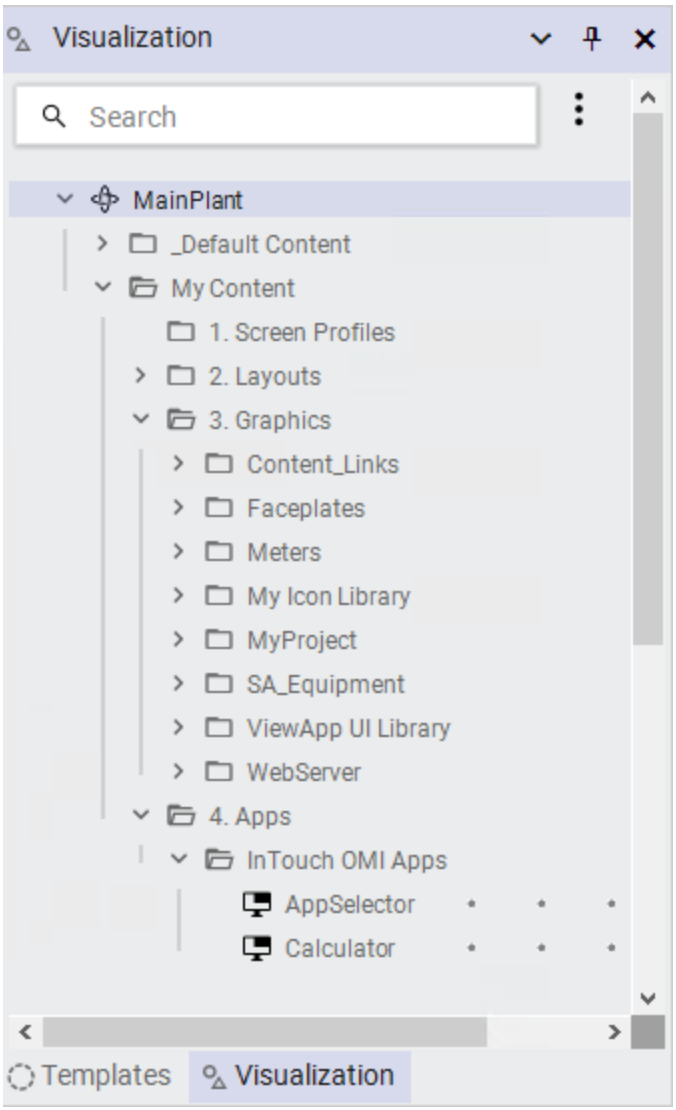
The **Errors/Warnings** tab lists any errors or warnings for the object.

For more information about specifying the properties of objects, see Work with objects.

Manage folders

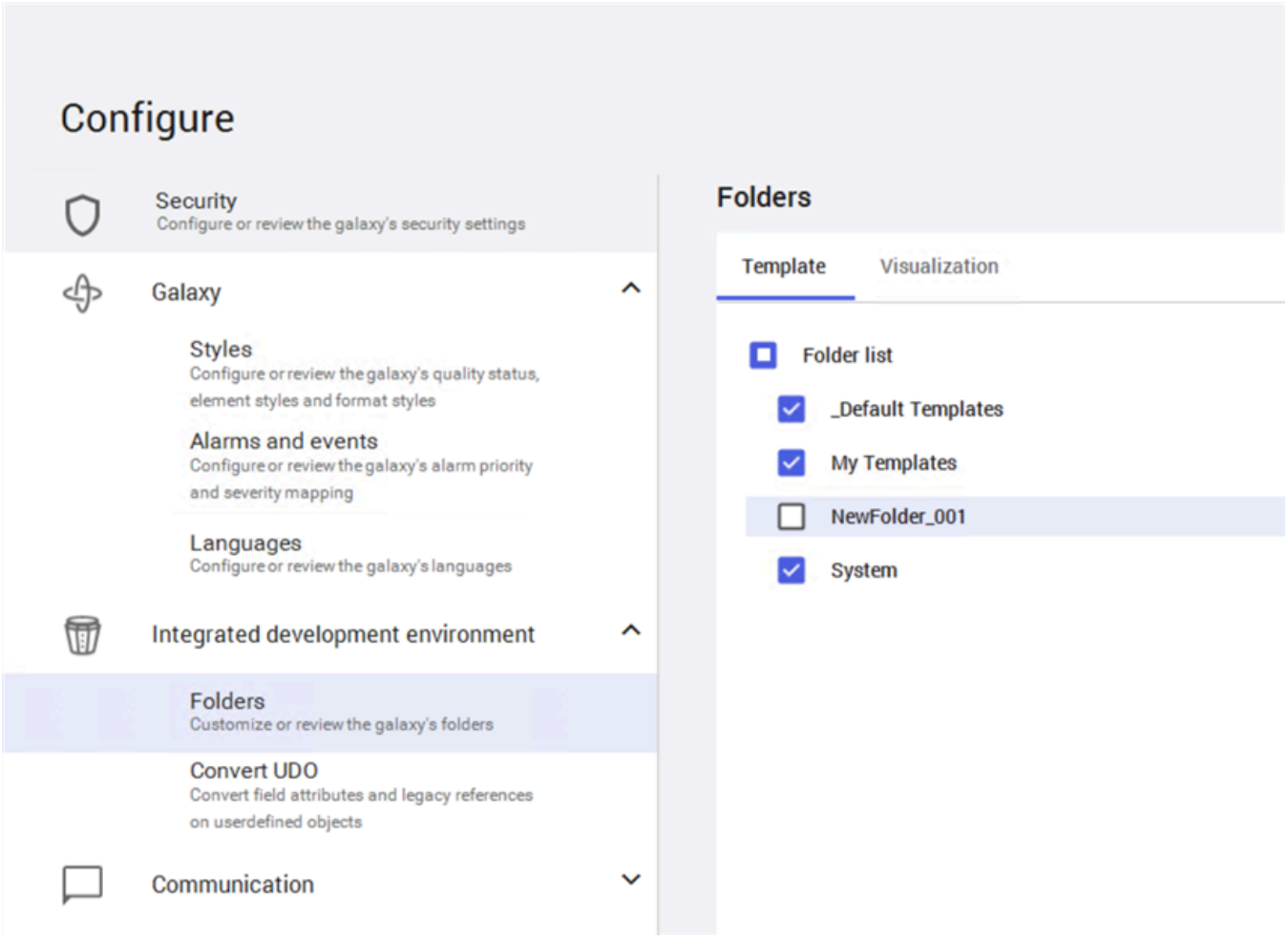
Folders are used to organize objects used to build the Galaxy, and broadly categorized into the Template and Visualization folders. Each folder is indicated by the folder icon. The Template folder contains application objects templates, used for creating a digital twin of the physical environment. The Visualization folder contains objects used for creating runtime displays, such as graphics, apps, controls, widgets, screen profiles, and layouts. The Template and Visualization folders each contain subfolders that further categorize the objects that they contain. These folders are initially created during the Galaxy creation process.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.



You can move items to different folders within the Template and Visualization folders, but you cannot move a Visualization object to the Template folder and visa versa.You can create additional folders and subfolders to store the templates and graphics used in your application. All folders contained within a Galaxy are displayed by default, but you can customize which top-level folders are shown to make the workspace less cluttered. To set the visibility of a folder, go to the **Galaxy** backstage menu and select **Configure**, then **Integrated development environment**. Then, select **Folders** and deselect any folders listed under the Template and Visualization folders that you want to hide. Click **Save** to preserve any changes. Subfolders do not appear in this configuration option and cannot be selected/deselected.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.



Create folders

When you create your own folder, it must have a unique name. Folder names are not case sensitive, so `Valves` is the same name as `valves`. A folder name can be up to a maximum of 64 alphanumeric and special characters, including spaces, except $.

To create a new top-level folder in the Template or Visualization folder

1. On the **Home** ribbon,in the **Create** area, select **Folder**, then **Template folder** or **Graphic folder**.
2. Type a name for the new folder.

   A new folder appears and is selected. You can now drag templates into the new Template folder, or you can drag graphics into the Graphics folder.

Create child folders

Folderscan be created within existing folders. Nested folders help in further organizing templates and graphics. You can create a maximum of ten levels of folders.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

To create a child folder

1. Select the parent folder.
2. On the **Home** ribbon,in the **Create** area, select **Folder**, then **Template Folder** or **Visualization Folder**.
3. Type a name for the new folder.

   A new folder appears beneath the parent folder and is selected.

   You can now drag templates into the new folder.

Delete folders

You can delete folders you no longer want or need. Before you start, make sure you move or delete all content from the folder. The folder you want to delete must be empty, or it cannot deleted.

To delete a folder

1. Select the folder to delete.
2. On the **Home** ribbon, in the **Edit** area, select **Delete**.
3. Click **Yes** to delete the folder.

About using CONNECT

AVEVA Connect is the common cloud repository allowing you to to manage common contents, such as, Industrial Graphics, Controls, and Widgets in the Cloud.

Graphics can be downloaded and uploaded on demand. Graphics are stored in ‘Stores’ in AVEVA Connect; there are three types of stores – global, tenant and user specific. Within each store users can configure multiple drives. You must have an AVEVA Connect user account to manage graphics in the cloud. Each drive can be configured with different access levels for different users.

Users will have read-only or read-write access depending on the access granted by the administrator.

- Users with Read-Only access can view graphics in the cloud. They can download the graphics to the cloud, but these graphics would be available as read-only. They cannot modify or upload the graphics.
- Users with Read-Write access can view, download, modify, and upload graphics to the cloud.

Multiple users can access the graphics in the same drive, but only one user can edit or save a graphic at one time.

Log into CONNECT

You must have a CONNECT account and authorization from your administration to access CONNECT.

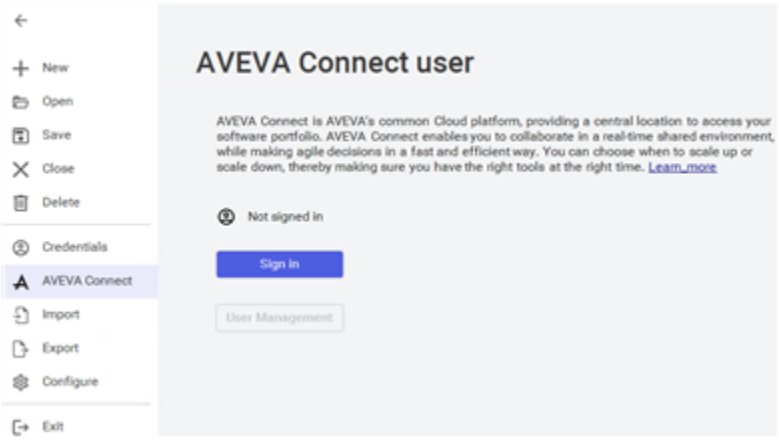
1. If you are using InTouch, launch WindowMaker, then, from the **File** menu, select **CONNECT**.

   If you are using the System Platform IDE, select the **Galaxy** menu, then select **CONNECT**.

   Note: The **CONNECT** option will not be available if you have selected **connected experience** mode under the **Licensing Mode** tab of the Configurator.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

The CONNECT user configuration screen appears.



2. Select **Sign In**.
3. Enter your CONNECT email address.
4. Enter your password.

   If the credentials are verified, you are signed in.

   Note: You must have the Content Contributor role in order to access the CONNECT Industrial Graphics drive. This is required for both Operations Control connected experience and non-connected experience use.
5. Select the **AVEVA Cloud Integration Studio solution.** For example: Pym Technologies.
6. The AVEVA Drive appears as a separate repository within the Visualization folder.

   Note: Once you have selected an Integration Studio solution, it is cached in the browser memory. To switch the Integration Studio solution, you must clear the browser cache.

Navigate between shared drives

Once you are logged into a specific Store you can move between different drives and upload and download graphics.

1. On the **View** menu, in the **Cloud drive** group, select **Shared Drive**.
2. Select a drive from the available options.

   The Visualization folder will refresh to display the selected drive.

   Note: You cannot move a graphic which is locked by another user.

Upload/download graphics to the cloud

The AVEVA Drive functions similar to your local Visualization folder. You can upload or create folders on the AVEVA Drive to organize your graphics. With Read-Write access, you can download graphics to the local Visualization folder and use it for your applications. When you download or upload graphics, the contents will be overwritten according to the selected overwrite option

- If the contents already exist in the destination drive, the contents will remain in the same toolset location. Otherwise, the contents will be added in the same toolset path as the source

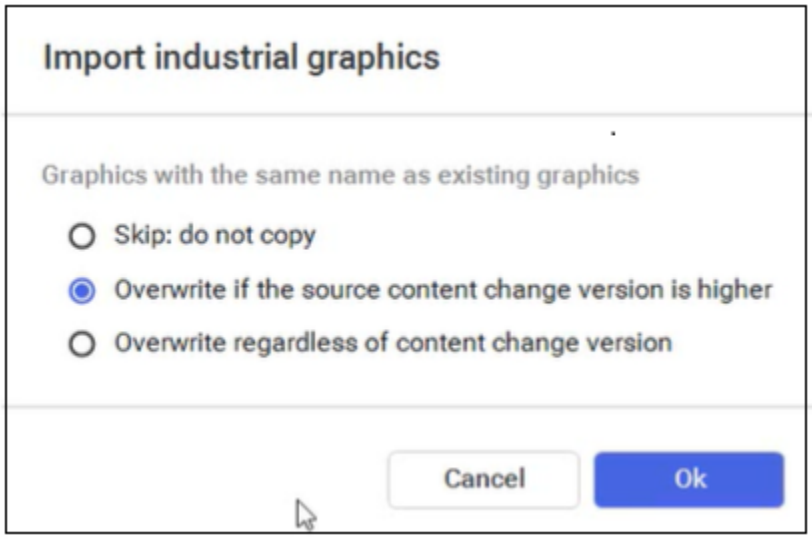
© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

For the selected embedded contents

- If the embedded contents already exist in the destination drive, the contents will remain in the same toolset location in destination.
- If the embedded content resides in the same toolset location as its parent content or within the same folder hierarchy, then the embedded contents will be added in the same toolset path as the parent. Otherwise, the embedded contents will be added in the same toolset path as the source.

Upload graphics to CONNECT

1. Drag the graphic(s) from the local Visualization folder and drop it to the AVEVA Drive folder.
2. If the graphic already exists in the drive then a dialog appears, requesting to overwrite or skip the upload.



3. Select the appropriate option, and select **OK**.
4. The **Upload Content** dialog appears and displays the progress of the operation. Select **View Details** to view the progress.
5. Select **Close**.

You can also upload folder(s) that contains multiple graphics.

Download graphics to the local visualization folder

Users with Read-Write access can download graphics from the AVEVA Drive to their local Visualization folder.

1. Drag the graphic(s) or folder from the AVEVA Drive and drop it to the local Visualization folder.

   The **Download Content** dialog box requests confirmation of the download.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.



- Select **Yes** to continue with the download.
- Select **No** to cancel the operation.
- Select **View Details** to view the list of controls and namespaces used. Any errors during downloading are also displayed.

2. If the graphic already exists, the Import industrial graphic dialog box appears.
3. Select the appropriate option.
4. The graphic(s) are downloaded to the local Visualization folder. Embedded graphics are also copied.

Upload/download support for various cloud content

The upload/download support of various cloud content is listed in the table below.

Cloud Content	Upload Support	Download Support
Embedded Client Control and HTML5 Widget	Upload will package the embedded graphics contents including Client Control and HTML5 Widgets as aaPKG format and copy/overwrite to the Cloud.	Download will extract the embedded graphics contents including Client Control and HTML5 Widgets and import/overwrite the control into the local application
Embedded application object graphic	Not Supported	Not Supported
Custom script library	Not Supported	Not Supported
Application Style Library	Not Supported	Not Supported

Manage graphics in CONNECT

You can also create, rename, update, duplicate and delete graphics directly on AVEVA Drive, just like managing local graphics.

- Right-click the AVEVA Drive or any of the folders to create new graphic or toolset.
- Double-click to edit graphics using the graphic editor.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

- Right-click and select the corresponding options to rename, duplicate or delete the graphic, and update thumbnails.

  The context menu options that are not supported for the Cloud appear greyed out. For example, Set Web Client Home Symbol, Set Web Client Root Folder, Export Symbol, and Export Localization.

  Note: Updating thumbnails for embedded graphics directly in the AVEVA Drive causes a lot of cloud traffic. Hence, we recommend that you update the thumbnails locally.

Manage graphics with multiple users

Multiple users can use the graphics available on AVEVA Drive at a time. However, a graphic can be edited by only one user at a time. When a graphic in the Cloud repository is being edited by one user, the graphic is checked-out and locked, preventing other users from making any edits. All other users can only view the read-only unmodified graphic when it is checked-out. The latest changes reflect only when it is checked-in the modifying user.

- A lock icon against the graphic icon in the folder indicates that the graphic is checked out by the logged in user.
- An edit icon against the graphic icon in the folder indicates that the graphic is being edited by another user.

After the first user saves and checks in the graphic, the second user can check it out and make changes to it. Once the graphic is saved and checked-in, all users can view the modified graphic.

About cloud graphic versions

The AVEVA Drive graphics supports Industrial Graphics, Client Controls, and Widgets in the Cloud, published from different products with different versions of the graphic subsystems.

Key concepts of cloud graphic version support

- Industrial Graphics are stored locally and in the Cloud in a specific format that supports all the features and capabilities available in the Graphic subsystem.
- When new features and functionality are added to the Graphic subsystem, the graphics need to be saved and checked-in to be available for other users.
- Different versions of the Industrial Graphics subsystem are identified by a unique, internal version number, not visible to the user. At the time of their release, each product will consume the most current version of the Industrial Graphics subsystem.
- The version number of the Industrial Graphics subsystem is included as part of each stored Industrial Graphic, and is used by the Industrial Graphic subsystem to ensure only a product with a matching or higher version of the Industrial Graphic subsystem is allowed to edit, embed, or download that Industrial Graphic.
- All users connecting to the same Cloud repository will see the same set of graphics. The ability to edit, embed, or download a Cloud graphic will depend on the version of the Industrial Graphics subsystem in the product through which the user is connecting.
- The Industrial Graphics saved to the Cloud repository by a product with a newer version of the Industrial Graphic subsystem will appear disabled in the Visualization folder of the products with older version accessing the same Cloud. The user from the old product version will not be able to edit, embed, or

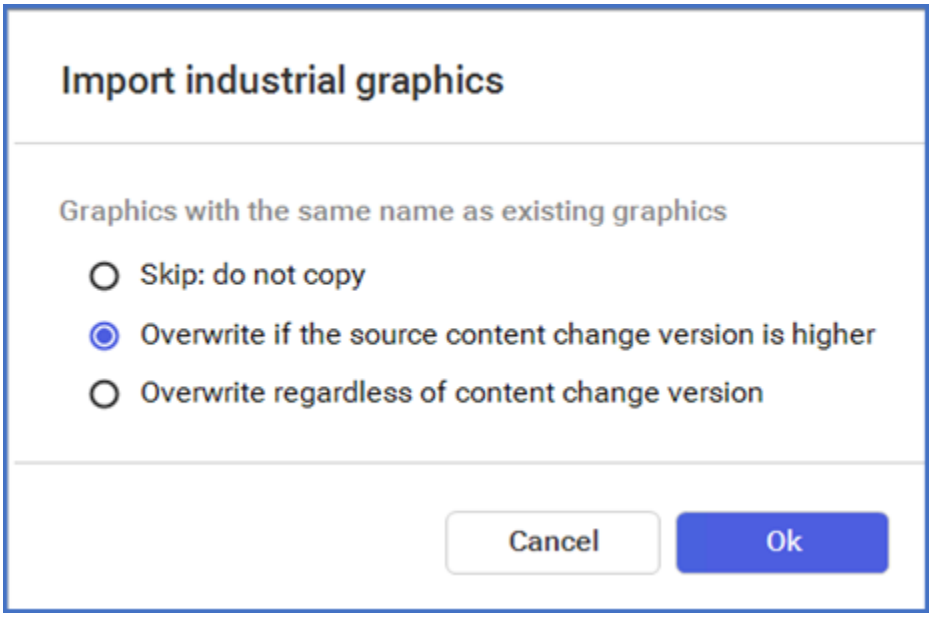
© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

download the graphic repository.

- Just like the local Visualization folder, graphic names must be unique within any single Cloud repository. This holds true even if the graphics are stored by products with different versions of the Industrial Graphics subsystem.

Overwrite graphic contents in the cloud

If the graphic content being uploaded or downloaded already exists, the system prompts to overwrite the graphic.



The overwriting depends on the graphic subsystem versions of the source and the destination, and the change log version of the graphic.

- Option 1 will skip the overwrite regardless of the graphic subsystem versions or the change log version of the source and the destination.
- If the graphic subsystem version is different between source and destination, both options 2 and 3 will overwrite the graphic, regardless of change log version.
- If the graphic subsystem version is same between source and destination:
  - Option 2 will overwrite only if source has higher change log version.
  - Option 3 will overwrite regardless of the change log version.

Overwrite custom client controls and widgets in the cloud

The guidelines of Cloud Graphic Version Management do not apply to the Custom Clients Controls and Widgets at the time of this release. Client Controls and Widgets are supported for both download or upload operation. However, overwrite is not supported. If a Custom Client Controls or Widgets is already available at the target location, irrespective of the client control internal version, the overwrite will be skipped. To overwrite the source repository client control, you must first delete the client control in the source repository. After that, upload/ download the client control will be successful.

Note: If a new Custom Client Controls or Widget is uploaded to the Cloud, all other users connected to the same

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Cloud repository must restart the WindowMaker for the latest Custom Client Controls or Widget to be available.

Cloud graphic versions - scenarios and examples

This section provides a non-exclusive list of high level scenarios and expectation for Cloud graphic version handling.

Let's consider two products P1_Old and P2_New, of different versions, which connect to the same tenant in AVEVA Cloud.

- P1_Old has the older version of Graphic Subsystem and P2_New has the newer version of Graphic Subsystem.
- To begin with, the cloud repository has graphics S1 and S2 uploaded by P1_Old.

Scenario / Steps	Expectation
Scenario 1: P2_New creates S3. P1_Old attempts to edit S3.	S3 is saved in newer graphic version and hence it cannot be edit, embed, or download in older version. P1_Old receives a message stating that S3 is saved in newer version and cannot be opened.
Scenario 2: P2_New opens S2. P2_New saves S2 in Cloud. Then, P1_Old attempts to open S2.	S2 created in older graphic version can be opened and saved in the newer graphic version. S2 is saved in newer graphic version and hence it cannot be edit, embed, or download in older version. P1_Old receives a message stating that S3 is saved in newer version and cannot be opened.
Scenario 3: P1_Old saves S1 and S2, where S2 is embedded in S1 P2_New saves and updates only S2. P1_Old attempts to edit S1.	S1 is opened because it is still the same version in P1_Old. The embedded S2 will fail to load.
Scenario 4: P2_New creates a new S4 in Cloud. P2_New embeds S1, S2, and S3 in S4.	S1 is in older graphic version but still can be loaded in S4 (newer version). S2 and S3 can be embedded because they are the same graphic version as S4.
Scenario 5: P1_Old creates a new S5 in Cloud. P1_Old embeds S1, S2, and S3 in S5.	S1 can be embedded because they are the same graphic version. S2 and S3 cannot be embedded due to newer version.
Scenario 6: P1_Old download S3.	S3 cannot be downloaded because it is in newer graphic version.
Scenario 7: Cloud repository has a graphic S3 which is in newer	By selecting Option 3, the graphic can be overwritten.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

graphic version. P1_Old attempts to upload a graphic with same name S3.	
Scenario 8: P2_New has a graphic S8. P1_Old creates locally S8, S9, and embed S8 in S9. P1_Old uploads S9 to Cloud.	S9 is uploaded to Cloud. S8 is prevented upload because a newer version exists in the Cloud. By selecting Option 3, the content can be overwritten.

Manage galaxy style libraries

You can import a Galaxy Style Library to load its Element Styles in a Galaxy. You can also modify Element Styles, and then export the Galaxy Style Library as custom user-defined Element Styles.

Optional Galaxy Style Library XML files are located in the ...\Program Files (x86)\ArchestrA\FrameWork\Bin\AdditionalElementStyles folder. The names of the XML files suggest the primary color schemes of the Element Styles within each optional Galaxy Style Library.

A Galaxy Style Library can be exported from the IDE and imported into other Galaxies.

Note: Only one Galaxy Style library can be loaded and used in a Galaxy at a time.

Import a galaxy style library

1. From the menu bar, select **Galaxy**, then **Import**, then **Styles**.
2. Browse for the library file. Galaxy Style Library files are XML files.
3. Select the file and click **Open**. The selected Galaxy Style Library is loaded in the IDE.

If you imported a Galaxy Style Library, it is now the active Galaxy Style Library for the entire Galaxy.

Note: If a Galaxy Style Library is imported into the IDE while a ViewApp is running, application graphics are refreshed with new Element Styles. The ViewApp refreshes automatically. A restart is not required.

Export a galaxy style library

You can create your own Galaxy Style Library by overriding the configuration of Element Styles and exporting the library. For information about setting Element Style overrides, see Change the visual properties of an element style.

If the galaxy has more than one style library, exporting exports all libraries, into separate XML files. You cannot export only one style library.

To export galaxy style libraries

1. From the menu bar, select **Galaxy**, then **Export**, then **Styles**. The **Select Folder** dialog box appears.
2. Browse to and select a folder.
3. Click **Select Folder**.

   The style libraries are exported to the selected folder, each into its own XML file.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

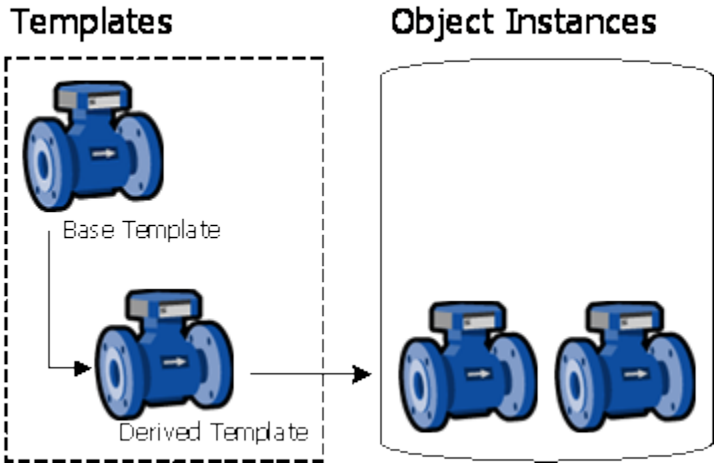
Reset a pre-defined or user-defined galaxy style library

If you changed the values of Quality and Status values, Pre-Defined or User-Defined Galaxy Element Styles, or Format Styles, you can restore the default values by clicking the **Reset to Default** button.

Create derived templates

All templates you create are derived templates. A derived template inherits attributes and behaviors from the parent template. You cannot change the attributes in a base template.

After you create a derived template, you can customize and modify its attributes. If you change locked attributes in the parent template, the changes propagate to the derived template.



After you create derived templates, you can customize them, derive other templates from them, and create instances of them. You can change and modify unlocked attributes in the instances, making adjustments to meet the needs of the specific object you are modeling.

For example, your plant processes can use several models of a pump made by a single vendor. Each model has unique characteristics that map to different attribute values of the DiscreteDevice base template.

To derive a template from another template

1. In the **Template folder** or **Derivation** view pane, select the template to use as the parent template
2. On the **Home** ribbon, in the **Create** area, select **Template**. A derived template is created in the same folder as its parent and placed in name edit mode. The default name is the same as the parent template followed by a numeric sequence.
3. Rename the derived template, if needed. Template names can be up to 32 alphanumeric or special characters, including the required $ as the first character. The second character cannot be $ and the name must include at least one letter. Template names cannot contain spaces.

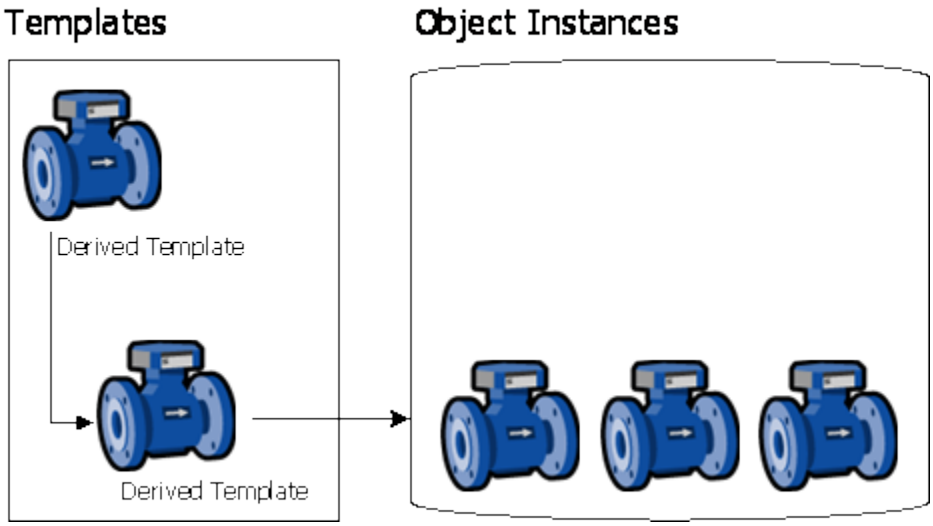
   Note: The following reserved names cannot be used as template names: Me, MyContainer, MyArea, MyHost, MyPlatform, MyEngine and System.
4. You are ready to customize your new template. For more information, see Edit objects.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Derive templates from another derived template

You can create derived templates from other derived templates. The child template inherits attributes from all parent templates. Any changed attributes in the immediate parent overrides attributes in grandparent levels.

If you change locked attributes in the parent template, the locked attributes propagate to the derived template.



A derived template is an exact copy of its parent with the exceptions of locking, security, and the unlocked attributes that have been edited. If you create a new derived template from an existing container template, the new derived template has the same contained templates.

A good practice is to create a hierarchy of derived templates until you reach logical endpoints. Then create instances from each unique derived template.

To create a derived template from a derived template

1. In the **Template folder** or **Derivation** view pane, select the template to use as the parent template
2. On the **Home** ribbon, in the **Create** area, select **Template**. A derived template is created in the same folder as its parent and placed in name edit mode. The default name is the same as the parent template followed by a numeric sequence.

   Template names can be up to 32 alphanumeric or special characters, including the required $ as the first character. The second character cannot be $ and the name must include at least one letter. Template names cannot contain spaces.

   Note: The following reserved names cannot be used as template names: Me, MyContainer, MyArea, MyHost, MyPlatform, MyEngine and System.
3. You can create another derived template by repeating the previous steps, or you can customize your new derived template. For more information about customizing your template, see Edit objects.

Create contained templates

Containment is the relationship in which one object includes another. Containment relationships organize objects in a hierarchy. You can build objects that represent complex devices consisting of smaller, simpler devices.

In scripts, these objects can be referred to by the name that derives from the containment relationship. This

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

name is called a hierarchical name.

An object can have three kinds of names if it is contained by another object. The three names include:

Name	Description
Tag Name	The unique name of the individual object. For example, `Valve1`.
Contained Name	The name of the object within the context of its container object. For example, the object whose Tag name is Valve1 may also be referred to as `Tank1.Outlet`, if `Tank1` contains it and it has the contained name "Outlet".
Hierarchical Name	Hierarchical names that are fully-qualified names of a contained object include the name of the objects that contain it. Because the object that contains it may also be contained, there are potentially multiple hierarchical names that refer to the same object. For example, if: "Reactor1" contains Tank1 (also known within Reactor1 by its contained name "SurgeTank"). "Tank1" contains Valve1 (also known within Tank1 by its contained name "Outlet"). Valve1 could be referred to as: "Valve1" "Tank1.Outlet" "Reactor1.SurgeTank.Outlet".

Note: Base templates cannot be contained by another template, either as the container or as the template being contained. You can only use containment with derived templates.

Higher level objects contain lower level objects. This allows you to more closely model complex plant equipment, like tank systems. You can nest templates to 10 levels.

Note: Objects can only contain objects like themselves. For example, ApplicationObjects can only be contained by other ApplicationObjects. Areas can only contain other Areas.

ApplicationObject containment

ApplicationObjects can be contained by other ApplicationObjects. This provides context for the contained object and a naming hierarchy that provides a powerful tool for referencing objects.

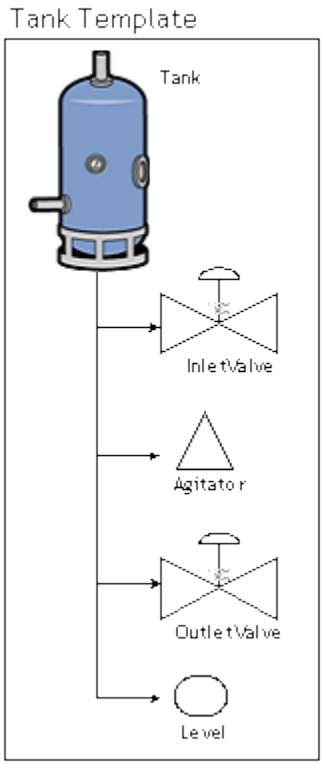
Note: Base templates cannot be contained by another template, either as the container or as the template being contained. You can only use containment with derived templates.

An example of a containment hierarchy is a tank that contains the following objects:

- Inlet Valve

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

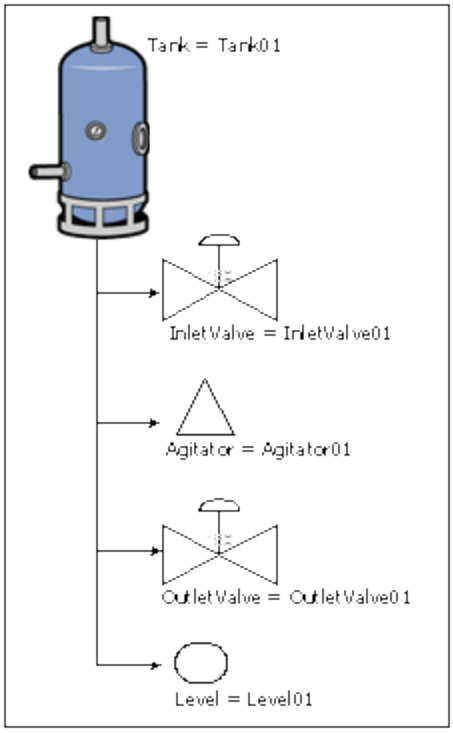
- Agitator
- Outlet Valve
- Level



To enable referencing and flexibility within scripting, these objects can be referenced in several different ways. Each object has a unique tag name, such as:

- Inlet Valve = InletValve01
- Agitator = Agitator01
- Outlet Valve = OutletValve01
- Level = Level01

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

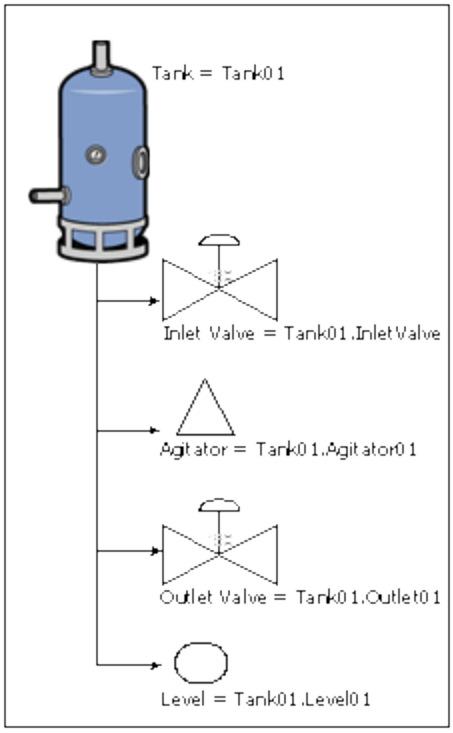


Within the context of each hierarchy, the contained names are unique, in that the names only refer to this tank system and the contained objects.

So if the tank is named `Tank01`, the contained names are:

- `Tank01.InletValve`
- `Tank01.Agitator01`
- `Tank01.Outlet01`
- `Tank01.Level01`

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.



This naming convention adds context to the instances contained by `Tank01`.

You can see any object's hierarchical name by opening it and looking at its **Object Information** page.

Additionally, you can use containment references in scripts such as:

- `Me.Outlet`: Allows a script running within the parent object to generically reference its child outlet instance.
- `MyContainer.Inlet`: Allows a script running in any of the children instances to reference another child instance named Inlet that belongs to the same parent.

Use contained names

The contained name of a contained object only has to be unique in the context of its container.

An object can have three kinds of names, depending on whether it is contained by another object. The three names include:

Name	Description
Tag name	The unique name of the individual object. For example, `Valve1`.
Contained name	The name of the object within the context of its container object. For example, the object whose tag name is Valve1 may also be referred to as `Tank1.Outlet`, if `Tank1` contains it and it has the contained name "Outlet".

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Name	Description
Hierarchical name	Hierarchical names that are fully-qualified names of a contained object include the names of the objects that contain it. Because the object that contains it may also be contained, there are potentially multiple hierarchical names that refer to the same object. For example, if: "Reactor1" contains Tank1 (also known within Reactor1 by its contained name "SurgeTank"). "Tank1" contains Valve1 (also known within Tank1 by its contained name "Outlet"). Valve1 could be referred to as: "Valve1" "Tank1.Outlet" "Reactor1.SurgeTank.Outlet".

For example, an instance of a $Tank is named Tank01. An instance of $Valve called Valve01 is contained within the instance Tank01.

Change the contained name of Valve01 to InletValve. Now Valve01 can also be referred to by its hierarchical name Tank01.InletValve. The name of the contained object can be changed, though, within the scope of the hierarchy.

Contained names can be up to 32 alphanumeric or special characters. The second character cannot be $ and the name must include at least one letter. You cannot use spaces.

Containment examples

You can have a Tank object that contains two DiscreteDevice objects that represent its inlet and outlet valves.

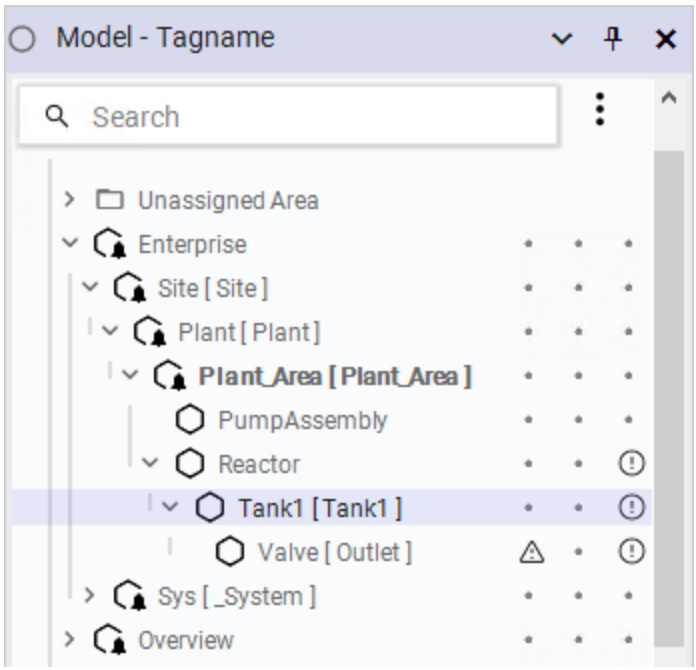
Note: Base templates cannot be contained by another template, either as the container or as the template being contained. You can only use containment with derived templates.

To implement containment

1. Create the following instances: `Tank1` from $UserDefined and `Valve` from $DiscreteDevice. `Valve` has only one name, `Valve`.
2. In the **Model** or **Deployment view**, drag `Valve` on to `Tank1`.

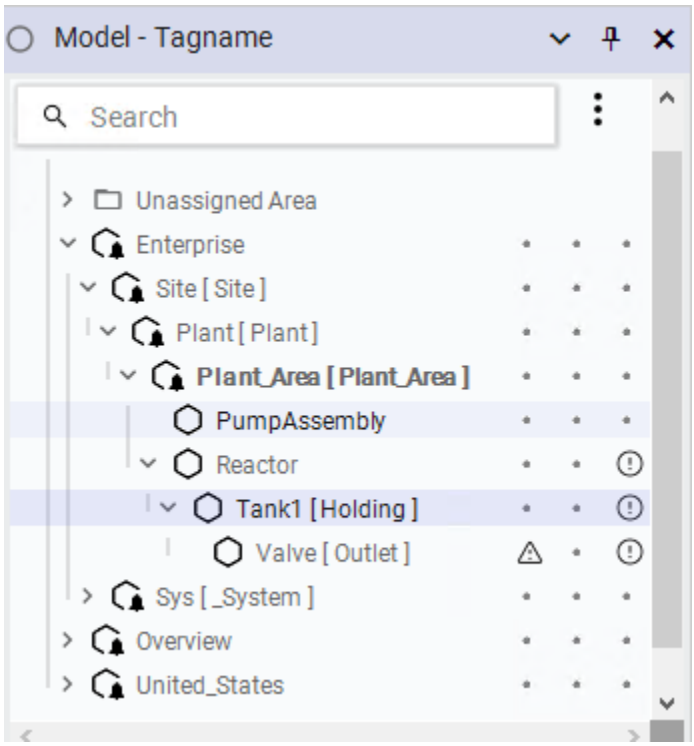
   Note: If `Tank1` already contains an object with a contained name of `Valve`, the Galaxy generates a unique contained name for the newly contained object, such as `Valve_001`.
3. Change the contained name of `Valve` within `Tank1` to `Outlet`. `Valve` can now be referred to by its tagname, `Valve`, as well as its hierarchical name, `Tank1.Outlet`.
4. Create an instance called `Reactor1` from $UserDefined.
5. In the **Model** or **Deployment view**, drag `Tank1` onto `Reactor1`.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.



6. Change the contained name of `Tank1` to `Holding`. `Tank1` now has two names, `Tank1` and `Reactor1.Holding`. Also, `Valve` has a three-part hierarchical name: `Reactor1.Holding.Outlet`.

   For the three objects in this example (`Reactor1` containing `Tank1` containing `Valve`), the following naming hierarchy exists:



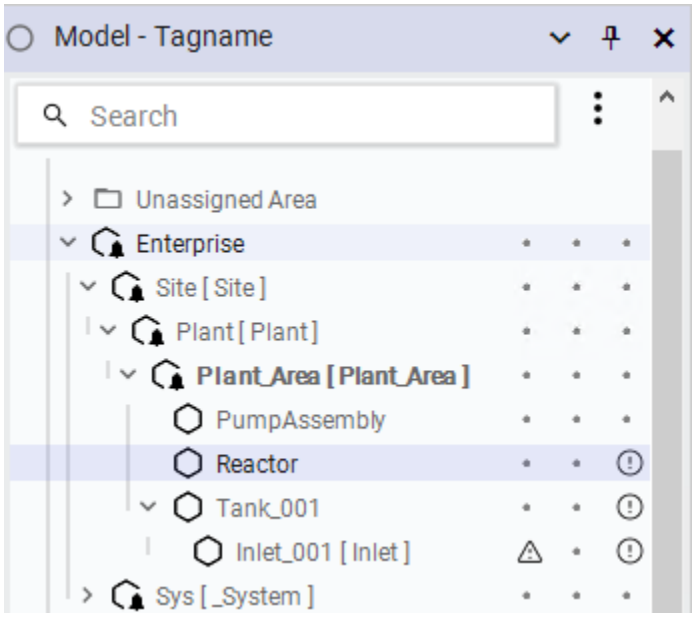
To implement template-level containment

Note: Contained Templates do not have tagnames. When an instance hierarchy is created from a template and its contained children, unique tagnames will be created for the instances based on their contained names.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

1. Create the following derived templates: `$Tank` from $UserDefined and `$Valve` from $DiscreteDevice.
2. Derive `$Inlet` from `$Valve`.
3. In the **Template folder**, drag `$Inlet` on to `$Tank`. If `$Tank` already contains a template named `Inlet`, the Galaxy generates a unique tagname for the new contained template, such as `Inlet_001`.

   The contained template now has a hierarchical name `$Tank.Inlet`.
4. Create an instance (`Tank001`) of `$Tank`.
5. The **Model** and **Deployment views** show an instance `Tank001` that contains an instance called `Inlet`.

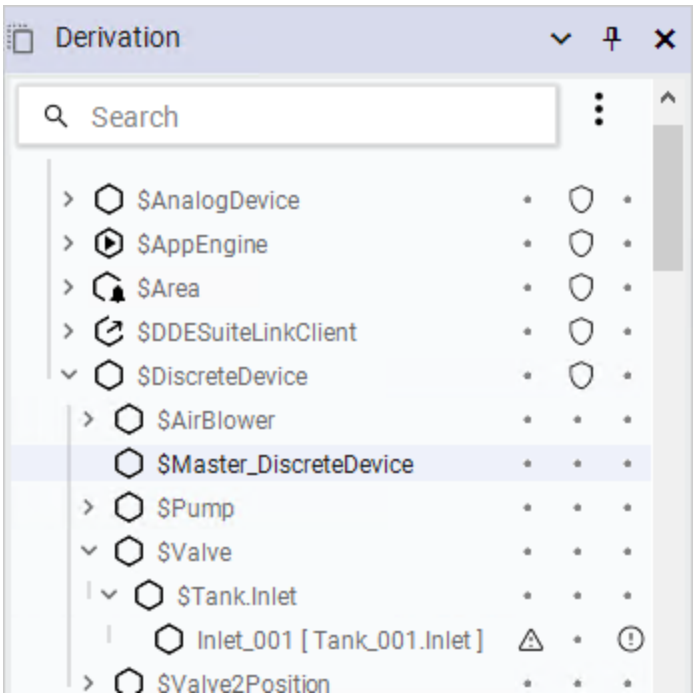


View containment relationships

Containment relationships appear for templates in the **Template folder**. For instances, the relationship appears in both the **Model** and **Deployment views**.

In the **Derivation** view, if a template contains other templates, you can expand it to show the containment under that template.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.



The **Derivation view** shows templates and instances with regard to containment in the following ways:

- Non-contained instances show their tagnames.
- Contained instances show their tagnames and hierarchical names.
- Non-contained templates show their template name.
- Contained templates show their hierarchical name.

Rename contained objects

Before you rename a contained name of an object, make sure that the object is not checked out to another user or currently deployed.

The new contained name must comply with naming restrictions. Template names can be up to 32 alphanumeric or special characters, including the required $ as the first character. The second character cannot be $ and the name must include at least one letter. You cannot use spaces.

Contained names also cannot be the same contained name as an existing contained object within the same level of hierarchy in the containment relationship.

WARNING! Be careful when renaming contained objects. References from other objects to the object being renamed are not automatically updated with the new name. You must update the references. Objects with broken references receive bad quality data at runtime.

To rename an object's contained name

1. Select the object in an Application view.
2. On the **Home** ribbon, in the **Edit** area, select **Contained Name**.
3. Type a new contained name.

   All IDEs connected to the Galaxy show the object's new contained name.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Edit objects

With the Object Editor, you define attributes specific to an object. When you open the Object Editor in non-ReadOnly mode, the object is checked out. No one else can edit an object while you are working with it. If someone else is already working on the object, you can open it to view but you cannot make changes.

The Object Editor contains pages that can be used to modify objects. To view a page in the editor, click its tab. Three pages – **Attributes**, **Scripts**, and **Object Information** – are common to all objects, while other pages are unique to certain object types. If you import a Galaxy or objects with field attributes, the **Field Attributes** page will be present for those objects; the Field Attributes page can also be enabled through a Galaxy user configuration option. See Configure user preferences for more information.

When you save an object, the configuration data for the object is validated. If errors or warnings are identified during validation, a message appears. You can cancel the save or save the object configuration as it is.

- If you cancel, the Object Editor remains open so you can correct the problems.
- If you save the configuration as it is, the object is placed into a bad or warning state. The object's status is marked in the Galaxy database as Good, Warning or Error. Error means the object is undeployable.

To edit an object in the Object Editor

1. Select the object.
2. On the **Home** ribbon, select **Open**, then select **Open**. A red check mark appears next to the object's icon indicating it is checked out, and the Object Editor opens.

   Note: If you are adding I/O attributes to an application object or system object, such as an area object, the preferred way to add and edit I/O references is through the **IO Devices** view and **Mapping** view. Editing I/O references set for automatic assignment in the Object Editor disables automatic assignment. For more information about I/O auto assignment, see I/O auto assignment.
3. Make your changes. For more information about locking attributes, see Attributes page. For more information about setting security, see Set object security.
4. To save your changes and keep the object open for more editing, on the **Home** ribbon, in the **Save** area, select **Save**.
5. When you are finished editing the object, click the **Save and Close** icon at the top left of the Object Editor.

   NOTE: To keep the object checked out, select the **Keep Checked Out** icon before closing.

Object editor user assistance

Tooltips are available in the Object Editor. Point to any editor option and a tooltip appears, showing the attribute name. This name is used when referring to the attribute in scripts, for example.

Each object also includes documentation about usage, configuration, runtime behavior, and attributes. For help with configuring the object, select the object in the Template folder or in an application view and press **Ctrl + F1**, or right-click on the object and select **Object Help** from the context menu. If you want, you can customize the object help file. See Customize help for more information.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Help file structure

The header part of the Help file contains the following information:

Tag name	The object's name.
Contained Name	The object's contained name. For more information, see Create contained templates.
Description	A short summary of what the object is for.
Code Base	The code version of the object.
Derived From	The immediate parent template for the object.
Object Version	The configuration version of the object.
Process Order	The runtime execution order within the host engine's scan (none, before, after) relative to the **Relative Object** element.
Relative Object	The object that runs before or after in the **Process Order**.

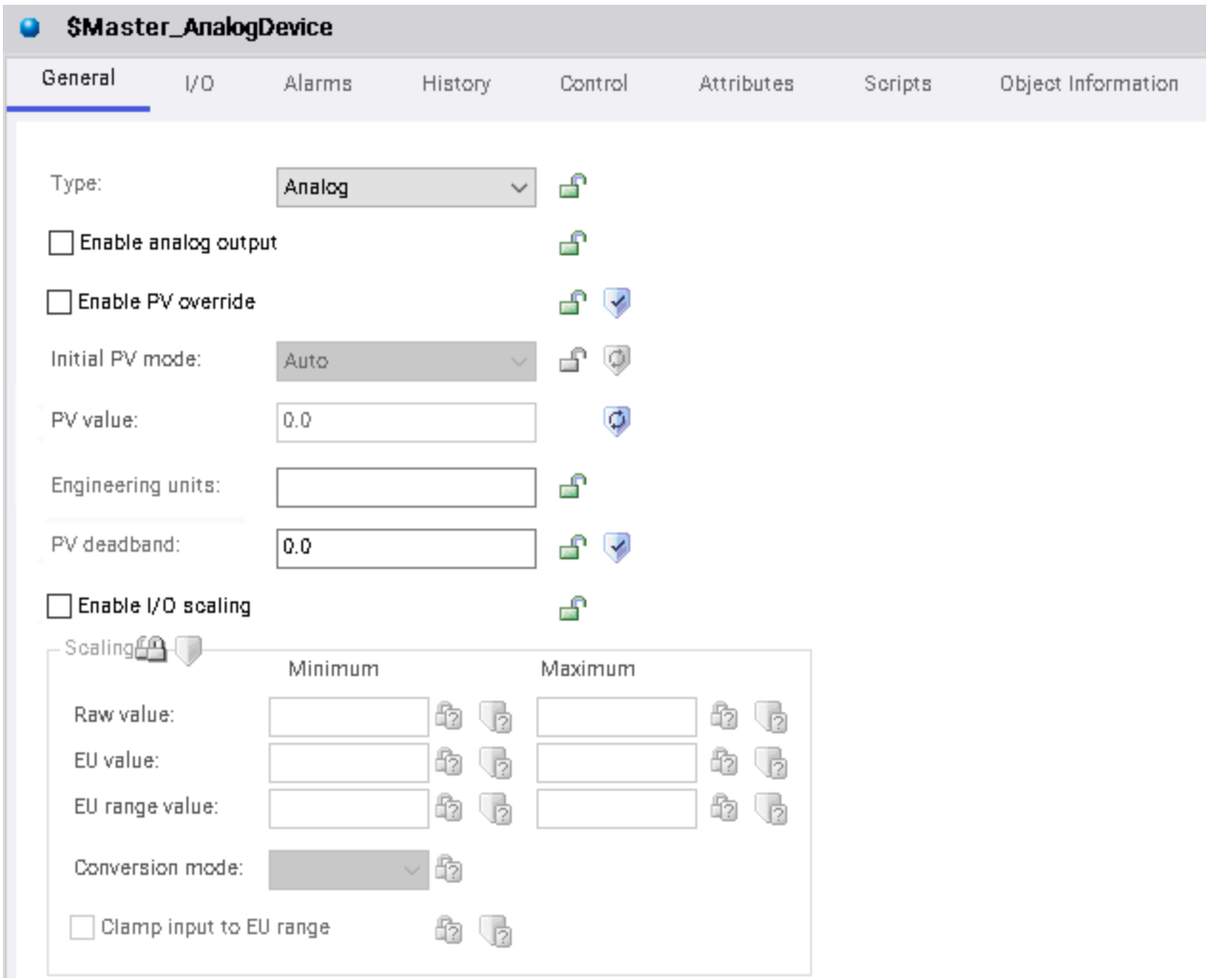
The rest of the help file shows general information about the object.

Object editor layout

When you open the attributes for an instance or a template, you see the Object Editor. The Object Editor is where you configure the object's attributes and add scripts or associated graphics to the object. The Object Editor has several pages related to the type of object you select. If you are working with an instance, you see different pages than if you are working with a template.

For example, the following illustration shows an analog device template.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.



When you open the Object Editor, the object is automatically checked out so no other user can work on it. When you close the Object Editor, the object is checked in to the Galaxy database, if it was automatically checked out when the editor was opened.

	To keep the object checked out, click **Keep Checked Out** before closing.
	To save configuration changes you made and check the object back in, click the **Save and Close** icon.

After the object is checked in, other users can edit it.

Lock and unlock template attributes

When you create derived templates, you can lock or unlock some or all of the attributes and enabled features. Locking an attribute prevents it from being changed in derived templates or instances. You can only lock attributes in templates.

Locking an attribute in a template specifies that its value or setting is inherited by all derived objects, both templates and instances. Locking an attribute also makes the attribute act as a constant during runtime.

Note: When you lock values and settings for an attribute, they are not visible when you use Object Wizards, and

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

thus the locked values and settings will be inherited by all derived objects.

	In the **Attributes** page, you must enable locking before an attribute or any of its features can be locked in a template. To enable locking, click the **Show/Hide Lock** icon to the right of the attribute name. When locking is enabled, lock symbols will appear next to values that can be locked. Lock symbols are not visible in the template or its derived instances unless locking is enabled.

- Attributes that are locked in a parent template are referred to as "locked in parent." This parent can be at any parent level above the selected object.
- Attributes that are locked in a template are referred to as "locked in me."

  Caution: When using I/O auto assignment, do not lock the "Read from" field (input source) or "Write to" field (output destination). If you lock these fields in the parent template, they cannot be updated with the resolved reference when the object is deployed and the runtime value will be "---Auto---".

See Group lock and security controls for information about locking or unlocking all of a feature's attributes with a single click.

Lock controls and status are shown with an icon. If the option is enabled, click the lock control to switch it between locked and unlocked. These icons mean:

Icon	Name	Description
	Locked (in me)	The associated attribute is locked (in me) and enabled. Only templates can have this kind of lock. The attribute value is read/write. Derived templates and instances do not have a unique copy of this attribute. Child objects share the locked attribute of the parent. Changing the value of a locked attribute in the parent template updates the value of that attribute in all derived templates and instances.
objeditorlock2	Locked (in parent)	The associated attribute is locked in the parent object and cannot be unlocked or modified by the child object. Both templates and instances can have these. The attribute is read-only. The templates and objects do not

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Icon	Name	Description
		have a unique copy of this attribute, but instead use the attribute value in the parent where the attribute is locked.
	Unlocked	The associated attribute is unlocked and enabled. Both templates and instances can have this kind of lock. The attribute is read/write. The object has its own copy of the attribute value and the value is not shared by derived objects.
	Indeterminate	Refers to a specified group of options. An indeterminate state indicates different lock states for individual options in the group.
	Undefined	The associated attribute doesn't exist. This indicates that another attribute must be enabled before the associated attribute is created and before its lock status can be determined.

Note: Locking an attribute during configuration makes its value a constant. You cannot write to locked attributes during runtime.

Example: Lock an attribute

1. Create a derived template from the $DiscreteDevice base template. Name the derived template `$Valve`.
2. Open the `$Valve` template and on the **States** page, lock one of the attributes by clicking the **Lock** icon.
3. Save `$Valve`.
4. Create a derived template from `$Valve`. Name it `$BigValve`.
5. Create an instance from `$Valve` named `Valve1`.

   In the editor of `$Valve`, the attribute lock icon shows the attribute is locked in `me`.

   You cannot change the attribute value in `$BigValve` and `Valve1`. The editor options for the attribute are disabled and the lock icon, if shown, indicates a lock in the parent. Also, the attribute lock icon in children derived from `$Valve` is now locked and disabled.

   If you change the attribute value in `$Valve`, the change propagates to `$BigValve` and `Valve1` after you save the changes.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Example: Unlock an attribute

1. Using the objects from the previous example, in the `$Valve` template's editor, unlock the locked attribute.
2. Save `$Valve`.

   In the editor for `$Valve`, the attribute lock icon shows it is unlocked.

   The lock type for this attribute of `$BigValve` now indicates locked in `me`. The lock type for this attribute of the `Valve1` instance shows unlocked but the locking icon is unavailable.

Set object security

Operators interact with objects through the individual attributes of those objects. Each attribute on the Object Editor that can be modified by operators at runtime can have an associated security control, which is used to modify its runtime security classification.

	In the **Attributes** page, security icons must be enabled before you can set security for an attribute or any of its features in a template. To enable security, click the **Show/Hide Security** icon to the right of the description field.

When security is enabled, security symbols will appear next to values for which security is configurable. Security symbols are not visible in the template or its derived instances unless enabled in the parent template.

If an attribute's security classification is configurable, click the security control to select one of seven possible states:

Security Icon	Description
seciconfree	Lets you change this value without restriction even if you have no defined permissions on the object. Anyone can write to these attributes to perform safety or time critical tasks that can be hampered by an untimely logon request, such as halting a failing process.
seciconop	Lets you work with Operate permissions to do certain normal day-to-day tasks. These include writing to attributes like Setpoint or Command for a Discrete Device object. This level of security requires you to have Operate permission for the security group for the object.
seciconsw	Requires you to authenticate using your user name and password each time you want to write to the attribute. You also need to have Operate permissions for the object.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Security Icon	Description
seciconvw	Requires you to have Operate permissions to log on again and a second, different user to also log on as the verifier before writing to the attribute. The verifier needs to have Can Verify Writes permission for the object.
secicontune	Allows end users with Tune Operational permissions to tune the attribute in the runtime environment. Examples of tuning are attributes that adjust alarm setpoints and PID sensitivity.
seciconconfig	Allows end users with Configure Operational permissions to configure the attribute's value. Requires that the user first put the object off scan. Writing to these attributes is considered a significant configuration change. For example, a PLC register that defines a Discrete Device input.
seciconro	Only allows users to read this attribute's value in the runtime environment. This attribute is never written to at runtime, regardless of the user's permissions.

If an attribute's security is shown in gray, its security classification is locked in its parent object and cannot be changed, or it requires the enabling of a group attribute.

Group lock and security controls

The lock and security controls associated with option groups and features quickly set those conditions for all options in the group.

	The group control typically reflects the setting for all options in the group or feature set. But, if at least one option in the group has a lock or security control that is different from the other options, the group control shows an indeterminate icon.

In addition to the undefined controls, the group controls for locking and security are the same as those for individual options.

You can lock or unlock all of the attributes associated with a feature by selecting the lock icon next to the feature name, after you activate the feature. This will lock or unlock all of the attributes for the feature, unless an attribute was locked at a higher level. For example, if you locked an attribute in a template, and then created another derived template, the attribute that was locked in the original template cannot be unlocked in the derived template (it is locked in parent).

If an attribute is locked in the template, you can change the value in that template, but not in the derived children. If you change the value in the parent template, the change propagates to all child objects. For more information, see Configure objects.

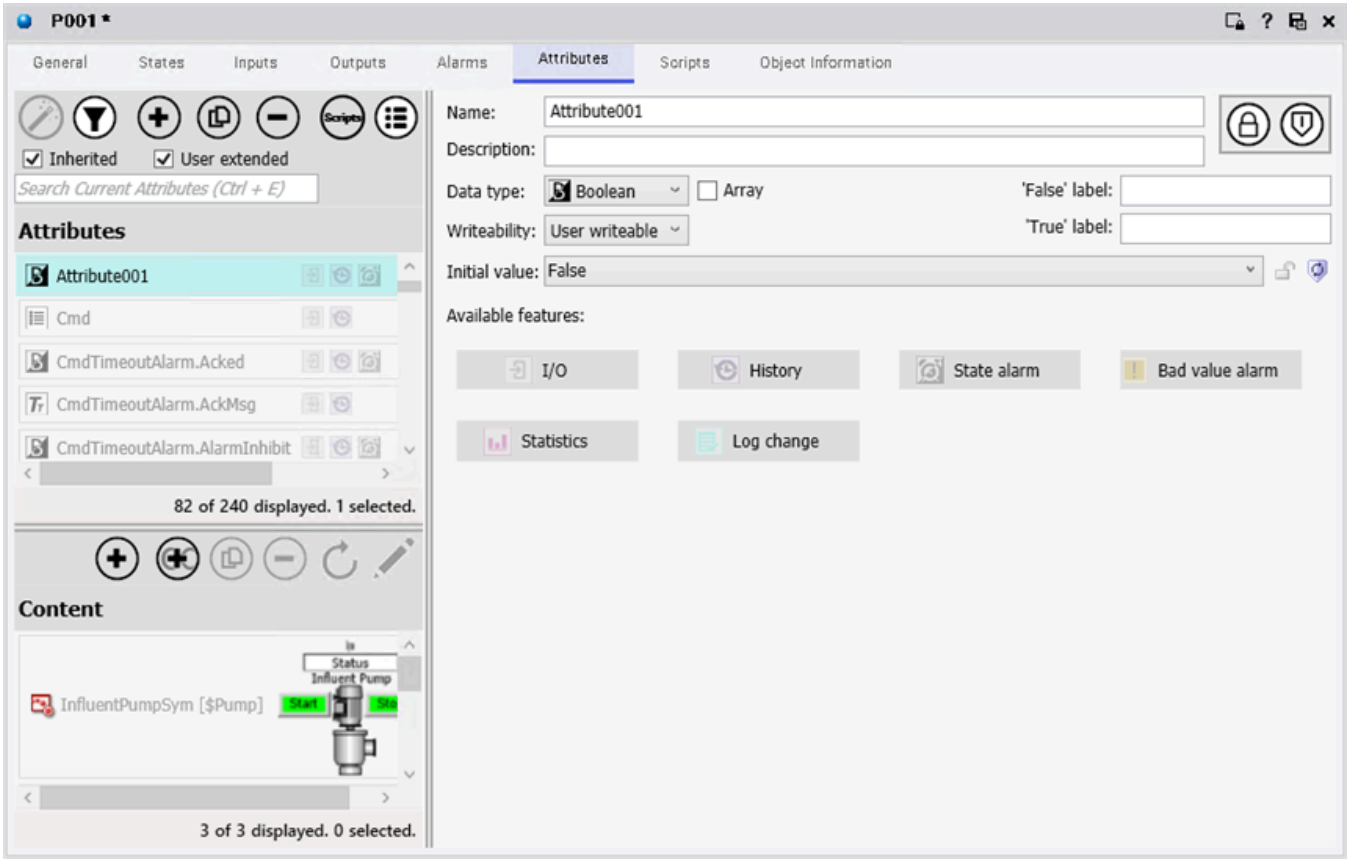
© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Attributes page

The **Attributes** page divides into two or more panes when you initially open it. The number of panes the **Attributes** page divides in depends on whether or not attributes or graphics have been defined for the object, whether the object is an instance or a template, and if it is an instance, whether or not it is derived from a template that includes an Object Wizard.

If the object you are editing does not have any attributes defined, the **Attributes** page will be empty. The **Attributes** page will also show a pane for graphics, and depending on the type of object (template or instance) and its contents, may show additional Object Wizard configuration panes.

You can activate various features, such as I/O, History, State Alarm and Statistics. See Configure objects for more information about attributes and features. When you add an attribute to an object, information about the attribute is shown.



	Opens the **Object Wizard** editor.
	To filter attributes, click the **Filter** button. You can then select filtering criteria by checking source type, enabled feature type, writeability type, lock status, data type, visibility type (hidden or not hidden), and diagnostic type (configuration, runtime, or both).

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

	**Adds** a new attribute.
	**Duplicates** the selected attributes.
	**Removes** the selected attributes.
	To display the **Scripts** pane used to associate scripts with an Object Wizard, click the **Scripts** button.
	**Shows Details** about an attribute. The name, description, and icons representing activated features for the object's attributes will be shown.

When you add an attribute to an object, the **Attributes** page divides into three sections. The left side of the page lists attributes, the top right shows information about the currently selected attribute, and the bottom of the right side contains fields for configuring features.

You can search attributes by entering the characters you are trying to find in the **Search Current Attributes** text box. This will display attributes that contain the characters you enter.

Note: The search function finds attributes that contain the search term anywhere within the attribute name. For example, if you want to locate all attributes associated with Pump125, you could enter "125" in the text box. All attributes with 125 anywhere in their names will be shown.

You can also search for attributes by typing the character you are searching for when the cursor is pointed at the attribute list. The first attribute that contains the character, anywhere within its name, will be highlighted.

Note: The search begins at the attribute that is currently selected. Selection will move to the next attribute that contains the character entered.

- On the left side of the page, all attribute names for the object are listed. Pressing the **Show Details** button reveals additional information about the listed attributes. Check boxes are provided for **Inherited** and **User extended**.
  - **Inherited**: Checking this displays attributes that were added in the parent template used to create the object.
  - **User extended**: Checking this displays details about any features you added to the attribute, for example, for I/O.
- At the top of the right side, information for the current attribute is listed. Additional fields may be listed, depending on the data type of the attribute. However, all data types include the following properties:

Property	Description
Name	Attribute name (for example, Attribute001).
Description	User defined; a description of the attribute.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Property	Description
Data type	Can be set to Boolean, Integer, Float, Double, String, Time, ElapsedTime, or InternationalizedString. The data type determines which features can be added to an attribute. For example, when configuring a Boolean data type, only two alarm features are available: **State Alarm** and **Bad Value Alarm**. When configuring an Integer data type, there are four available alarm features: **Limit Alarms**, **ROC Alarms**, **Deviation Alarms** and **Bad Value Alarms**. To create an array, check the **Array** option and specify the array's length in the **# of elements** box. You can create an array for each data type except **InternationalizedString**.
Writeability	Can be set to Calculated, Calculated retentive, Object writeable, or User writeable. See the *Object Viewer User Guide* for additional information about writeability categories. • Calculated: Permits only scripts within the same object to write to the attribute. Calculated attributes are not saved across restarts. If you select Calculated for an attribute, only scripts running on the same object can write to the attribute. • Calculated Retentive: Permits only scripts within the same object to write to the attribute. Calculated Retentive attributes are saved across engine restarts. • Object Writeable: Permits other objects to write to this attribute in addition to being set by scripts within this object. Object Writeable attributes are saved across restarts. This category is not user writeable. • User Writeable: Permits other users to write to this attribute in addition to being set by scripts and objects throughout the system. User Writeable attributes are saved across restarts. They can be locked at configuration time. If locked, they are read-only.
Initial value	Specifies the initial value for the attribute when the object is deployed. Enter value data for each data type. In the case of a non-arrayed **Boolean**, select **True** or **False** in the list box. For an arrayed Boolean,

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Property	Description
	select the appropriate checkbox.

Some attribute text boxes may show --- (dash dash dash). The --- is a placeholder reference that does not cause the associated object to be placed in a warning configuration status when it is validated. You may also see attribute text boxes showing a ---.--- (dash dash dash dot dash dash dash). You need to provide a valid reference in the text box. The ---.--- placeholder causes the associated object to be placed in a warning configuration status when the associated object is validated.

Important: Edit the I/O auto assignment placeholder in the Object Editor ONLY if you do NOT want to use I/O auto assignment for the object. In most cases, I/O auto assignment is the preferred method.

- At the bottom of the right side, fields for configuring activated features are displayed. Available features include:

Feature	Description
	The I/O feature is used to activate the attribute for I/O. You can select **Read** (input), **Read/Write** (input and output), or **Write** (output). When you select the I/O feature for application objects and system objects, such as areas, the "Read from / Write to attribute text box contains the placeholder reference `<IODevice>.[HierarchicalName].[AttributeName].InputSource` or `<IODevice>.[HierarchicalName].[AttributeName]. OutputDest`. You should not edit these placeholders in the Object Editor, but instead use the **IO Devices view** to assign the object to a scan group. This will allow the placeholder to automatically resolve to the correct I/O reference. See I/O auto assignment for additional information.
	The History feature enables historization to the Historian. **Note**: The AppEngine hosting the object must be configured for historization.
	The State Alarm feature is available for Boolean data types. You can set alarm priorities, the alarm state, an alarm message, and time deadband.

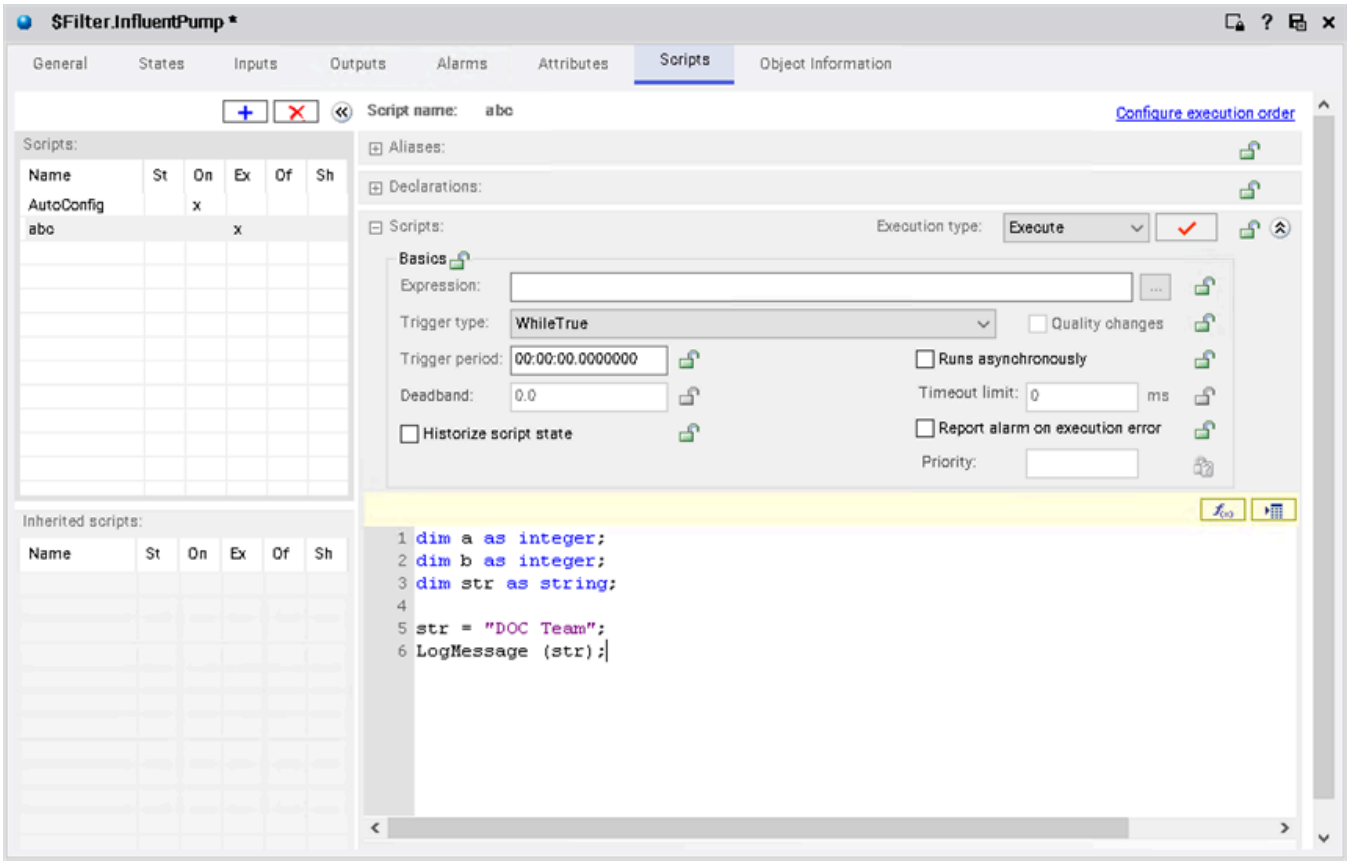
© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Feature	Description
Limit alarms	The Limit Alarms feature is available for integer, float, and double data types. With it, you can set value alarms (Hi, HiHi, Lo, LoLo), and the limits, priority, and messages to apply to each alarm limit.
ROC alarms	The Rate of Change (ROC) Alarm feature is available for integer, float, and double data types. With it, you can set alarms if changes within a time period exceed specified values, and the limits, priority, messages, and time period to apply to each ROC alarm.
Deviation alarms	The Deviation Alarm feature is available for integer, float, and double data types. With it, you can set deviation alarms (major and minor), and the tolerance, priority, messages, target value, deadband, and settling period.
Bad value alarm	The Bad Value Alarm feature adds an alarm if the value returned from the attribute is determined to be bad quality.
Statistics	The Statistics feature monitors statistics associated with the object.
Log change	The Log Change feature will cause the attribute to generate an event each time the attribute value changes.

Scripts page

The **Scripts** page is divided into several areas. To learn more about using scripts, see Write and edit scripts.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.



The main areas of the **Scripts** page include:

- **Scripts** list: Shows all scripts currently associated with the object. The columns indicate which kind of trigger the script uses: Startup, On Scan, Execute, Off Scan and Shutdown. Click the **Add** button (+) to add a new script.
- **Inherited scripts name** list: Shows all scripts associated with the object's parent. The columns indicate which kind of trigger the script uses: Startup, On Scan, Execute, Off Scan and Shutdown.
- **Aliases** area: Lets you create and modify aliases that apply to the script you are working on. Aliases are logically descriptive names for typically long reference strings that you can use in the script to make the script more readable.
- **Declarations** area: Provides a place to add variable declaration statements, such as `DIM MyArray[1] as FLOAT;`. These declared variables live from the start to the shutdown of the object and can be used to hold values that persist from one execution of the script to the next. They apply only to the script in which they are declared.
- **Basics** area: Provides a location in which you set the expression, triggering conditions, and other settings that run the script in the runtime environment. See Write and edit scripts for descriptions of triggers and when they are executed. This area includes:

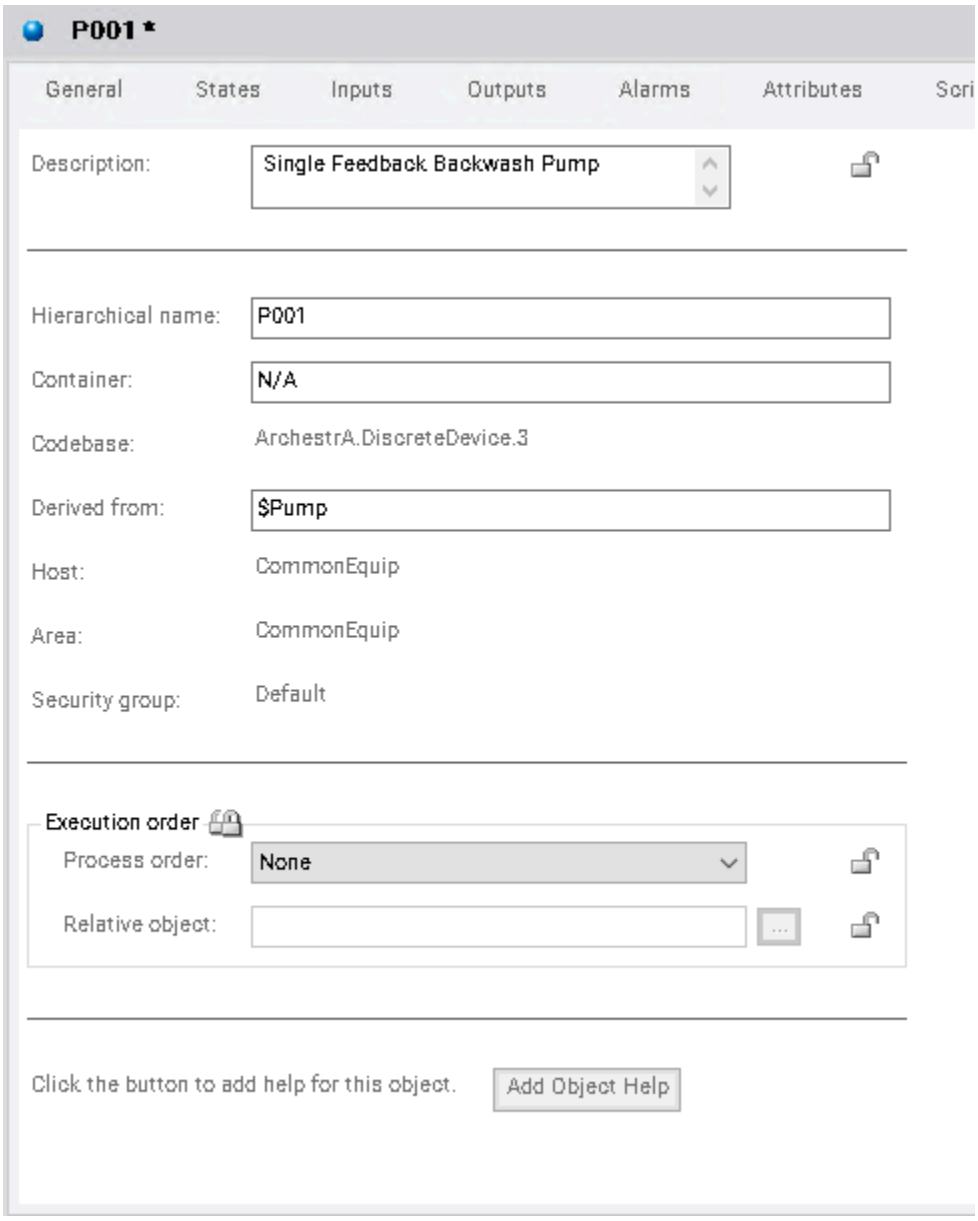
  Configure Execution Order: Sets the execution order of multiple scripts (inherited and local) associated with this object.

  Historize Script State: Select to send the state of the script to the Historian.
- **Script Creation** box: Shows the script you are writing.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Object information page

The **Object Information** page is common to all object configuration editors.



This page includes the following fields:

- **Description**: A short summary of the object's purpose.
- **Hierarchical Name**: The fully qualified name of a contained object, including the container object's TagName.
- **Container**: The name of the other object that contains this object, if applicable.
- **Codebase**: The code version of the object.
- **Derived From**: The immediate parent template of the object, either a base or derived template.
- **Host**: Another object to which the object is assigned (for example, a WinPlatform hosts an AppEngine). An object's host determines where an object will run when it is deployed.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

- **Area**: An object that represents a logical grouping to which this object belongs. An object's area mostly affects the way in which its alarms are reported to alarm clients.
- **Security group**: The security group the object is associated with.
- **Execution order**: If you want this object to run before or after another object within its engine's scan, select from the **Process order** list. Click the **Browse** button to specify the **Relative object** in the **Attribute Browser**. For more information about the **Attribute Browser,** see Use the galaxy browser to reference objects .
- **Add Object Help**: Opens a copy of the HTML help page for the template this object is derived from. You can edit this information. This allows you to create Help about the object you are currently configuring for downstream users. This Help appears when you select an object in a view and then click **Object Help** on the **Help** menu. See Customize help for more information.

Customize help

Do not use Microsoft Word as an editor to create downstream object HTML help pages. Use an HTML or plain text editor instead.

If clicking **Add Object Help** opens Word on your computer, change the program associated with editing HTM files. Open the Windows Explorer's **Folder Options** dialog box and go to the **File Types** page to make this change. For more information about associating programs with files, see your Windows help.

Locate the help folders

The path to each object's Help folder is unique. It depends on the path you selected when you installed the Galaxy Repository. The default path to an object's Help is:

```
\<Installation Path>\Program Files (86)\ArchestrA\Framework\
FileRepository\<YourGalaxyName>\Objects\<TheObjectID>\Help\1033
```

To add images to the Help file, place the images in the proper folder on the Galaxy Repository computer and use a relative path to those images in the HTML file.

For the previous example, place images in the `\1033` folder or create an images folder under it.

Field attributes, UDAs, and extensions pages

Field Attributes, UDAs and attribute extensions are deprecated and have been combined into a single unit of functionality simply called an Attribute. The functions and capabilities of the **Field Attributes**, **UDAs**, and **Extensions** pages in the **Object Editor** have been combined and made available in the **Attributes** page.

The **UDAs** and **Extensions** pages no longer exist in the System Platform IDE. By default, the **Field Attribute**s page does not normally appear, but can be configured to do so.

View the Field Attributes Page

While the **Field Attributes** page is not displayed by default, it will be displayed when you open a UserDefined Object in the Object Editor and any of the following conditions apply:

- The derived template or instance created from the $UserDefined base template contains Field Attributes.
- You import UserDefined Objects created in a previous version of Application Server that contain Field

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Attributes.

- You have an old, existing Galaxy with Field Attributes already defined and have migrated it.
- You select the **Show Field Attributes** option on the **Preferences** page. See Configure user preferences for more information.

Determine whether to convert field attributes

Converting Field Attributes to Attributes provides the following advantages:

- It is easier and faster to work with Attributes, rather than Field Attributes.
- All of your Attributes will appear on the same Object Editor page if you convert Field Attributes.
- Attributes can provide performance advantages over Field Attributes during run time (the actual performance improvement can vary between systems).

You may not want to convert Field Attributes if:

- You have a large number of objects that use Field Attributes.
- You do not want to modify and test scripts that reference Field Attributes.
- The objects are working well using Field Attributes, and there is no active development on these objects. Additional effort will be required for converting, testing, and redeploying objects.

Optimize performance if you use field attributes

If you choose to continue using Field Attributes, use the following guidelines to maximize performance:

- No more than 64 Field Attributes (maximum of 32 analog and 32 discrete) should be used for an individual object. If this limit is exceeded, the performance of object editing and check-in operations slows down significantly.
- Do not extend an existing Field Attribute by activating features on the **Attributes** page.

  For example, to add historization to an existing Field Attribute, select the **Enable history** option from the **Field Attributes** page. DO NOT use the Attributes page to add the History feature.
- Converting from Field Attributes to Attributes will greatly enhance performance, particularly if many Attributes are used.

Convert field attributes to attributes

You can choose which objects with Field Attributes to convert to Attributes, but only first-level derived objects can be selected. That is, you can only select templates and instances that are directly derived from the $UserDefined base template. Objects derived from first-level templates or individual Field Attributes cannot be individually selected for conversion.

When you start the conversion process, you are presented with a list of all first-level templates and instances that contain Field Attributes. First-level templates without Field Attributes, but that have derived objects with Field Attributes, are also listed. From this list, you will select which objects to convert.

The conversion process is hierarchical and cascades to all child templates and instances of the selected objects. All Field Attributes contained in a template's hierarchy of derived objects are converted.

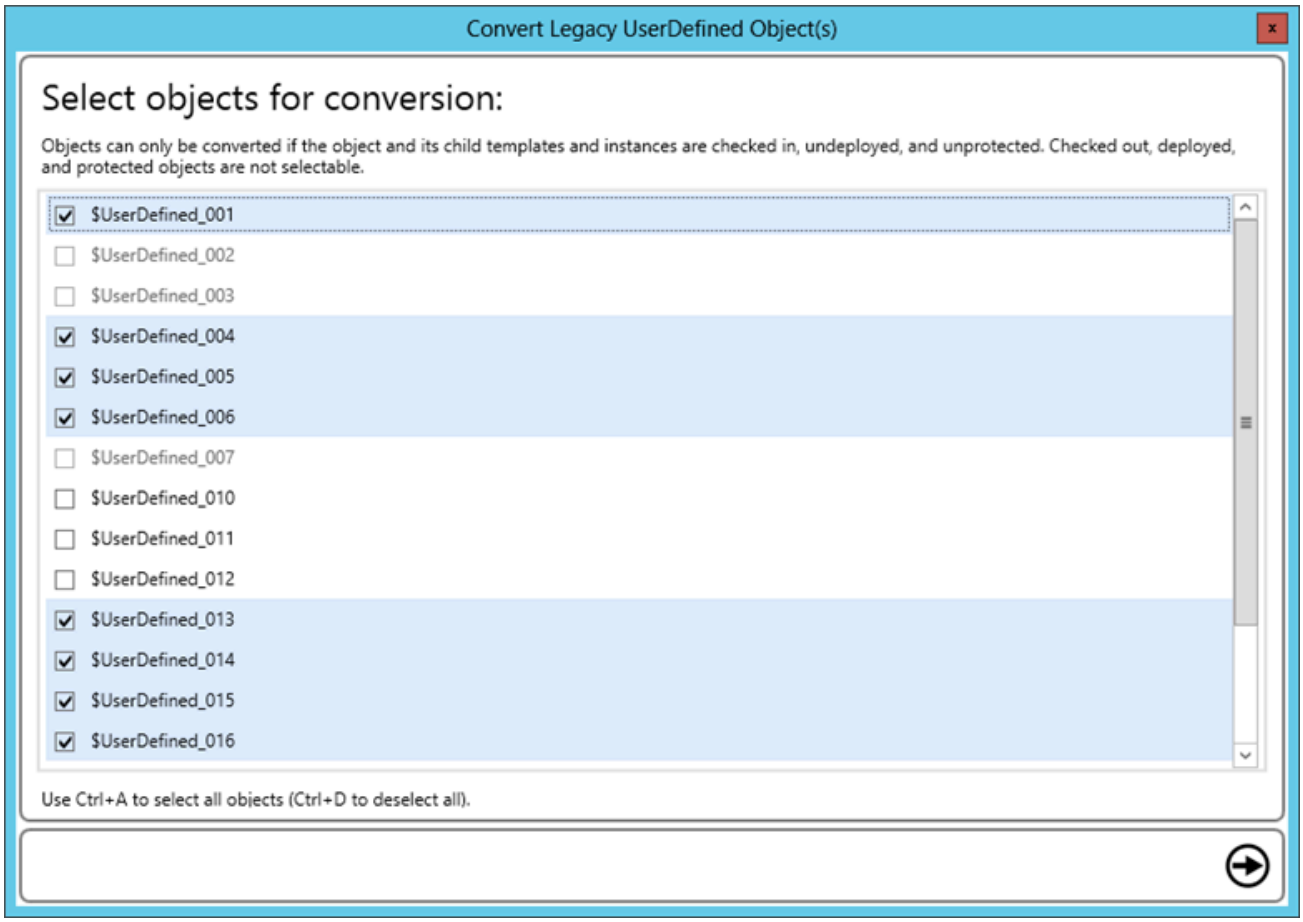
© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Note: Objects will be unavailable for selection (grayed out) if the object or any of its derived objects are deployed, checked out, or protected. See Protect objects on export for more information about protecting objects.

To convert field attributes to attributes

Before starting the conversion process, see Special considerations when converting field attributes for a discussion of potential differences in behavior between converted Attributes and the original Field Attributes.

1. Back up all UserDefined Objects (UDOs) before starting Field Attribute conversion.
2. Ensure that all UDOs that you want to convert are checked in and not deployed. You will not be able to convert Field Attributes for objects that are deployed, checked out, or protected.
3. From the ribbon, select **Galaxy**, then select **Configure**, **Integrated Development Environment**, **Convert UDO**.
4. A list of all first-level objects with Field Attributes is displayed. First-level templates that do not themselves include Field Attributes, but have child objects with Field Attributes, are also listed.

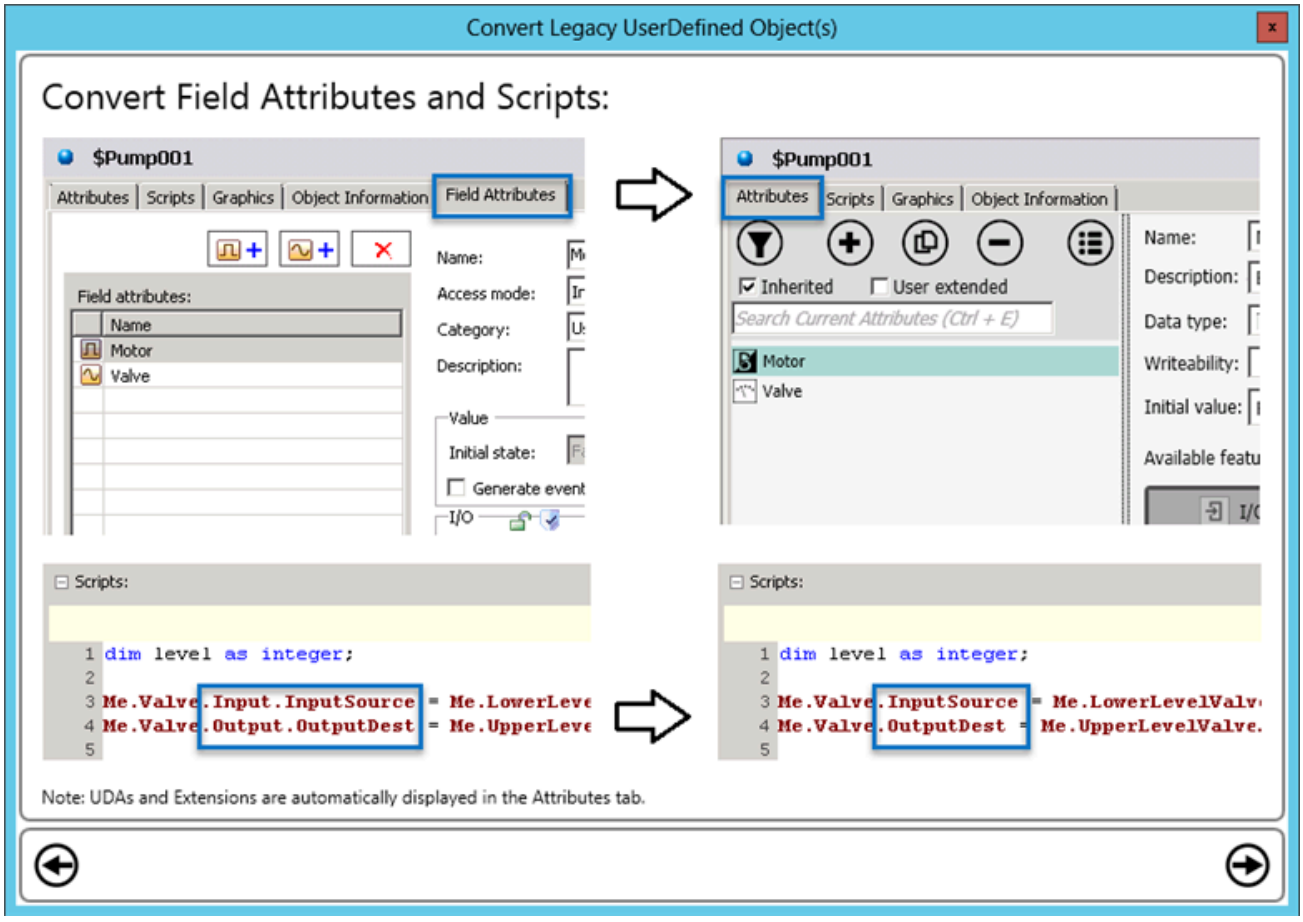


5. Select the objects you wish to convert and click **next**.

   Note: Objects will be grayed out and unavailable for selection if the object or any of its derived objects are deployed, checked out, or protected.

6. The next screen is informational, and shows that Field Attributes will be consolidated onto the Attributes page. Click **next** to proceed.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.



7. The screen that follows is informational as well, and lists the limitations that apply to script conversion. These are:
   - Only field references that begin with "me." are converted.
   - Field references outside the selected objects are not converted.
   - Field references in scripts that use indirect functions or string manipulations are not converted.
   - Field references within Industrial Graphics are not converted.
8. Click **next** to proceed.
9. To begin converting Field Attributes for the selected first-level objects and their child objects, click the **Convert** button ( ) at the lower right.

   WARNING! Once you start the conversion process, it cannot be cancelled. The time needed to convert Field Attributes depends on the number of objects with Field Attributes, the number of Field Attributes, and the depth and complexity of the object hierarchy. Field Attribute conversion requires at least as much time as Galaxy migration.

   - Click **back** if you need to make any changes.
   - To cancel, simply close the dialog.

   WARNING! While the conversion process is active, you cannot deploy or edit objects.
10. Each object is listed as its Field Attributes are converted. The status of the conversion process is also reported to the OCMC Logger.

    Important: In the event that Field Attribute conversion fails for an individual template, the template will

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

remain checked out after the conversion process stops. Undo the checkout for the template, and then you can resume attribute conversion for any remaining objects. To find which template did not convert, check the OCMC Logger.

11. When all objects containing Field Attributes have been converted, click **Close**.

Special considerations when converting field attributes

When you convert Field Attributes to Attributes, there may be some differences between the original Field Attribute and the converted Attribute.

- If the Field Attribute "Bad.Condition" was extended (for example, by adding historization), it will send alarm notifications after conversion, even if alarming was not enabled.
- Prior to conversion, the Field Attribute description is `Analog_001.Desc.` After conversion, this changes to `Analog_001.Description.`
- Prior to conversion the History Extension description points to `Analog_001.Desc.` After conversion, the History Feature points to `Analog_001.Hist.DescAttrName.`
- If a Field Attribute has "Generate event upon change" enabled, then the **Log change** feature (`LogDataChangeEvent`) will be added to the converted attribute.
- When the Field Attribute I/O option, "Output destination differs from input source," is selected and left unlocked, but **Output destination** is locked, the converted Attribute will have the following locking properties:
  - "Write to" will be locked. This is the same as the original Field Attribute property.
  - "Output destination differs from source" will be locked. This differs from the original Field Attribute property.

  Note: Default names for Field Attributes begin with either "Discrete" or "Analog." The conversion process preserves this part of the Field Attribute name, even though this division does not apply to the converted Attributes.

Limitations and exclusions when converting field attributes

Objects that are protected, checked out, or deployed will not be converted. If an object derived from a first-level template you want to convert is protected, checked out, or deployed, the first-level template will not be selectable for conversion. The entire hierarchy of the first-level object must be available for conversion (conversion works on an all or none basis).

The following Field Attribute features are not converted to Attribute features. You will have to be enter the information for them manually.

Field Attribute Feature	Equivalent Attribute Feature
Enable Deadband	Limit Alarms
Description	Description
Engineering Units	Eng units

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Update scripts to reference converted attributes

The conversion process applies certain changes to scripts that contain I/O references. In the conversion process, the Field Reference I/O namespace is truncated to match the Attribute namespace. These reference changes are as follows:

- `Me.Analog_001.Input.InputSource` converts to `Me.Analog_001.InputSource`
- `Me.Analog_001.Output.OutputDest` converts to `Me.Analog_001.OutputDest`
- `Me.Analog_001.Input.Value` converts to `Me.Analog_001.Value`

  Note: Only Field Attributes that begin with "Me." are converted. No other changes are made to scripts. Scripts within graphics are not converted.

While Field Attribute conversion updates I/O references within scripts, other attribute references are not updated. For example, while a Field Attribute description is truncated to "Desc," the description is not truncated for Attributes. Therefore, you will have to do some manual updating.

Use the galaxy browser to reference objects

Use the Galaxy Browser to browse for:

- Attributes of objects. You can quickly find an object attribute or attribute property and add a reference to it when you are configuring an object.
- Industrial Graphics.
- Graphic element properties.

The Galaxy Browser shows attributes, graphics, or attributes and elements, depending on what you are doing at the time you access the browser.

Browse for attributes

You use the Galaxy Browser to browse for:

- Attributes of objects, either instances or own relative references.
- Attributes of templates.

You can open the Galaxy Browser to browse for attributes from:

- Within an AutomationObject Editor. For example, from a script, from an attribute of type MxReference, or from a custom alarm message attribute field).
- Within an Industrial Graphic Editor. For example, to use in scripts, animation links and references, properties and custom properties.
- Within InTouch WindowMaker. For example, to use in a reference expression from an animation link or a script.

The Galaxy Browser shows objects in the left pane and the attributes associated with the current selection in the

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

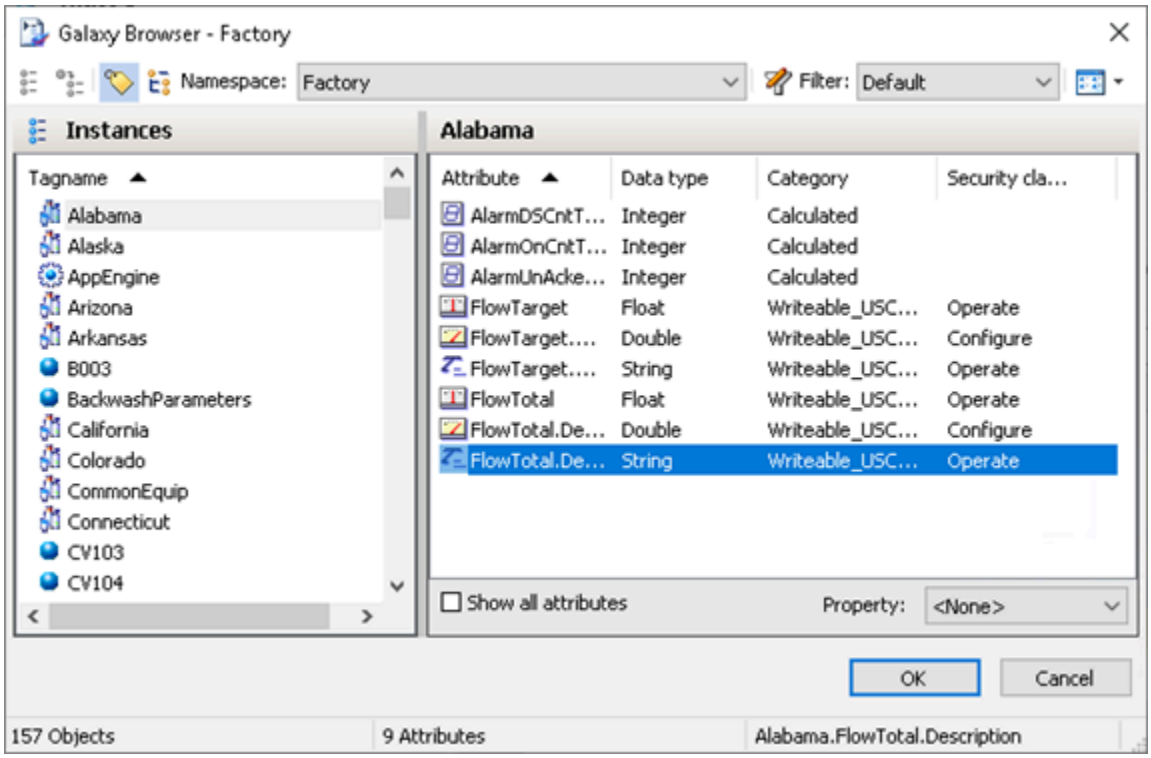
right pane. Only attributes that can be referenced at runtime are shown. You can browse "Me." references for alarm messages only.

When you open the Galaxy Browser, the last browsed location for attributes is shown. The Galaxy Browser shows the object list based on the last used state (Tagname or Hierarchical name). If the last used state of the browser was Tagname and the selected editor reference is a Hierarchical name, the browser opens in Tagname mode.

The status bar displays the attribute property name and the graphic element attribute name and description.

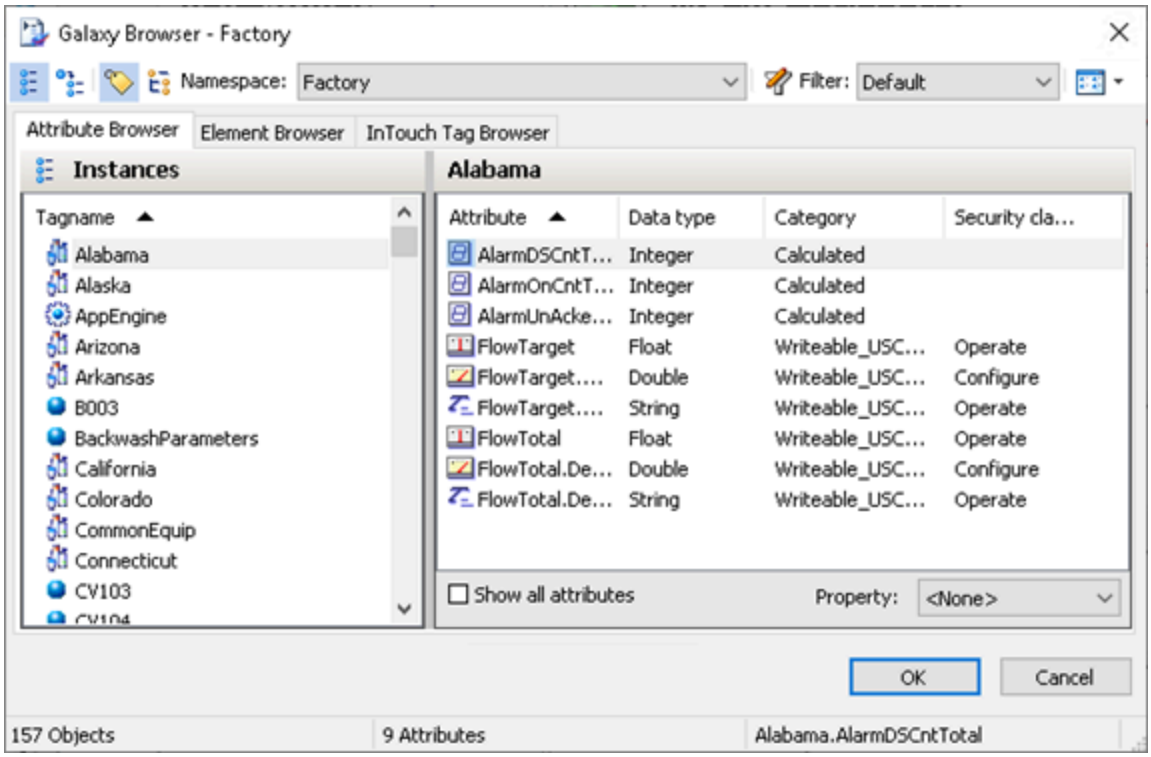
To browse for attributes

1. In any area on a page, click the **Browse** button (...), if available. The Galaxy Browser opens.



If you are browsing for attributes to use with an Industrial Graphic, such as for an animation or script, the Galaxy Browser shows the attributes in an **Attribute Browser** tab.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.



2. By default, the browser shows only those attributes that are frequently accessed. If you are viewing the attributes of an object for the first time, the right pane can be blank. Select the **Show all attributes** check box to show all of the object's attributes.
3. To filter the list of tagnames, click the **Filter** button. For information about configuring a filter, see Create a filter for the galaxy browser. To switch the content of the left pane between a **Tagname** and **Hierarchical Name** list of objects, click the **Show Tag name** or **Show Hierarchical name** buttons.



4. You do not have to explicitly make a selection in the **Property** list. If you only select an attribute (leaving **Property** set to <none>), the property of the attribute defaults to **Value**.

   If the option in the Object Editor is already configured with an object reference, the **Attribute Browser** shows it or expands to the nearest matching object/attribute/property currently configured in the Galaxy.

   If you selected text in the script editor, that text is used as the initial reference string and the browser finds the nearest attribute reference to the selected text
5. When you are done selecting the attribute/property, click **OK** to place the reference into the Object Editor and close the **Galaxy Browser**.
   - The fully-qualified reference string appears in the editor option.
   - If you are working in the script editor, the selected reference appears in the script at the current cursor position and replaces text that was selected.

View attribute details in the galaxy browser

When you view attributes in the Galaxy Browser, you see two areas. The objects shown in the left area include all of the logged in user's checked-out objects plus the checked-in versions of all other objects.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Important: The Galaxy Browser shows only the Primary AppEngine and its attributes of a redundant pair. Any Backup AppEngine is not shown. See AppEngine Redundancy in the Application Server help for more information.

The right area shows the attributes of the object selected in the left pane. Depending on the attribute selected, you can see these properties:

<none>	Automatically defaults to the Value property of the selected attribute.
Category	Determines when and where the attribute's data exists (for example, configuration or runtime), which users can write to it, and whether the attribute is lockable or unlockable.
Dimension1 (only for arrays)	Returns the dimension of the attribute if it is an array.
Locked	Determines whether the attribute is currently locked. Valid values are: Unlocked LockedInMe LockedInParent.
Quality	The quality of the attribute as defined in the OPC Draft 3.0 quality definition. OPC quality is stored and transported as a 16-bit value. OPC quality is stored for an attribute as a current quality, and it can be historized and sent to clients.
SecurityClassification	Determines which permissions a user has with respect to the attribute when using an AVEVA application in the runtime environment. Relevant only for attributes that can be written to by users in the runtime environment. If an attribute has no security, this column is blank.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Data Type	The data type of the attribute: • Integer • Boolean • Float • Double • String • Internationalized String • Time • ElapsedTime • ReferenceType • CategorizedStatusType • DataTypeEnum • SecurityClassificationEnum • DataQualityType • CustomEnum • CustomStruct For information about each of these, see the help for the object.
Value	The primary value of the attribute. Sometimes, a list of numbers is included in the **Property** list. Those numbers map to single bits in an integer attribute's Value property. Valid bit field specifiers are: .00 (least significant bit) .01 .02 .03 .04 .05 .06 .07 .08 .09 .10 .11 .12 .13 .14 .15 .16 .17 .18 .19 .20 .21 .22 .23 .24 .25 .26 .27 .28 .29 .30 .31 (most significant bit)

Important: Bit field specifiers are not allowed for integer arrays. Although bit field access is only supported in integers, they appear to be allowed for data types besides integer because they do not cause a warning during configuration. They cause errors in the runtime environment.

Browse for graphics

You use the Galaxy Browser to browse for graphics from:

- The Industrial Graphic Editor, when editing a graphic from the Galaxy, being either a graphic from an object (Template or Instance) or a graphic from the Visualization folder.
- InTouch WindowMaker when editing a managed InTouch application hosted by the Galaxy.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

The graphic currently being edited (or browsed from) does not appear in the list of graphics.

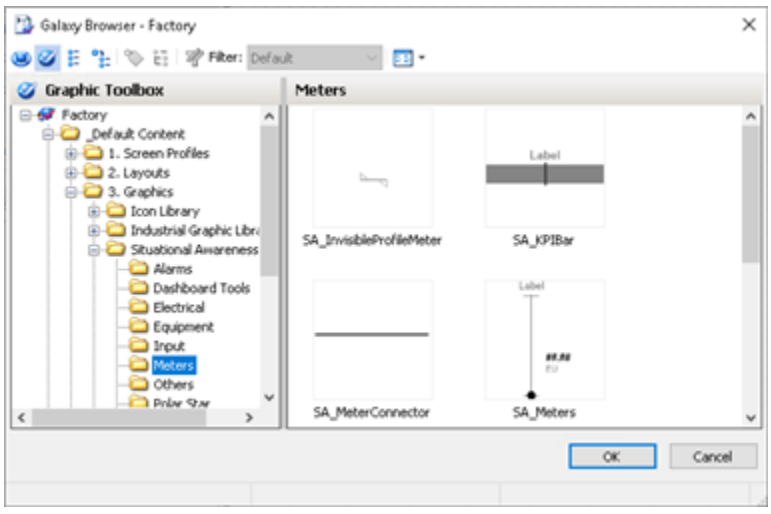
Within the same user session, the Galaxy Browser remembers the last browsed location for graphics and presents it whenever called as the starting location so that context is kept. Initial default location for graphic browsing is the Visualization folder with the root selected (Galaxy node).

You can use the Galaxy Browser to browse graphics from:

	The Visualization folder. The browser shows the Visualization folder organization on the tree in the left pane and a list of the graphics contained in the currently selected node (Galaxy node or folder node) in the right pane.
	AutomationObject templates. The browser shows the Template folder in the left pane and the graphics associated with the currently selected template in the right pane.
	AutomationObject instances. The browser shows a flat list of existing instances of objects in the left pane. The right pane shows the graphics associated with the currently selected instance. You create or apply filters to reduce the scope of the instances shown in the left pane.
	Relative references. This is possible only when you edit a graphic belonging to an object and browse for graphics from that specific object.

To browse for graphics from the graphic editor

1. Click the **Embed Graphic** button in the Graphic Editor. The Galaxy Browser opens, showing the location of the graphics in the left pane and the graphics associated with the current selection in the right pane.



2. Select a graphic from the list and then click **OK**.
3. Click in the canvas to place the graphic.

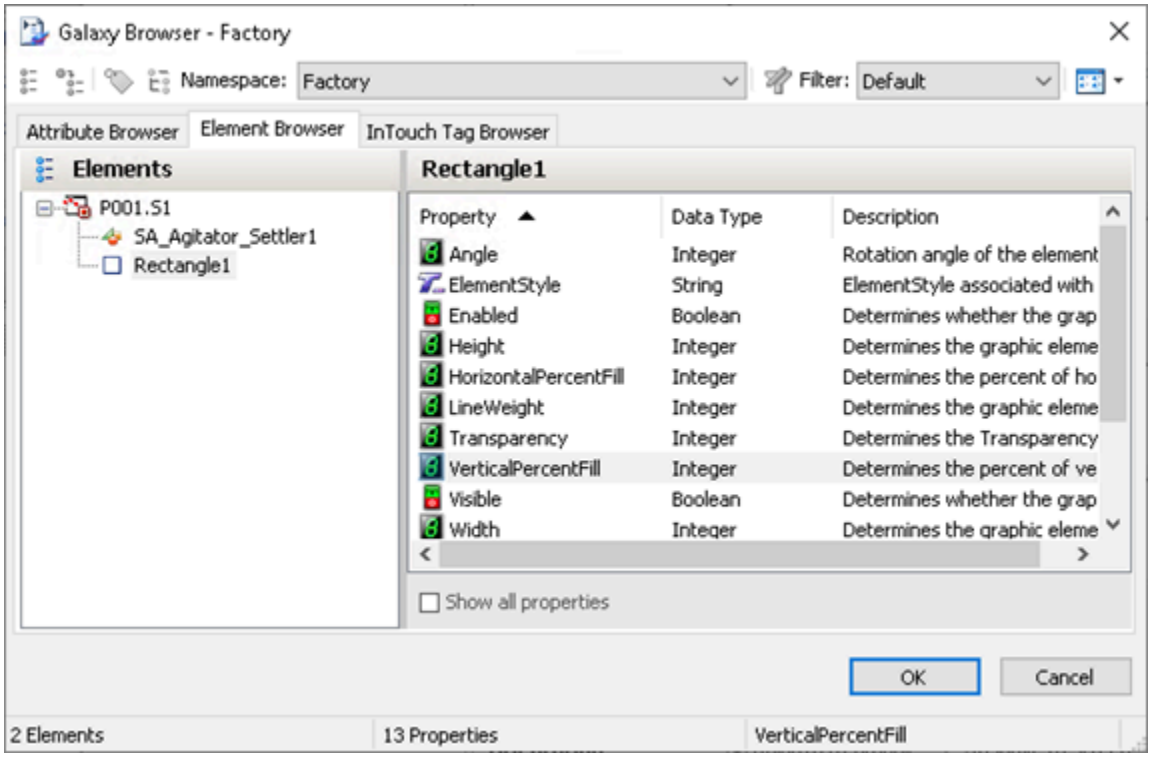
© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Browse for element properties

If you are working on a graphic, you can create references to properties of other graphics. For example, you can reference another graphic's properties from an animation link or script. You can browse the properties of all elements on the canvas or custom properties.

To browse for element properties

1. In any area on a page, click the **Browse** (...) button, if available. The Galaxy Browser opens.
2. Click the **Element Browser** tab.



3. By default, the browser shows only those properties that are frequently accessed. If you are viewing the properties of an element for the first time, the right pane can be blank. Select the **Show all properties** check box to show all of the object's attributes.
4. Select the property and then click **OK**.
   - The fully-qualified reference string appears in the option.
   - If you are working in the script editor, the selected reference appears in the script at the current cursor position and replaces text that was selected.

For more information, see "Working with element styles" in the *Creating and Managing Industrial Graphics User Guide*.

Create a filter for the galaxy browser

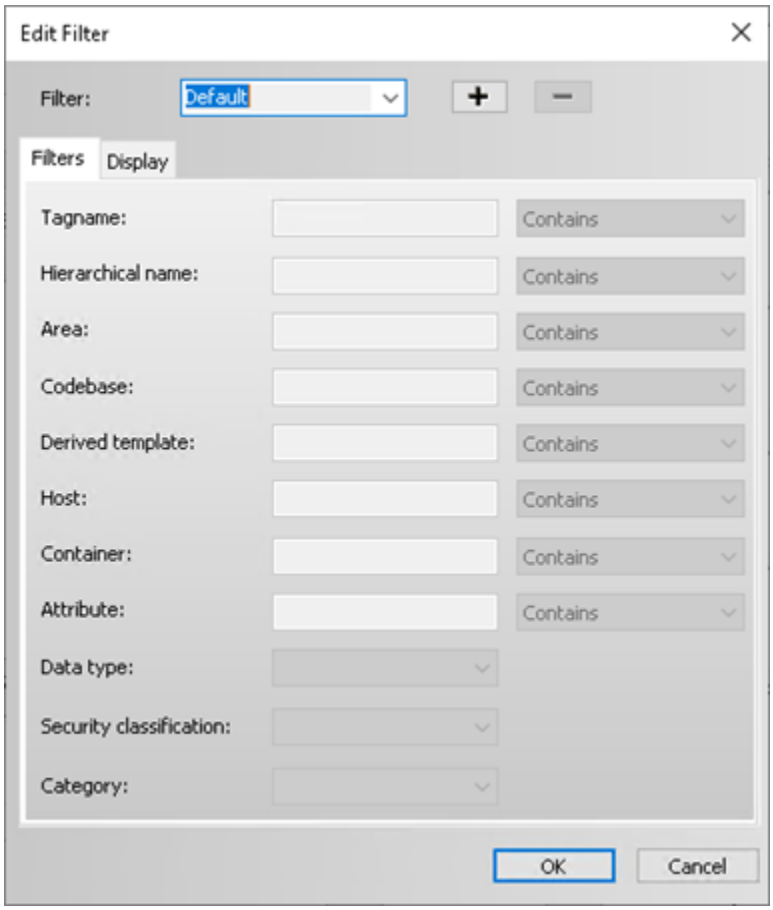
You can create one or more filters that limit the list based on the object name or common attributes. You can also configure the columns you want to show for the list.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

The Default filter provides an unfiltered list of objects and attributes. It cannot be edited.

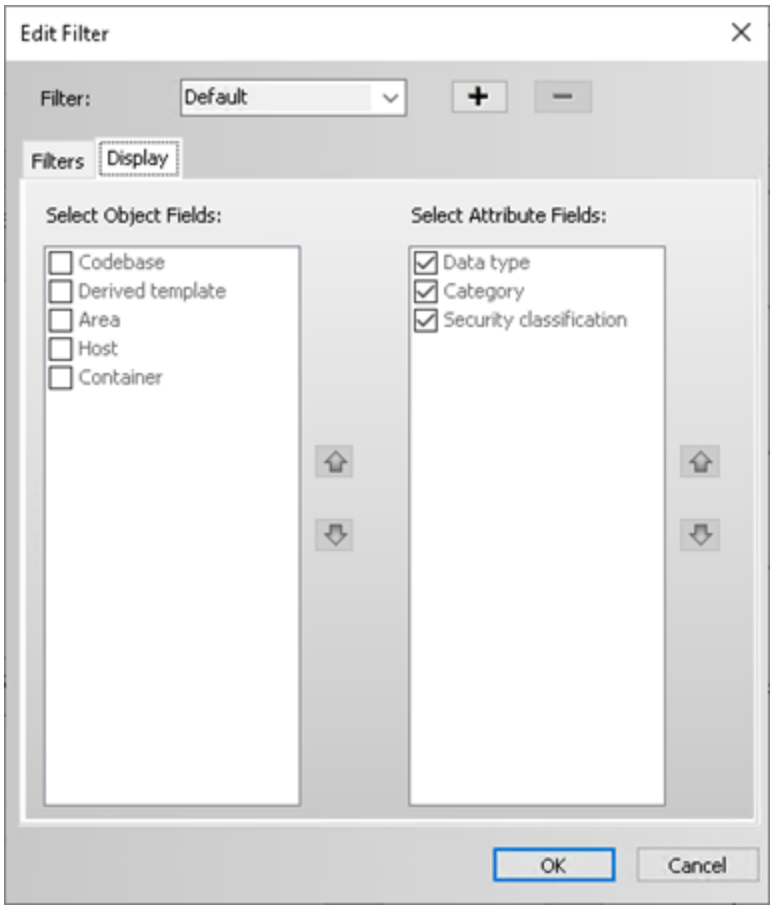
To create a filter

1. Click the **Filter** icon. The **Edit Filter** dialog box appears.



2. Click the **Plus** button and type a name for your new filter.
3. Click the **Filter** tab.
4. Configure the filter details.
5. Click the **Display** tab.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.



6. Configure the columns to show in the right pane of the **Galaxy Browser**. Use the up and down arrows to set the order for the columns.
7. Click **OK**. The new filter appears in the **Filter** list.

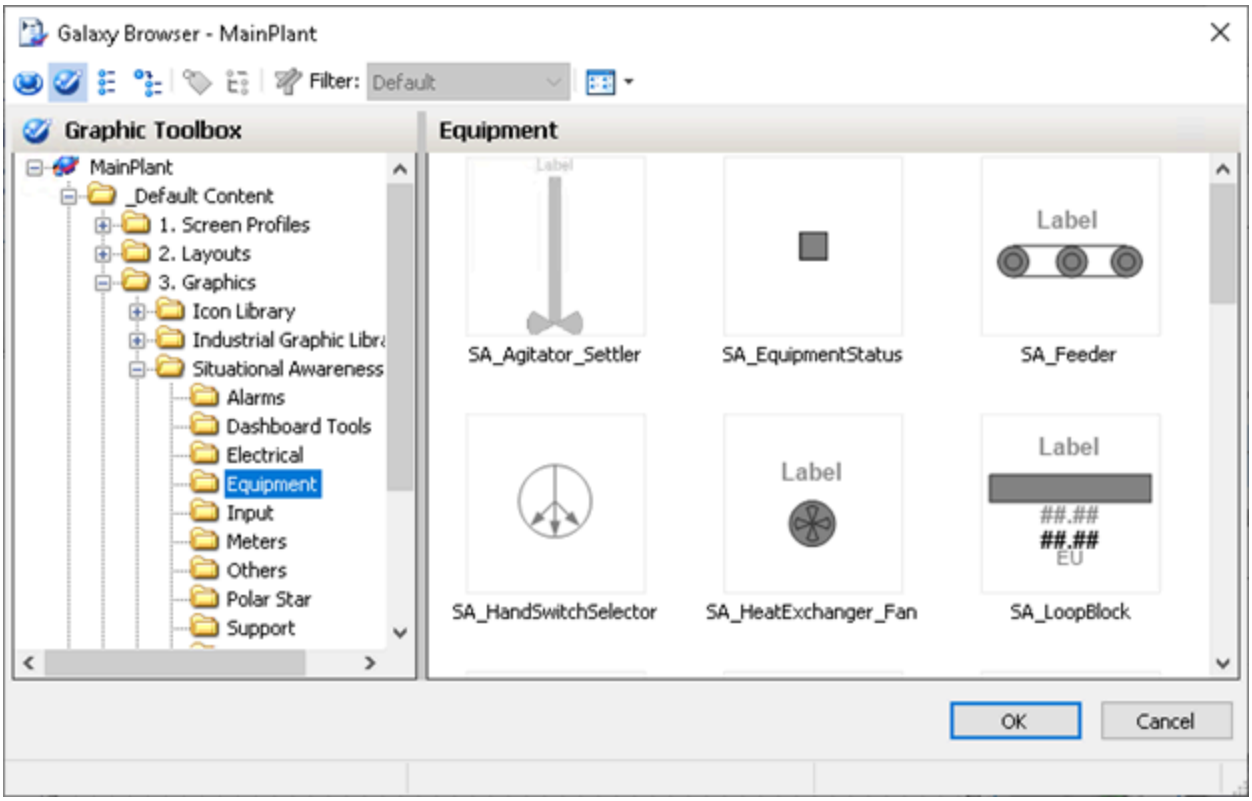
Change how information is shown in the galaxy browser

You can change how information shown in the right pane of the Galaxy Browser. You can view:

- A list of only the attribute or graphic names.
- A list of details for attributes or graphics.
- Named graphic icons.
- Thumbnails for graphics.

The following figure shows graphic thumbnails.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.



Configure security

This section describes System Platform security, and how to use it to manage access to configuration and run-time aspects of your Application Server application.

For the latest information about mitigating security threats that can affect Application Server and Microsoft products, sign into the AVEVA Global Customer Support website:

https://softwaresupport.aveva.com/

Then, navigate to **Security Central**.

For general security guidelines and how you can apply them to Application Server, see Managing Security for Application Server.

About security

Galaxies are created without security. After creating a Galaxy, you can add security to manage access to it. User permissions are set within the IDE. Adding security to a Galaxy authenticates users across all editors and interfaces and controls access to the:

- System Platform IDE
- Operations Control Management Console (OCMC)
- Layout Editor
- Screen Profile Editor

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

- ViewApp Editor

Security lets you set specific permissions for operations within the IDE and OCMC, such as:

- Starting the IDE
- Importing, exporting, creating, and modifying instances, derived templates, graphics and ViewApps
- Deploying and undeploying instances
- Uploading run-time changes
- Managing alarms
- Performing maintenance and system administration functions within the OCMC.

You can also configure credentials that AVEVA OMI apps can use for connecting to resources that do not recognize Windows credentials.

While there are not specific permissions that can be set for the Layout, Screen Profile, and ViewApp Editors, restricting permissions for graphics and objects in the IDE will also restrict a user's ability to modify objects and graphics in these editors. See Assign roles to users for additional information.

Security model for galaxies

Security settings in the IDE control access to:

- User and editor interfaces in the System Platform environment. The signed-in user remains authenticated across all editors and interfaces. The user does not have to sign in again when switching to a different editor, for example, when switching from the IDE to the Screen Profile Editor.

  Note: Regardless of the Galaxy security setting, a user must be a member of the **aaConfigTools** group to connect to a Galaxy from the IDE, or the user must launch the IDE using the "Run as administrator" option. Use the Windows Control Panel to add a user to the aaConfigTools group. Since aaConfigTools is an OS user group, you cannot use Galaxy Security Configuration to add a user to the group.
- Object attributes and the data they represent.
- Connection to the SQL Server database used for the Galaxy Repository.

  Note: For additional information about SQL Server security, see the section on SQL Server Rights Requirements in the *System Platform Installation Guide*.
- Credentials that allow a logged-in user to an AVEVA OMI ViewApp to access third-party resources used by the ViewApp.

Each Galaxy in the Galaxy Repository manages its own security model. The security schema managed in a Galaxy is a three-level configuration model to create and maintain the following:

- Users associated with specific roles
- User roles associated with specific system administration, configuration and run-time (operational) permissions, which map to security groups
- Security groups associated with specific objects in the Galaxy

The default Galaxy Security model includes:

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

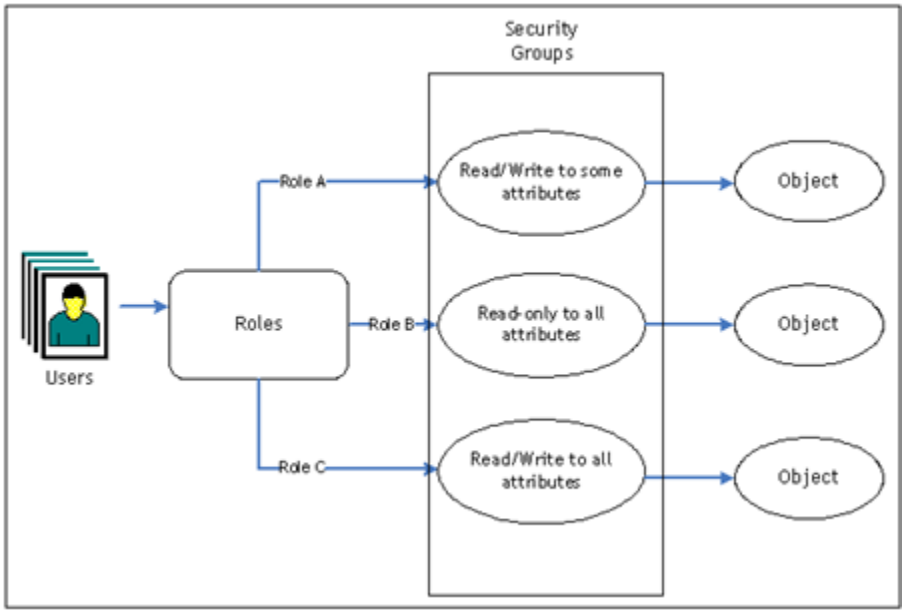
- Two users: DefaultUser and Administrator, both with full access to everything.

  Important: Be sure to change the passwords for the Administrator and DefaultUser accounts after enabling galaxy security. We recommend that you use Microsoft Active Directory to implement your galaxy password policy because it is the easiest and most effective way to set the policy.

- One security group named Default.
- Two security roles: Default and Administrator, both with full privileges.

  Important: We recommend that you set the privileges for the Default role to the minimum level needed by all users.

The security matrix defines a cascading model of users associated with specific roles that are associated with specific security groups that are associated with specific objects. User run-time permissions can vary from object to object, action to action, and process to process. The security icons associated with object attributes map directly to control points in the security model.



Authentication modes

Enable security for a Galaxy by selecting one of four authentication modes:

- **Galaxy**: Uses local Galaxy configuration to authenticate users. All security for the Galaxy is specified and contained at the specific Galaxy level. When the user logs on, security credentials are checked and access to areas and activities is granted at the Galaxy level.

  Note: If you are implementing a multi-Galaxy environment or creating credentials for ViewApps, Galaxy authentication mode is not supported. Use one of the OS-based authentication modes instead. See The multi-galaxy environment and About named credentials for more information.

  You cannot use Galaxy authentication mode if you use the OMI web client. See the OMI help file for more information about the OMI web client.

- **OS User Based**: Uses the operating system's user authentication system on an individual user level. All security for the Galaxy is specified and contained in the operating system (OS) on a user level basis. When the user logs on, security credentials are checked and access to areas and activities are decided at the OS

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

user level.

- **OS Group Based**: Uses the operating system's user authentication system on a group basis. All security for the Galaxy is specified and contained in the user-to-roles mapping you created in the OS to assign security. When a user logs on, security credentials are checked and verified at the OS group level. OS groups are mapped to security roles in the Galaxy to allow access to areas and activities in the Galaxy. For more information, see About OS group-based security.
- **Authentication Providers:** Uses the AVEVA Identity Manager (AIM) to create a unified security management infrastructure across your local System Platform nodes and Azure VMs by leveraging operating system security and using either Azure Active Directory (AD) or AVEVA Connect as the user authentication provider.
  - To use Azure AD as your authentication provider, you must have an Azure AD account available in the Azure portal. See Configure Azure AD as an Identity Provider for guidance about creating an account. The identity manager is normally configured during System Platform installation. See Common Platform Services for details.
  - To use AVEVA Connect your authentication provider, you must have an AVEVA Connect account.

The AVEVA Identity Manager (AIM) is a Platform Common Service (PCS) that creates a unified security management infrastructure across your local System Platform nodes and Azure VMs by leveraging operating system security and an external authentication provider.

Note: If you are using OS user-based security or OS group-based security and you have permissions to use the IDE, the Log In dialog box does not appear.

Important: Do not use the "Administrator" user account to log in to Application Server, InTouch ViewApps, or other System Platform components. "Administrator" is a reserved System Platform name. Some modules of Application Server and System Platform view "Administrator" as a system admin, while other modules view it as a Galaxy admin.

Galaxy migration to support connected experience

When you open an existing galaxy that was configured with the security mode set to Authentication providers (using Azure AD), you must set the security mode to “None” (when in non-connected mode), prior to opening the galaxy in connected experience mode. Connected experience mode is enabled and disabled under **License Mode**, in the System Platform Configurator. See the *System Platform Installation Guide* for details.

Multiple accounts per user

Regardless of the security system, a single user can have multiple accounts. For example, a user can have an account that provides permissions for working with instances but not templates. The same person can have another supervisory account for working with templates and managing users in the System Platform environment.

Each account requires a different user name and password. For example:

User Name	Password	Access
bsmith	password	Instances, not templates
bobsmith	super	Instances, templates, not managing users

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

User Name	Password	Access
Robertsmith	admin	Instances, templates, managing users

Using the local administrator account with single sign on

To ensure that the local administrator account is not locked out when using an external authentication provider for Single Sign-On, a separate login workflow is provided. This allows the user to enter the "administrator" account and without being redirected to the SSO provider. The user is instead validated by Application Server.

To log into Application Server as a local administrator when using an Authentication Provider

1. Enter the admin user name.

   If the user name is not the local administrator, the log in request is forwarded to the external Authentication Provider.
2. Enter the admin password when prompted.

   The administrator is logged in when Application Server authenticates the user.

Changing security settings

After you change security for a Galaxy, you see the following behaviors and conditions:

- When you change the authentication mode security, the IDE restarts.

  Important! Make sure that you have a valid login for the IDE before changing the authentication mode, especially if changing from "None." If not, create a login before changing authentication modes.
- To switch users:
  1. Select **Galaxy** from the menu bar
  2. Select **User** from the backstage options.
  3. Click the **Change User** button.
  4. Enter your credentials (credential of the new user).
- If you previously configured security under one authentication mode and then switch authentication modes, only those users created while configuring the new mode are available. Other users are not deleted, just unavailable in the new mode.
- Objects that are reassigned to different security groups are marked as "pending update" and require redeployment for the change in security group to take effect.
- If security was previously configured for an OS-based authentication mode, reconfiguring security synchronizes the user's full name and OS groups if some data in the OS has changed.

About security groups

Every object in the Galaxy belongs to only one security group. You can create and manage security groups that

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

make sense for your organization. These security groups are mapped to roles on the **Roles** page.

Permissions determine what kind of access users have for each attribute. There are five basic operational permissions:

- Acknowledge alarms
- Change the value of attributes with security mode Configure
- Change the value of attributes with security mode Operate; this includes also security modes Secured Write and Verified Write
- Change the value of attributes with security mode Tune
- Confirm writes to attributes that require Verified Writes

By default, all currently used objects are assigned to a security group called Default.

A user who is a member of a role assigned to Security Role "Default" has permission to:

- Acknowledge alarms
- Change attribute values with "configure" security mode
- Change attribute values with "operate" security mode, including "secured write" and "verified write"
- Change attribute values with "tune" security mode
- Verify writes to Attributes with "verified write" security mode

For example, you want users in certain roles to only have permission to acknowledge alarms that are generated from objects contained in Area1. You have a role named Area1Acknowledgers. You need to:

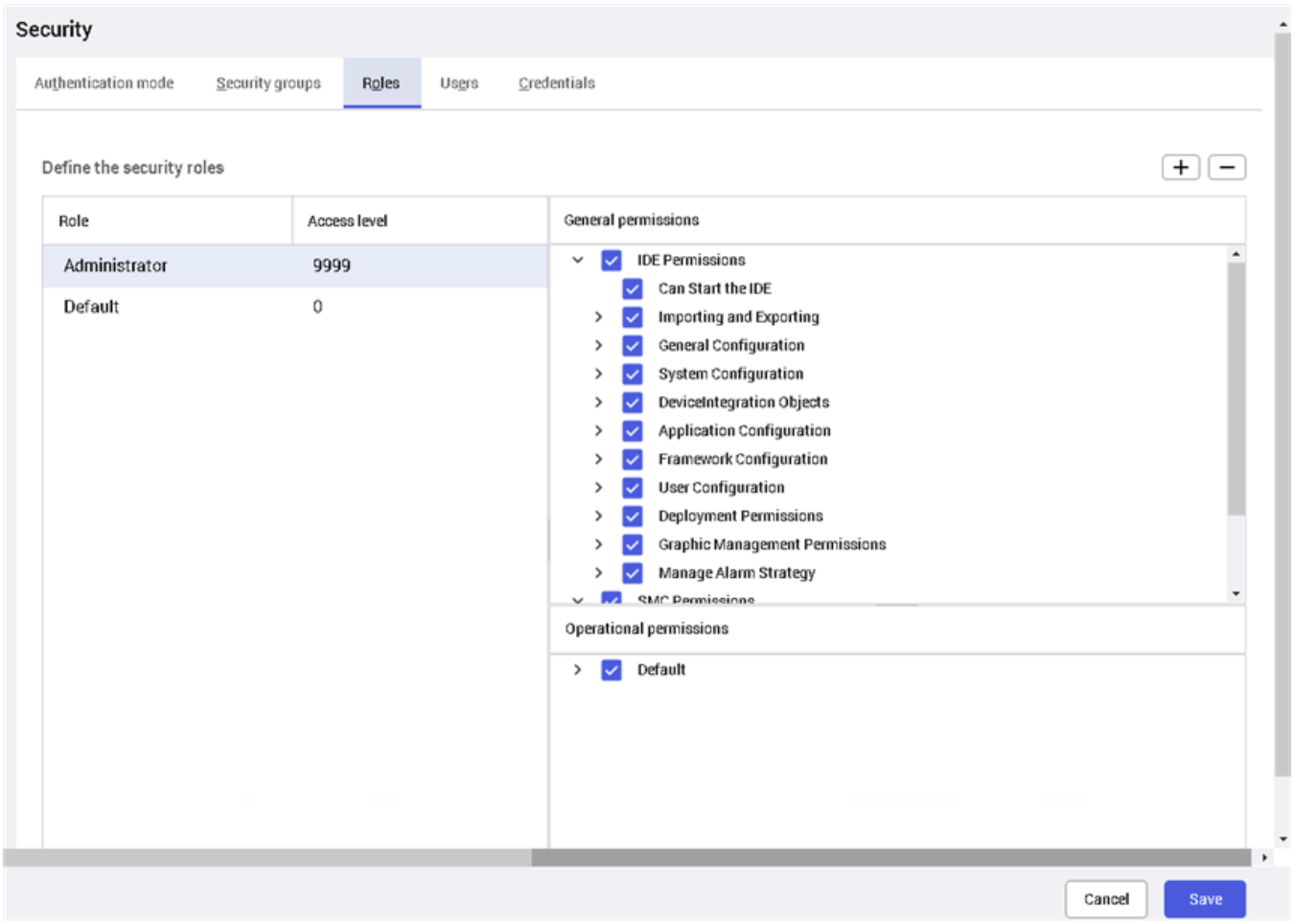
1. Create a new Security Group, for example **SecGrpArea001**.
2. Assign all objects that are contained in area Area1 to Security Role **SecRoleArea001**.
3. On the **Roles** page, select the **Area1Acknowledgers** role. In the **Operational Permissions the Security Group** for **SecGrpArea001**, select **Can Acknowledge Alarms**.
4. Any user that belongs to the **Area1Acknowledgers** role can at least acknowledge alarms of objects contained in the security group **SecGrpArea001**. They do not have any other operational permissions for those objects.

About roles

You can create and manage user roles that apply to your organization’s processes and work-based authorities. A role defines a set of permissions. After defining roles, you assign roles to authorized users as needed to provide the users with the permissions that they need. A user can be given multiple roles.

Application Server automatically creates two roles called **Administrator** and **Default** and gives both roles full permissions to everything. When defining roles, first remove any permissions that are not needed by all users from the **Default** role because all configured users will be assigned to the Default role and you cannot remove the Default role from a user. For example, if the minimum set of permissions required by all users is the ability to acknowledge alarms and modify "Operate" attributes, remove all other permissions. Add new roles by selecting the **+** button above **Roles available**, then enter a name for the role and an access level.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

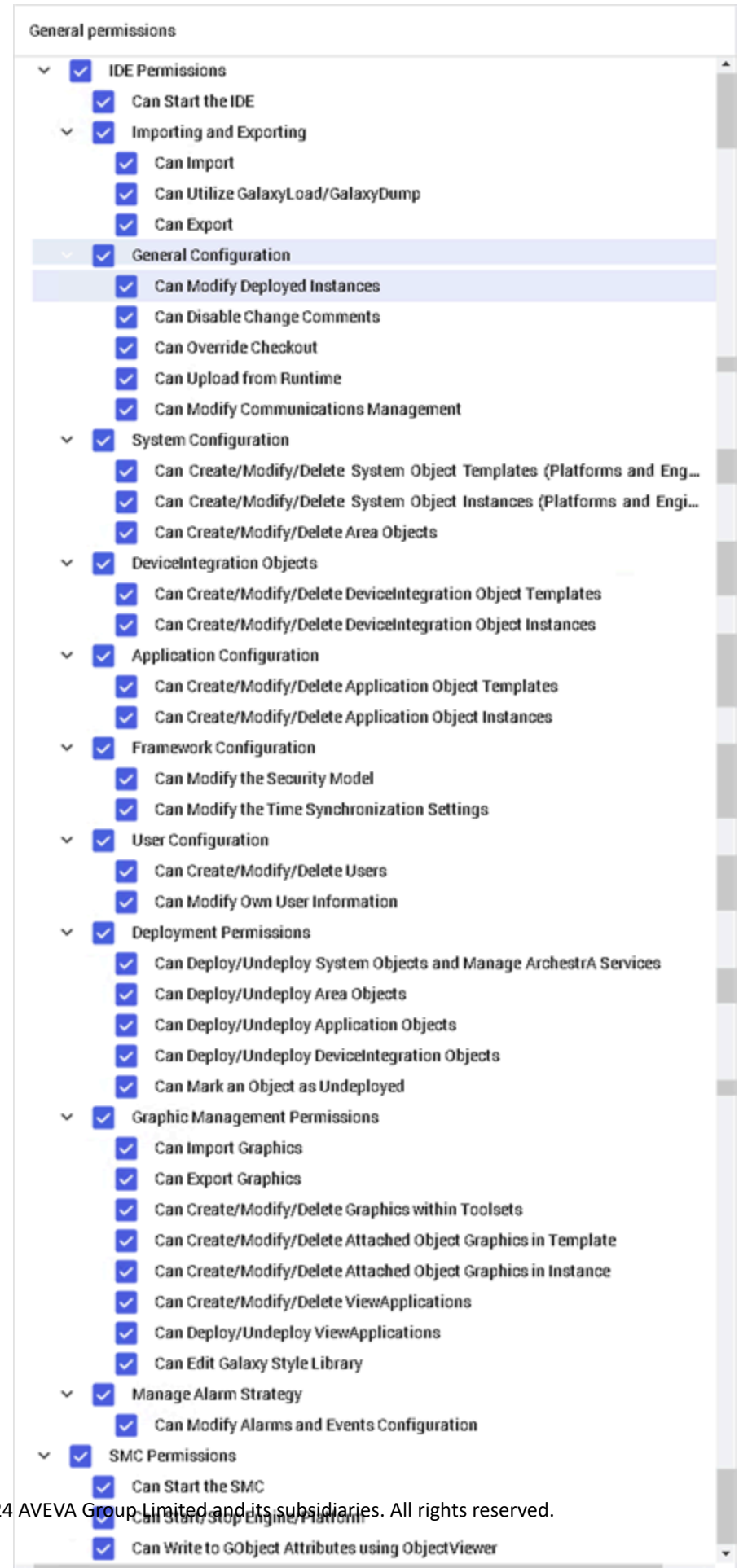


You can specify General and Operational Permissions for each role.

- **General permissions** relate to application configuration and administration tasks within the IDE and the Operations Control Management Console (OCMC). By default, the Administrator and Default roles have all general permissions. Remove any unnecessary general permissions from the Default role. Leave only the permissions that are needed by all users.

  Note: You cannot remove any General permissions from the Administrator role. You can, however, remove Operational permissions.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.



© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

- **Operational permissions** relate to the security groups listed on the **Security Groups** page. For each role, modify security group permissions as needed. The Administrator and Default roles initially are given all operational permissions for the Default Security Group. If you created any new security groups on the Security Groups page, the Administrator and Default roles are not automatically granted any permissions to these security groups. If you do not intend to have users with the Administrator role working with alarms or attributes at run time, you can limit operational permissions from the Administrator role.

  Operational permissions are defined as follows:

  - **Can Acknowledge Alarms**: Allows users to manually acknowledge an alarm in the run-time environment.
  - **Can Shelve Alarms**: Allows users to manually shelve and unshelve alarms.
  - **Can Modify Alarm Modes**: Allows users to modify the mode of an alarm.
  - **Can Modify Plant States**: Allows users to modify plant states for state-based alarming.
  - **Can Verify Writes**: Allows users to provide an authentication signature for attributes configured with Verified Writes security classification. Only users with this permission can verify a task performed by users with the Can Modify "Operate" Attributes permission.
  - **Can Modify "Operate" Attributes**: Allows users with operational permissions to do certain normal day-to-day tasks like changing setpoint, output and control mode for a PID object, or commanding a Discrete Device object.
  - **Can Modify "Tune" Attributes**: Allows users to tune the attribute in the run-time environment. Examples of tuning are attributes that adjust alarm setpoints and PID sensitivity.
  - **Can Modify "Configure" Attributes**: Allows users to configure the attribute's value. Requires that the user first put the object Off scan. Writing to these attributes is considered a significant configuration change, for example, a PLC register that defines a Discrete Device input.

Once you have defined a role by setting operational permissions and general permissions, you can assign users to that role.

About users

If you select either OS based authentication mode, users with local accounts are added to the **Authorized Users Available** list in the following format: `.\<username>`. Domain users are added in the format `<domain>\username>.`

If you select **OS Group Based** authentication mode and use local accounts, the local account must exist on each node in the Galaxy for successful authentication of that user on any computer in the Galaxy. OS groups are added in the format `<domain>\<groupname>.`

Two users are defined by default when a new Galaxy is created: Administrator and DefaultUser. These cannot be deleted in an open security setting and both users are associated with the default roles, Administrator and Default. See About roles for more information.

Important: Do not use the "Administrator" user account to log in to Application Server, InTouch ViewApps, or other System Platform components. "Administrator" is a reserved System Platform name. Some modules of Application Server and System Platform view "Administrator" as a system admin, while other modules view it as a Galaxy admin.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

About named credentials

The **Credentials** tab lets you add login credentials that some AVEVA OMI ViewApps may need for access to third-party data and applications that do not support standard authentication methods, such as Windows OS credentials, Active Directory, OpenID Connect, or other standard authentication method.

Initially, a galaxy does not include any credentials. Credentials are created on a galaxy-wide basis, not for individual ViewApps. Although you enter credentials as plain text, the credentials are encrypted when saved, and the credentials are sent to run-time nodes as encrypted data.

Each credential you create is associated with one and only one OS user group. In order to create credentials, OS group-based or OS user-based security must be configured. The credentials you create here are associated with an OS group name. When a user logs into a ViewApp that has an app with a configured credential, the logged-in user obtains access to the credential, provided that the user is associated with the same OS group that matches the OS group for the credential. For example, if you create a credential that is associated with the OS group "Operators," a user that is also a member of "Operators" has access to that credential.

App developers can use the AVEVA OMI SDK to include a dropdown menu that lists configured credentials. When you add an app that uses credentials to a layout, you can select one of the credentials as you configure the app. This is the credential that the app will use at run time, when deployed in a ViewApp. Only one credential can be configured per layout, and users cannot select a different credential from the running ViewApp.

Note: Credentials are supported only when **OS User based** or **OS Group based** authentication mode is enabled. If authentication mode is set to **None** or **Galaxy**, you cannot add or delete credentials, even if the credentials were created while the authentication mode was set to OS User or OS Group.

Create a new credential

From the **Configure Security** page, select the **Credentials** tab. The **Credentials** tab is divided in three columns:

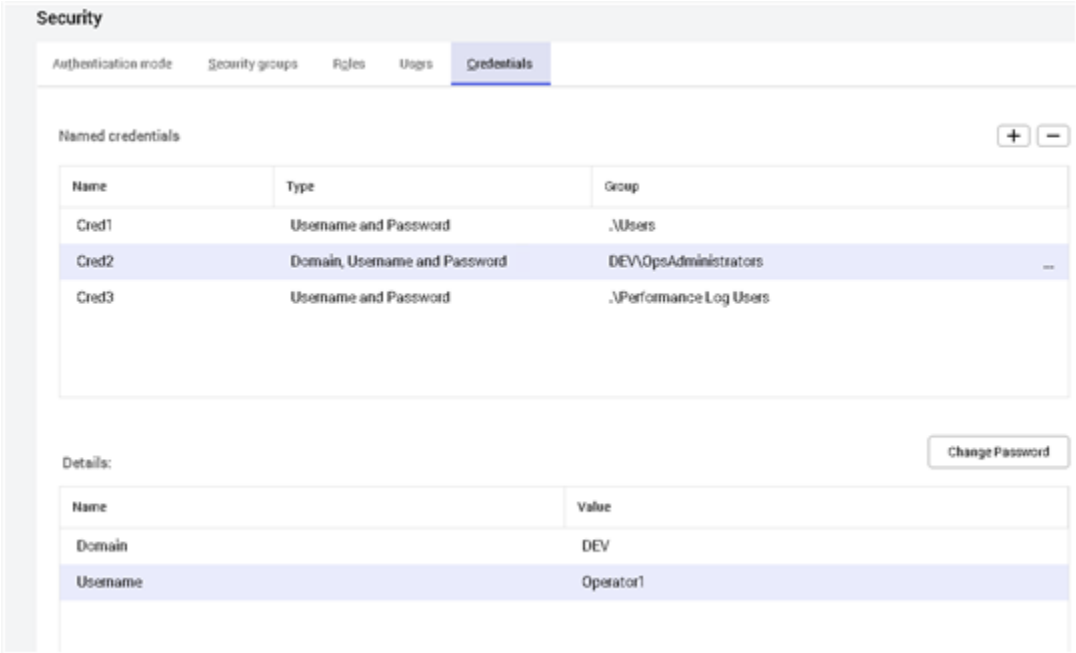
- Name
- Type
- Group

To create a new credential

1. Click the **Add Credential** button. A new credential is added to the **Available Credentials** window.
2. Rename the credential by selecting the default name from the **Name** column and then type the new name.
3. Set the credential **Type** by selecting the down arrow at the right edge of the **Type** column. Type can be one of the following:
   - Username and Password
   - Domain, Username and Password
4. To select the **Group** to associate with the credential:
   a. Click the ellipses button (...) at the right of **Group** column. The **Select Groups** dialog opens. See Associate an OS group with a named credential for more information.
   b. Choose the applicable OS group name from the list of available OS groups, or type the name of an existing OS group.
5. Configure the credential after you have created a credential by entering values for Domain, Username, and

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Password. See Configure credentials for additional information.



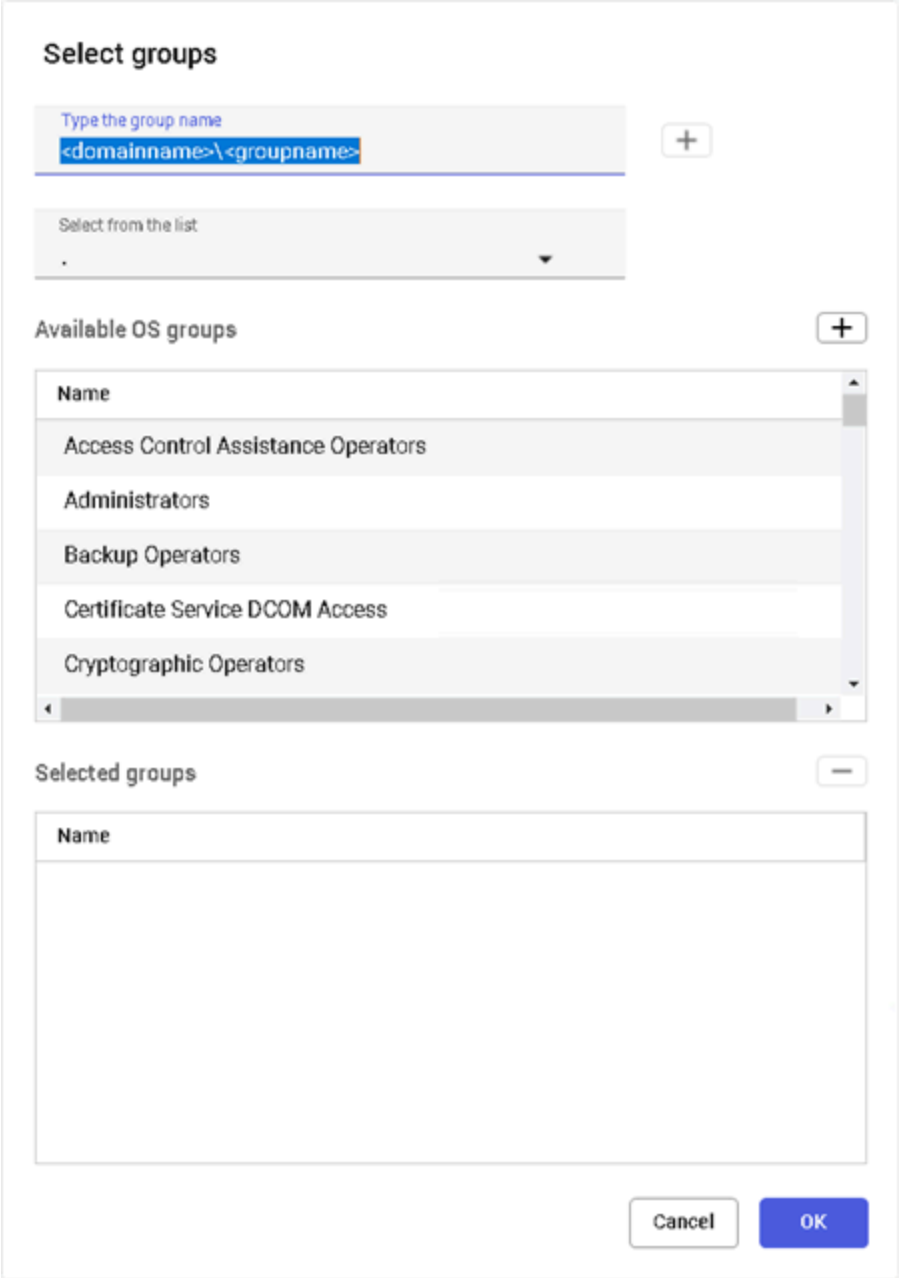
Associate an OS group with a named credential

To associate an OS group with a credential

1. Select the the ellipses button (...) next to the group name in the **Credentials** tab.

   The **Select groups** dialog opens with the local machine selected, and the list of OS groups that exist on the local machine will populate the list of **Available OS groups**.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.



2. Choose an existing OS group from the **Available OS groups** list. Or,type the domain and OS group name in the **Type the group name** text box.
3. Add the OS group by clicking the **+** button above the list, on the right side. The OS group is added in the **Selected groups** window.
   - If you are creating a credential that includes domain, select the applicable domain from the **Select from the list** dropdown list, then select the OS group name from the **Available OS groups** window. If the credential type you created does not include domain, you will not be able to select another domain.
   - If you manually enter the domain and/or the OS group name, use a backslash to separate the domain from the group name.
   - If the OS group is on the local machine, you must include ".\" before the group name.

Note: Only one OS group can be added per credential.

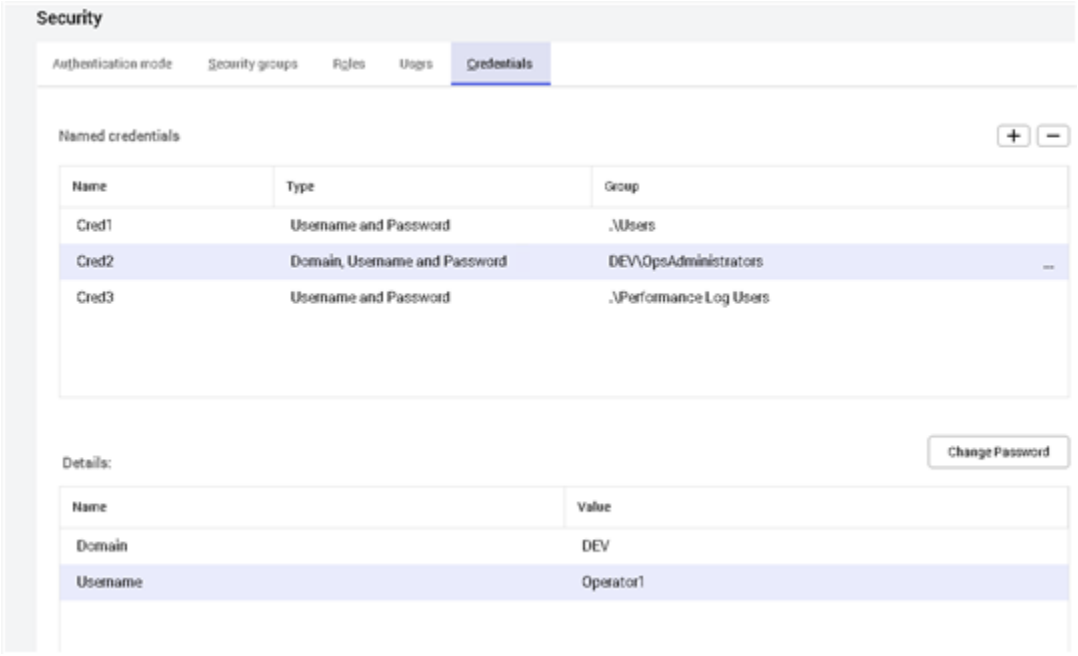
© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

4. Click **OK** to close the **Select groups** dialog and return the **Credentials** tab.

Configure credentials

To configure a credential

1. Select the credential to be configured in the **Named credentials** lsit.



2. In the **Details** window, enter the **Username** for the credential. If the credential type is Domain, Username, and Password, also enter the **Domain** name.
3. To enter the password for the user, select **Change password**.
4. On the **Change password** dialog box, enter and confirm the password for the user name you entered. Select **Save**.

   You can enter null values for the username and password. If a domain value is required, enter the domain name that was used to create the credential.
5. Select **Save** to save the credentials.

Credential definitions

Name: This is the name of the credential. The default credential name is "Credential," and subsequent names are appended by a number that automatically is incremented by one. To rename a credential, just highlight the name and enter new one. Credential names must contain at least one character and must be unique.

Configured credentials are typically shown in a dropdown menu in a ViewApp that has been configured to use credentials. The user can then select the applicable credential. In some cases, an app developer may choose to hard-code a credential name. In this case, no user action is required to access the credential at runtime.

Type: Two types of credentials are supported:

- **Username and Password.**
- **Domain, Username and Password.**

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Note: Null values are valid for all fields, except the credential name.

Group: When you add credentials, you must associate each credential with an OS Group name (local or domain) that has been defined previously.

To associate a credential with an OS group, simply select the group name when you add the credential. You can change the group at any time in the **Credentials** tab, but doing so may impact AVEVA OMI apps that have been configure to use the credential. Associations are saved when you click **OK**. See Associate an OS group with a named credential for more information.

About SQL Server security

SQL Server has its own security settings. When you install Application Server, the following operating system groups and users and user accounts are created:

- aaAdministrators group
- Network Account (for inter-node communication)
- aaGalaxyOwner user account

Caution: aaGalaxyOwner and ASBService are reserved OS user names. aaAdministrators and ASBSolution are reserved OS group names. Do not create users or groups with these names.

The aaAdministrators group, Network Account, and aaGalaxyOwner user account are defined for use with SQL Server, and are used for Galaxy operations. Do not delete this group or these accounts. See the section on SQL Server Rights Requirements in the *System Platform Installation Guide* for additional information. This section also describes how to set the SQL Server Security Mode with the **aaConfig SQL** utility.

Configure roles and groups

Before you open the security editor for a Galaxy, make sure:

- No other user is connected to the Galaxy.
- All objects in the Galaxy are checked in.
- Your user profile has configuration permissions to change Framework Configuration/Modify Security Model, if security was previously configured.

If you try to open the security editor before these conditions are met, a warning message appears and you are denied access.

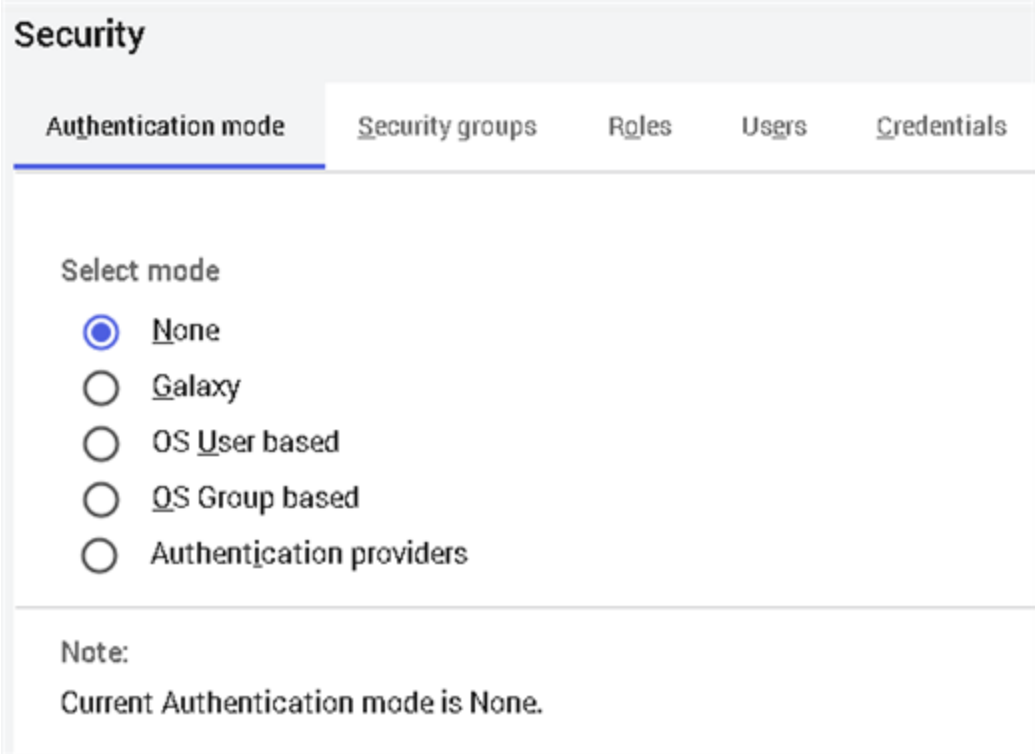
Caution: Do not configure security settings of the IDE while an IDE-managed InTouch application is opened for editing in WindowMaker.

Other users who try to open the Galaxy while you are configuring security are denied access to the Galaxy.

To configure Galaxy security

1. From the ribbon, select **Galaxy**, then **Configure,** and then **Security**. The **Security** page appears.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

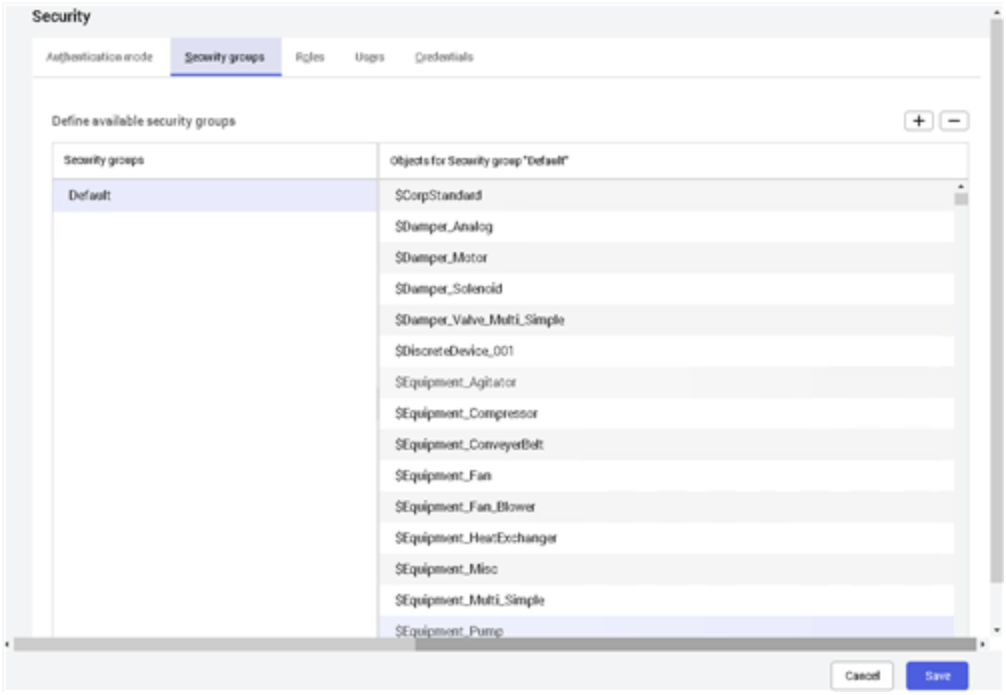


2. On the **Authentication mode** tab, select the security type you want: Galaxy, OS User Based, OS Group Based, or Authentication providers (AVEVA Identity Manager. To use the AVEVA Identity Manager, see Configure the AVEVA Identity Manager.

   If you select the **OS Group-based** authentication mode, you can specify the time-out period for login attempts and role verification. Allowing a longer interval can be helpful if your network connection is slow or intermittent.

   - **Login time**: The time-out period (measured in milliseconds) during which the system validates the user's membership against the OS groups selected as System Platform Roles. Minimum value is 0 (zero), maximum is 9,999,999. The default value is 1,000. If the login time is set to 0 (zero), which turns this feature off, the operation does not time out. Specify a value, based on the speed of your network and the number of groups configured in System Platform. The slower the network or the larger the number of groups, the greater the value.
   - **Role update**: The time between each validation attempt per OS group for the user's membership when a log on is attempted. The user membership update is done one role per **Role update** interval to minimize network usage. The minimum allowed value is 0 (zero) and the maximum is 9,999,999. The default value is 0 (zero), which turns off this feature so the operation does not pause between validating user membership and groups. This option operates independently of the **Login Time** option. Even if **Login Time** times out, the role update operation continues in the background and eventually updates user-to-role relationships for this user in the local cache.
3. Select the **Security Groups** tab to add and configure new security groups. Security groups define which objects (templates and instances) the logged-in user can access. To assign objects to a new Security Group, select the objects listed under the Default Group and drag them to the new group. Security Groups are displayed in Roles tab, under "Operational Permissions." See Assign roles to users for more information.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.



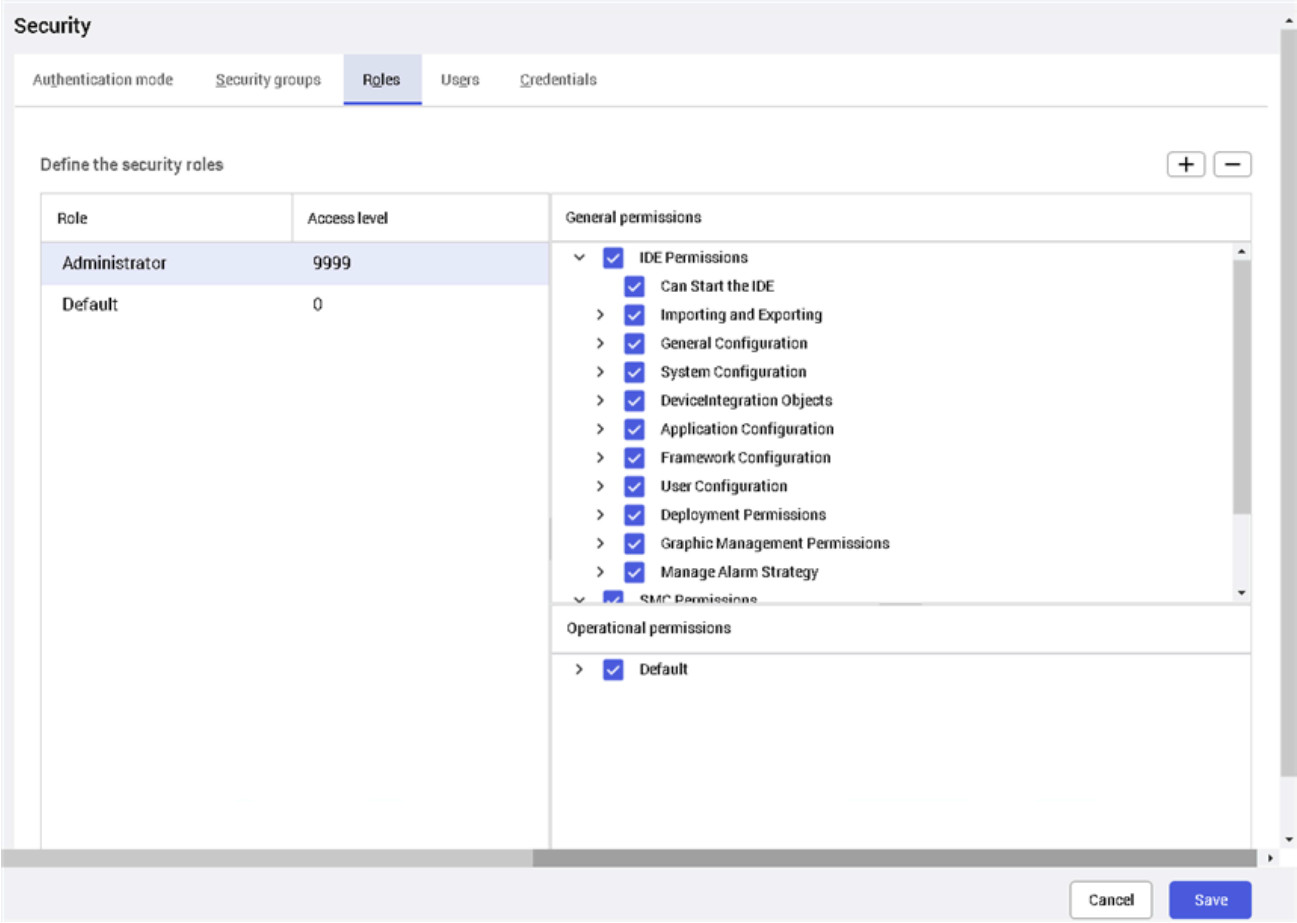
- Create a new security group by clicking the **Add** button. Type a unique name for the new group in the **Security Groups Available** pane. Security group names can be up to 32 alphanumeric characters, including a period. The name must include at least one letter and cannot start with $.

  Note: Security group names are not case sensitive. `Admin` is the same as `admin`.

- Objects can only be associated with one security group. To move objects from the Default security group to the new security group, click the **Default** group, then drag objects to the new security group.

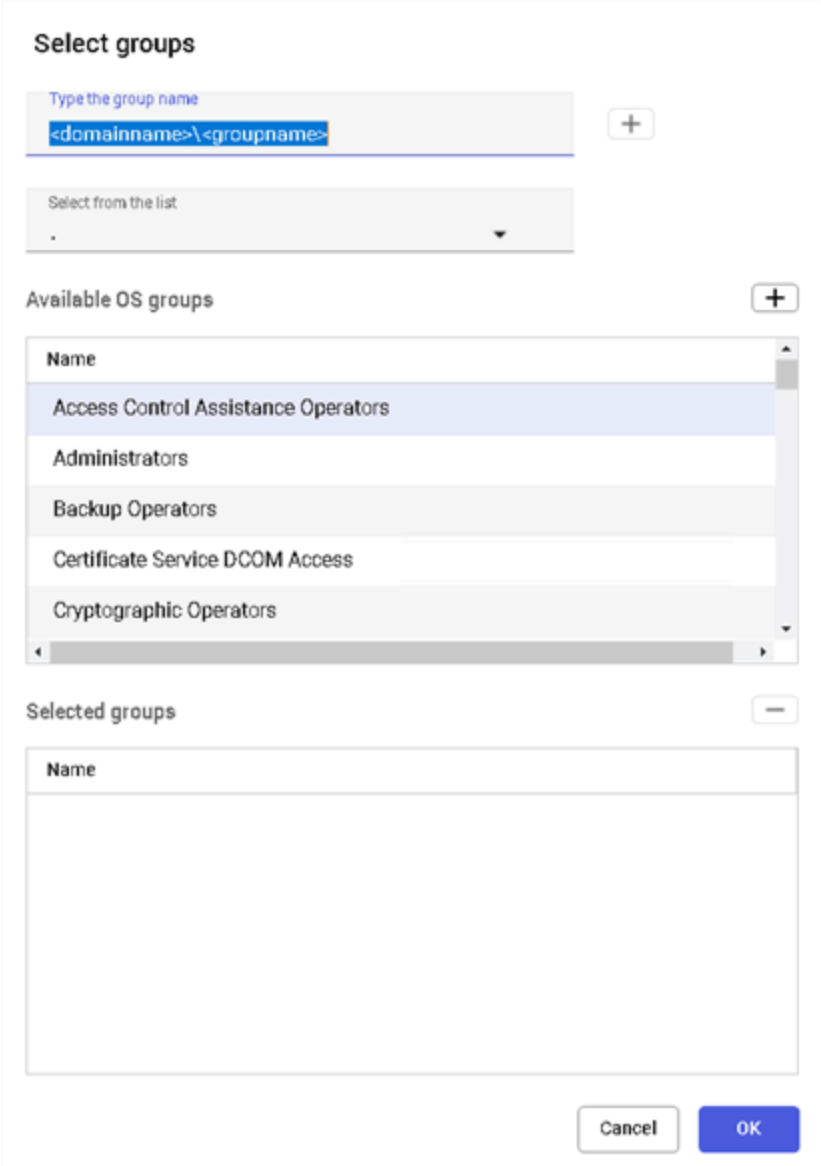
4. Select the **Roles** tab to set the permissions (general and operational) for each Security Role, and to create new roles or delete existing roles.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.



- Remove any unnecessary general and operational permissions from the **Default** role (you can also remove operational permissions from the Administrator role). Leave only the permissions that are needed by all users. See About roles for more information.
- If you selected **Galaxy** or **OS User based** authentication mode, you can create new roles by clicking the **Add** button. Type a name for the new role in the **Roles Available** pane. Role names can be up to 512 alphanumeric characters, including a period.
- If you selected the **OS Group based** authentication mode, the **Select Groups** dialog is displayed after you click the **Add** button. This lets you add roles, based on existing OS groups.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.



- In **Enter the OS Group name**, type the group name (preceded by the domain name, if not part of the local domain) and click the **Add** button, or,
- In **Select in**, select the domain that has the OS group you want to add as a role. Then, select the group from the "Available OS Groups" dropdown and click the **Add** button.

- Select the **General** and **Operational Permissions** for the new role. General permissions define Galaxy configuration and management actions that a user is allowed in the IDE and the OCMC. Operation permissions define the actions a user can take at run time, for example, permission to acknowledge alarms or modify attribute values.

  Important: In the **General permissions** area, clearing the **Can Start OCMC** check box will still allow a user assigned to this role to start the OCMC, but not to connect to Platform Manager.

  Important: In the **General permissions** area, clearing the **Can Start/Stop the Engine/Platform** check box will still allow the user assigned to this role to set the engine or platform On Scan or Off Scan.

  Important: If a role is given "Can Modify Deployed Instances" permission, make sure "Can Create/Modify/Delete..." permissions in the System Configuration, Device Integration Objects, and Application

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

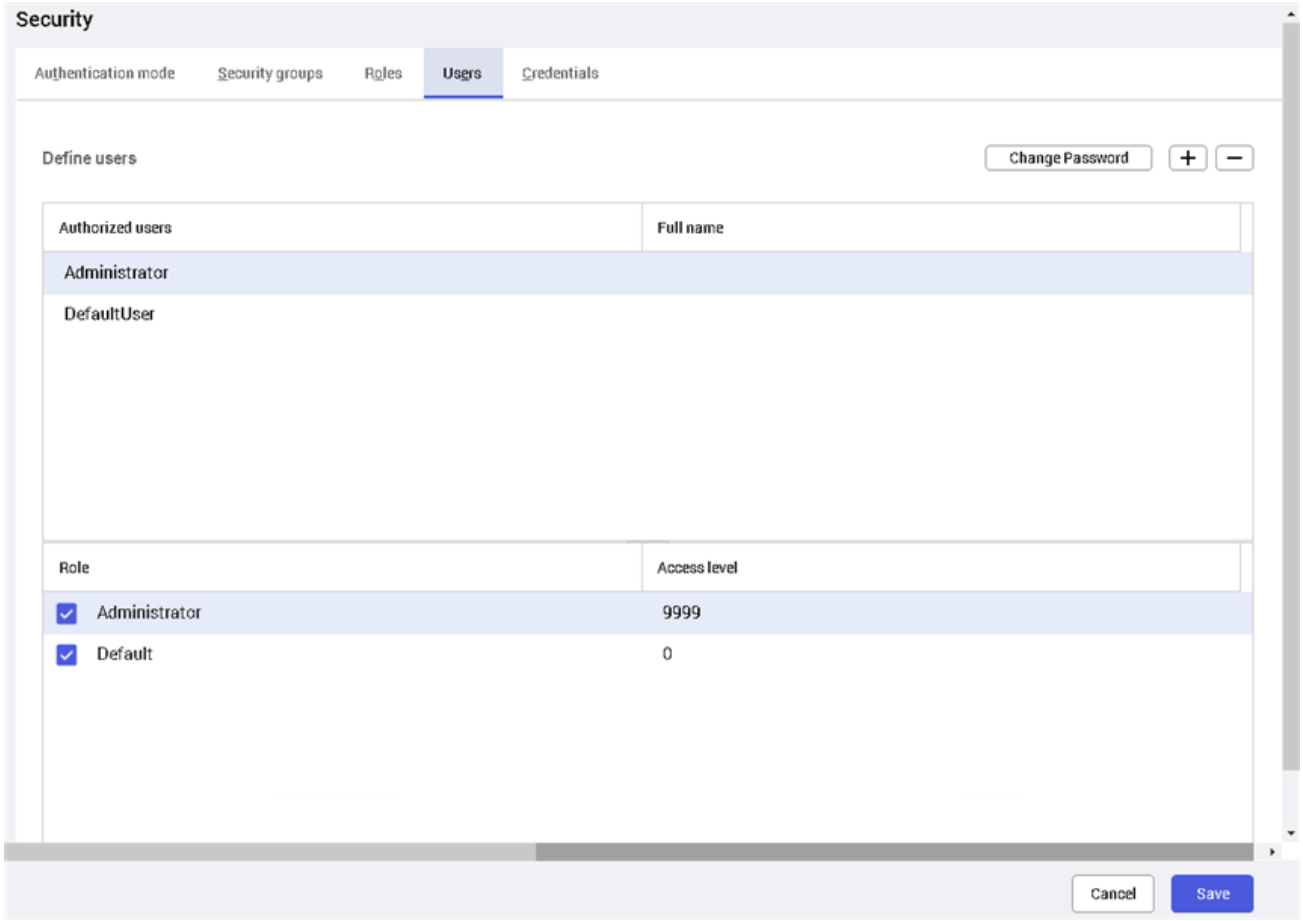
Configuration groups are also selected. This provides the role with check in and undo checkout abilities.

5. Select the **Users** tab to associate authorized users with their roles, add new users, or delete existing users.

Note: Bold red text indicates that the User or Role is invalid within the selected Authentication Mode.

If you selected **Galaxy** or **OS User based** authentication, create a new user by clicking the **Add** button. User names can be up to 255 alphanumeric characters with no spaces. To configure user roles, see Assign roles to users .

Note: You cannot add users if OS Groups based authentication mode is selected.



While viewing Application Server events and alarms in InTouch, the "." appears as the user's domain if it is a local name. Otherwise, it appears as `<domain name>\<username>`.

6. When you are done, click **OK**. You are prompted to log on to the currently open Galaxy.

Important: To improve security, do not use the default user accounts (Administrator or DefaultUser) when logging in to the IDE or other System Platform components. These are reserved system names. In general, you should avoid using generic or easily-guessed user names such as "Admin," "Administrator," "DefaultUser," or "User." Be sure to change the passwords for these accounts to prevent malicious or accidental access to the Galaxy.

Assign roles to users

After you create users and roles, you can assign users to roles. On the **Users** page, all users in the Galaxy and the roles they are assigned are listed.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

New users are automatically associated with the Default role (every user must belongs to the Default role). To assign additional roles to a user, select the checkbox for the role while the user name is highlighted.

Note: All users are automatically assigned to the Default role in addition to any new roles you create and assign to them. The best way to manage permissions is to limit the permissions of the Default role to those permissions you want everyone to have. Then, add users to other roles with permissions for other parts of Application Server.

The Administrator user can log on any authentication mode except when security is disabled. When you are logged in as Administrator on the Galaxy Repository node, you can change the password of any Galaxy user without providing the old password.

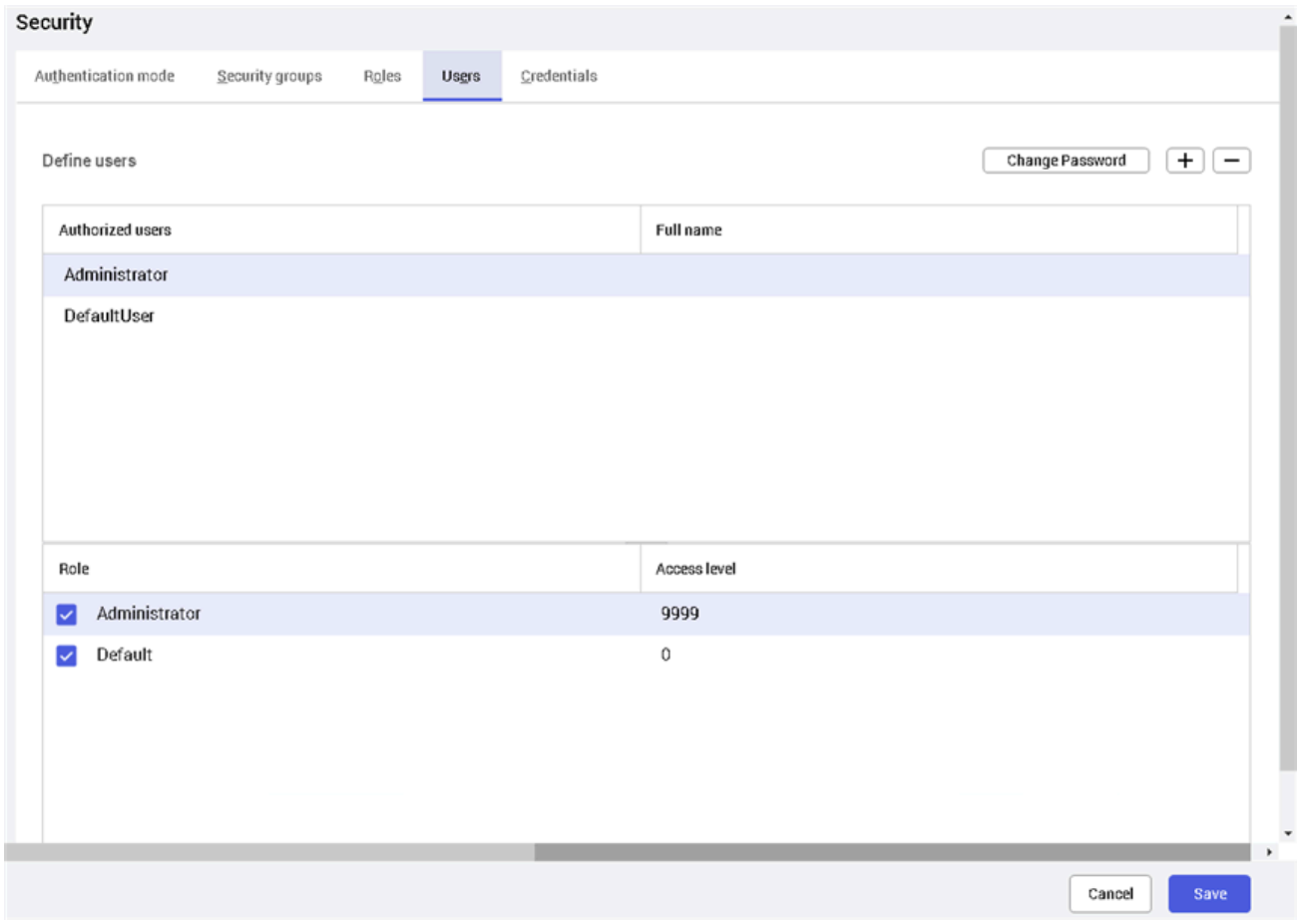
- In Galaxy authentication mode, you can edit the **User Name** in the **Change Password** dialog box.
- In OS-based authentication modes, the **User Name** of the OS user is shown. User information cannot be edited.

You can assign users to more than one role.

To assign a role to a user

1. From the ribbon, select **Galaxy**, select **Configure,** and then select **Security**. The **Security** page appears.
2. Click the **Users** tab.

   Note: The **Change Password** button is displayed only if **Galaxy authentication mode** is selected.

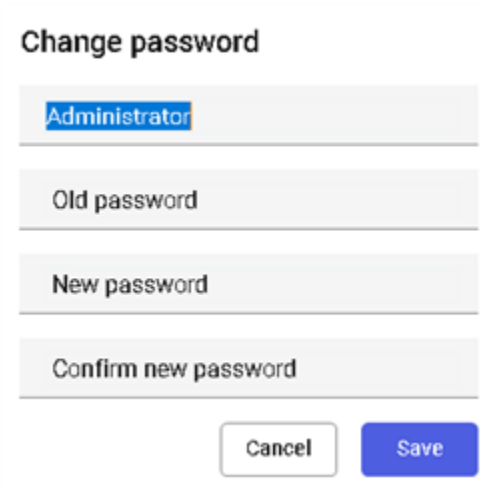


3. Select the user in the **Authorized Users available** area. Select a role in the **Associated Roles for <user name>** area.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Every user is assigned to the Default role. Add other roles that they need to perform their job functions.

4. Provide each user with a password. Make sure that you set new passwords for the **Administrator** and **DefaultUser** accounts.
   - If you are using Galaxy Authentication Mode, click **Change Password**. The **Change Password** dialog box appears.



     You can leave the **Old Password** field blank when you initially set the password for the Administrator and DefaultUser accounts.
   - If you are using an OS-based security mode, use the Windows Control Panel to manage passwords.

Important: If an OS-based security mode is selected on the **Authentication Mode** page, changing a user's password changes the OS password for the user. Any System Platform password may be set to a maximum of 127 characters.

5. When you have finished configuring users, click **OK**. The security information is used in the configuration, administration and run-time environment to authenticate users.

Delete security groups

You can delete a security group you no longer need. Before you can delete a security group, make sure no objects are associated with it.

You cannot delete the Default security group.

Note: The aaAdministrators group is required. If you accidentally delete it from Windows, you can run either the **Change Network Account** utility or the **aaConfig SQL** utility to restore it. If you delete the group from SQL Server, you must run **aaConfig SQL**. See the *System Platform Installation Guide* for more information.

To delete a security group

1. From the ribbon, select **Galaxy**, select **Configure,** and then select **Security**. The **Security** page appears.
2. Select the **Security groups** tab.
3. Select the group you want to delete.
4. Select **Delete** (**-**).

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Delete roles

You can delete roles you no longer need. You cannot delete a role if any users are associated with it.

You cannot delete the Default and Administrator roles.

To delete a role

1. From the ribbon, select **Galaxy**, select **Configure,** and then select **Security**. The **Security** page appears.
2. Select the **Roles** tab.
3. Select the role you want to delete.
4. Select **Delete** (**-**).

Delete users

You can delete users you no longer need.

Note: You cannot delete a user who is currently logged on.

To delete a user

1. From the ribbon, select **Galaxy**, select **Configure,** and then select **Security**. The **Security** page appears.
2. Select the **Users** tab.
3. Select the user to delete.
4. Select **Delete** (**-**).

   Note: The Network Account and aaGalaxyOwner account are required. If you accidentally delete the Network Account from Windows, you can run either the **Change Network Account** utility or the **aaConfig SQL** utility to restore it. If you delete either account from SQL Server, you must run **aaConfigSQL** to restore it. *See the System Platform Installation Guide* for more information.

About OS group-based security

If you use OS Group-based Authentication Mode, make sure you understand the Windows operating system, particularly its user permissions, groups and security features. OS Group-based security uses these Windows features. For more help, see the Microsoft online help or a 3rd party book about Windows security.

When you use local OS Groups as Roles, each node within the Galaxy must have the same OS Users, Groups, and user-group mappings to get the same level of access to the user at each node.

Note: Application Server uses Pre-Windows 2000 Group names for OS Group-based Authentication Mode, not the Active Directory Common Name.

Connecting to a remote node for the first time

A newly-added user working on a computer with no access to the Galaxy Repository node cannot write to an attribute on a remote node if that user has never logged on to the remote node. This is true even if the user is given sufficient run-time operational permissions to do writes. To enable remote writing capabilities, log on to the remote node at least one time.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

When attempting to access a remote Galaxy Repository node that contains an IDE, the user who is logging in must have admin privileges.

Cached data at log in

If you log on to System Platform on a workstation that belongs to Domain A and Domain Controller A fails, locally cached login data is used on subsequent log on attempts. When the domain controller returns to operation, your log on fails during the time period that trusts are being reestablished by the controller.

If, during the controller outage, your username/password data changed, you can use the old log on data if you intend to work locally.

If you want to perform remote operations, you should log on with the new log on data. If that fails, the trusts are being reestablished by the controller, and you should retry at a later time.

The user's full name is not available to any client (for example, an InTouch window) if the domain controller is disconnected from the network when the user logs on to System Platform for the first time.

If the user previously logged on to System Platform when the domain controller was connected, the user's full name is still available to the client from data stored in cache even if the domain controller is disconnected when the user subsequently logs on to System Platform.

Mixed or native domains

Your unique listing maps to the list of domains and user groups you see when you use the Windows tool for managing your domain. The list of domains and user groups appears differently in the group browser depending on whether your domain is configured as a Mixed or Native domain.

The two modes cannot be used interchangeably. We recommend using Native-mode domains for the mode's advantages in security and use of domain local groups.

Domain local groups

Domain local groups (DLGs) cannot be used in a mixed-mode domain. If you use DLGs to define role permissions, when group members attempt to access secured content, those group members might see "access denied" error messages.

This problem can occur if your domain is operating in mixed mode. In mixed mode, the scope of a DLG is limited to the domain controllers only. The DLG is not valid for member servers. The DLG can still appear valid as certain applications, such as SharePoint Portal Server resources, do not filter out invalid DLG entries.

If you grant the necessary role permissions to the domain user accounts individually, those users can gain access to the secured content, but this approach nullifies one of the advantages of using DLGs.

Security and distribution domain configurations

You must select one of two group types when you create a new group.

- Security Group Type

  Use security groups to manage user and computer access to shared resources and to control which users receive group policy settings.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

- Distribution Group Type

  Use distribution groups as email distribution lists with email applications such as Microsoft Exchange or Microsoft Office Outlook. You cannot use distribution groups to assign permissions or to filter group policy settings.

  For this reason, a domain group configured as a distribution type rather than a security type cannot be used for security purposes.

InTouch access levels security

The **Roles** page includes the **Access Level** column. The **Access Level** is an InTouch function.

In InTouch, access levels are a schema for prioritizing run-time functions. In the System Platform security model, it only maps to InTouch values and has no prioritizing characteristics.

The maximum value is 9999 and the minimum is 0 (zero). If a user is assigned more than one role with different access levels, the higher access value is passed to InTouch.

Secured and verified writes

You can assign Secured Write or Verified Write security classification to a configurable attribute. These security classifications require authentication to perform run-time writes to the configured attribute. With authentication, users can write to such attributes by:

- Any assignment in a script that sets the value of the attribute, such as `"A=B";`

  where A references an attribute that is configured for Secured Write or Verified Write security classification.
- Any action on an animation graphic that alters the value of an attribute that has Signed Write or Verified Write security classification, such as a user input, a slider, an up/down button on a counter, or any other such actions.
- A script that uses the SignedWrite() function.

  For information about the SignedWrite() function, see Chapter 2, "QuickScript .NET Functions," in the *Application Server Scripting Guide*.

The operator and verifier provide credentials for authentication by entering a valid security account (domain name, user name, and password), or by using a Smart Card if a Smart Card reader is attached to the system.

Note: Smart Card authentication is not supported in multi-galaxy environments for read/write operations to remote galaxies.

Operators can write to attributes configured with Secured Write or Verified Write security classification even if another user is logged on. This does not affect the session of the logged-on user.

- The operator must have the Can Modify Operate Attributes operational permission to perform the Verified Write.
- The verifier must have the Can Verify Writes operational permission to confirm the Verified Write.

  For more information, see Set object security and About roles.

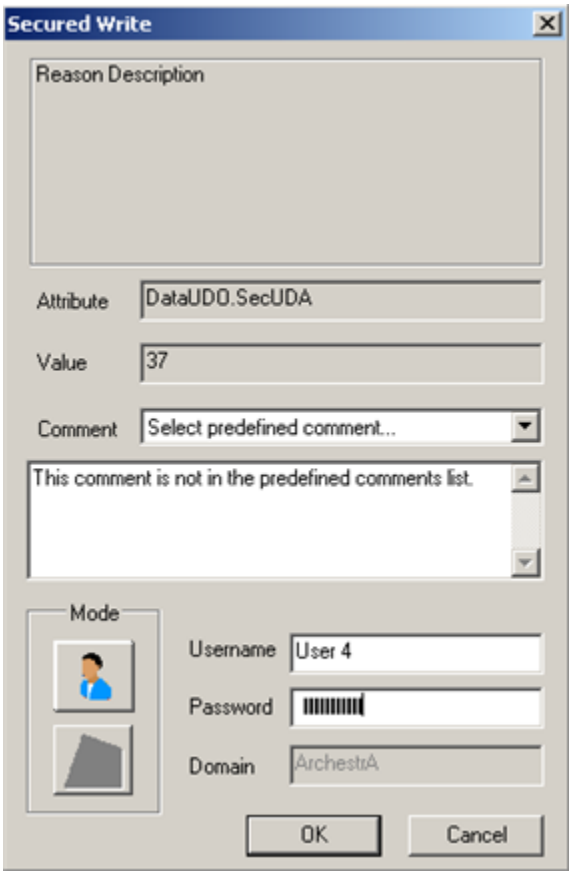
  Note: The Secured and Verified Write features work only when security is enabled on both the Galaxy and InTouch applications and do not work if either Galaxy or InTouch security is set to None.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Configurable descriptions and comments for secured and verified writes

At run time, the Secured Write or Verified Write dialog boxes appear when the operator attempts to write to an attribute configured for Secured Write or Verified Write. The dialogs enable configurable user input that can be used to provide information about the Secured or Verified Write.

Use the SignedWrite() script function to configure these inputs. The script function can be used only for attributes configured with Secured Write or Verified Write. It can be used only for symbol scripts and not for application object scripts.



Reason description

The reason description is specific to an attribute, and explains the purpose of the attribute and the impact of changing it. If you do not configure a reason description, the reason description area is blank.

It is possible to use a script to directly assign a value to an attribute or field attribute that requires Secured or Verified Write. When the script is executed, the Secured or Verified Write dialog box appears and the reason description area displays an Attribute description, if there is one. If the Attribute does not have a description, then the reason description area displays the description of the Application Object to which the attribute belongs.

Predefined comment list

The predefined **Comment** list enables the operator to comment on changing the attribute by selecting from a predefined list of comments.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Editable comment box

You can enable the operator to enter a comment in the **Comment** box.

For more information on configuring the reason description and the predefined **Comment** list and box, see Working with the SignedWrite() function for secured and verified writes.

Note The reason message and **Comment** box and list are displayed in the Secured Write or Verified Write dialog box only in InTouch WindowViewer and not in Object Viewer.

Logged information for secured and verified writes

Following a Secured or Verified Write a security Event is written to the event log, including the following information:

- The signee name
- Verifier name, if any
- Type of write: "Secured Write" or "Verified Write"
- Comment, if any entered by user
- Reason Description, if any provided
- Attribute or field attribute description, if any, or the Short Description of the Application Object, if no Attribute description exists

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Design standards

Designing object templates before beginning to define specific assets makes the process easier and your organization more consistent.

Galaxy options

There are a number of Galaxy-wide preferences that you can set from the System Platform IDE backstage, which you access by selecting **Galaxy** from the menu bar. These include settings for:

- Auto-buffering
- Advanced communications management
- Out-of-order data handling
- Alarm latching
- Configure the IDE Authentication Mode

Auto-buffering

The **Enable auto-buffering** option lets you enable the data buffer property for all user-created dynamic attributes on a Galaxy-wide basis, without having to set the data buffer property for individual attributes. Application Server automatically configures buffer when auto-configure buffering is enabled. If your Galaxy includes legacy Field Attributes, data buffering is enabled for these also when buffering for the Galaxy is enabled.

Data buffering ensures that any and all data changes are captured if a given attribute changes its value several times during a single scan cycle. There is no folding of data, which occurs when an accumulation of values of multiple data changes within a single scan are overwritten and only the latest value is stored. This would be the case without buffering.

Even with data buffering enabled, only the final value is propagated to HMI or other user-facing touch points within System Platform, but interim values are historized. If data buffering is not enabled, interim values accumulated within a single scan cycle are not preserved. With data buffering enabled, the entire set of buffered data values is historized along with time and quality (VTQ). The entire set of buffered values can also be viewed within the Object Viewer.

If Galaxy-wide data buffering is not enabled, you can enable buffering for individual attributes that are I/O-enabled. See Enable I/O for an attribute for details. The behavior of the buffer property remains the same, whether it is applied across the Galaxy or if it is enabled only for individual attributes.

Note: All user-created attributes are enabled for buffering when auto-configured buffering is enabled, not just those attributes with the I/O feature enabled. With auto-configured buffering enabled, Application Server dynamically creates a buffer for the user-created attributes, even if no values are being buffered.

Enable auto-buffering

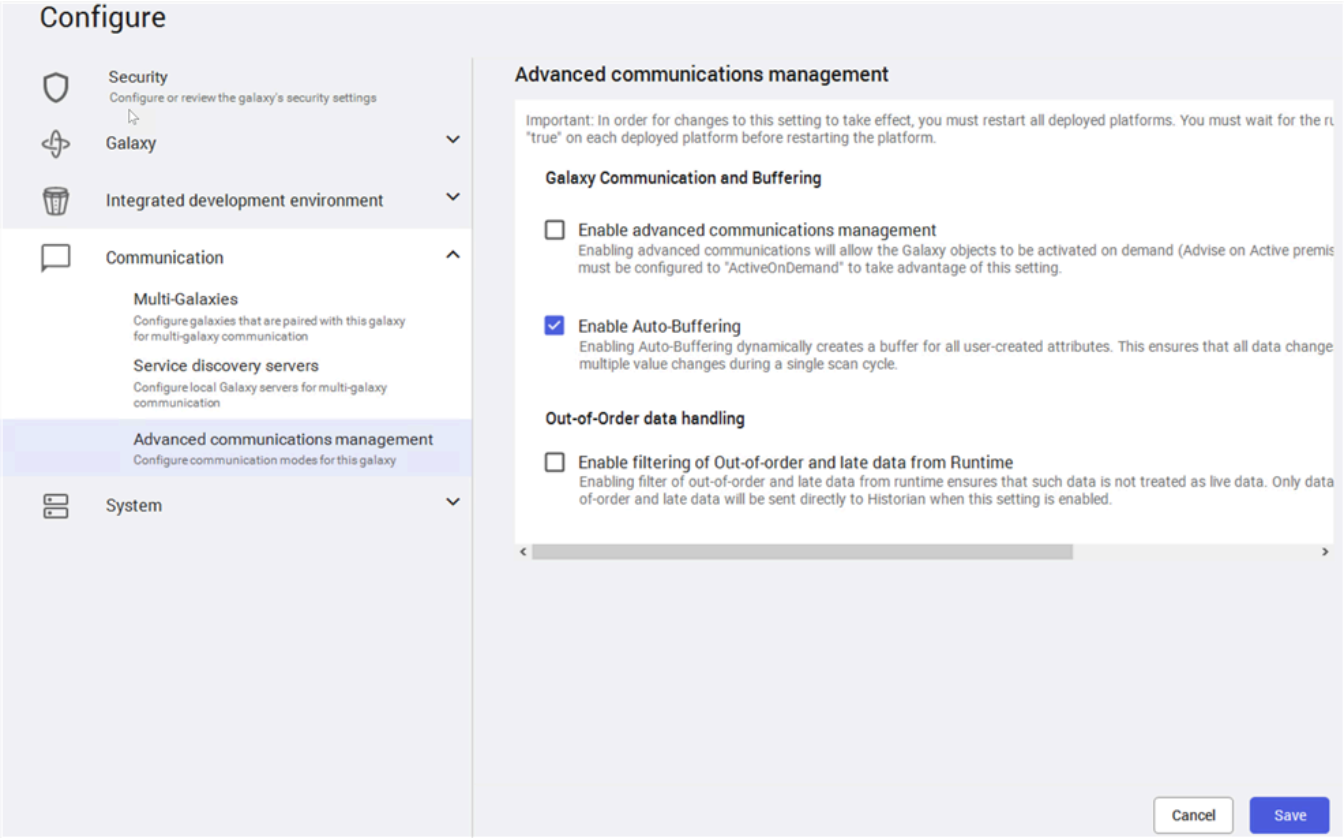
When auto-buffering is enabled for the entire Galaxy, data buffering for individual attributes cannot be enabled

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

or disabled. Application Server will automatically generate a buffer attribute, with the extension .buffer, for each user-created dynamic attribute. Note that when auto-buffering is enabled, the buffer setting for individual attributes is ignored. Buffering remains in effect for all user-created dynamic attributes, whether or not buffering is enabled for individual attributes. This behavior also applies to legacy field attributes, if your Galaxy has any.

To set buffering for the Galaxy, select **Galaxy** from the menu bar to open the backstage, select the **Configure** tab, then select **Communication**.

- To enable Galaxy-wide buffering, set the checkmark for **Enable Auto-Buffering**. The automatically-created buffer for each user-created dynamic attribute appears at run time.
- To allow buffering to be set explicitly for individual I/O-extended attributes, make sure the checkmark is not set for **Enable Auto-Buffering**.



With auto-buffering disabled for the Galaxy, you can enable buffering for individual attributes when I/O is enabled for the attribute. The extension for these individually-configured buffer attributes is the same as for automatically configured buffer attributes (.buffer).

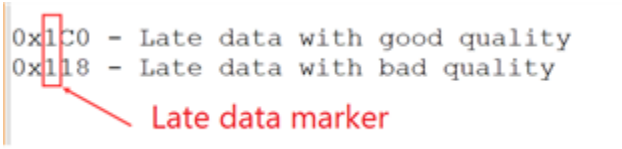
Advanced communications management

The **Enable advanced communications management** option, when enabled, allows Galaxy objects at run time to be activated only when necessary. Enabling this setting can improve run-time performance. Only objects in scan groups that are configured as **ActiveOnDemand** are affected. For additional information, see Configure scan modes.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Out-of-order data handling

The **Enable filtering of out-of-order and late data from run time** option controls how data that has a VTQ late data marker is handled.



The late data marker bit is set to signify that one of two conditions exists:

- That VTQ time stamp of the data update is older than the current or last known data that was received from the same source.
- That the "is_storeforward" attribute is set to true in the VTQ for the data that is published by an MQTT provider.

When Out-of-order data handling is enabled, Application Server will not use data that is marked as late (whether it is late, out-of-sequence, or has the "is_storeforward" attribute set to true) for any of the following purposes:

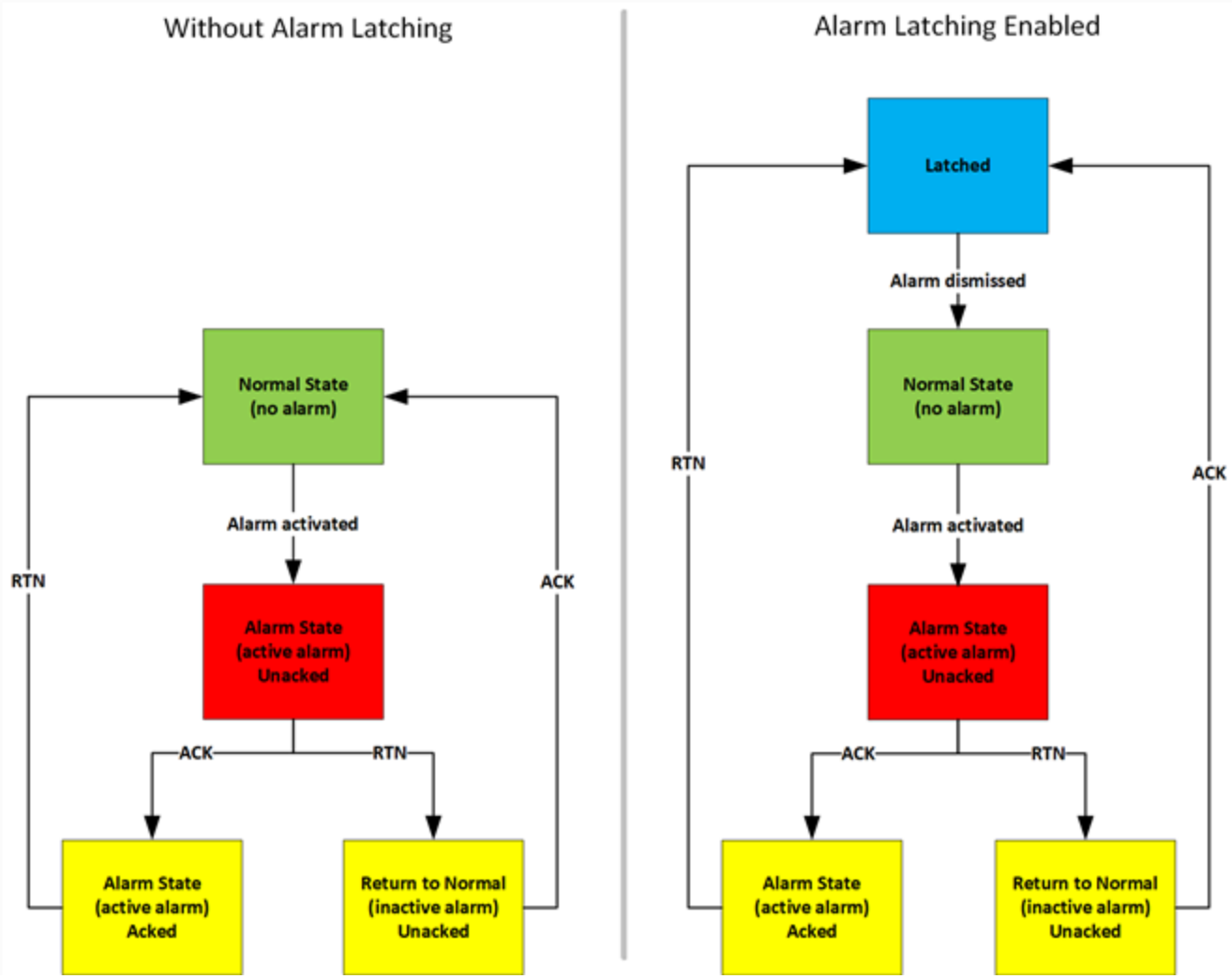
- The data will not be used to generate alarms.
- The data will not be used to trigger scripts.
- The data will not be distributed to any internal or external IO references.
  - Internal IO references include cross-engine and cross-galaxy references.
  - External IO references include references to OPCUA clients.

The late/out-of-sequence/storeforward data is not displayed and is sent directly to the Historian for historization.

Alarm latching

Alarm latching is an additional alarm state that you can configure on a Galaxy-wide basis. With alarm latching enabled, run-time clients continue to display resolved alarm conditions, even if the condition that caused the alarm state has since returned to normal and/or has been acknowledged. A **dismiss** command from an authorized user is required to remove the latched alarms from run-time displays. The dismiss command is similar to an acknowledge (Ack) command. The latched state provides enhanced visibility and an additional check for alarms that have been acked and/or returned to normal. See Dismiss a latched alarms for details.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.



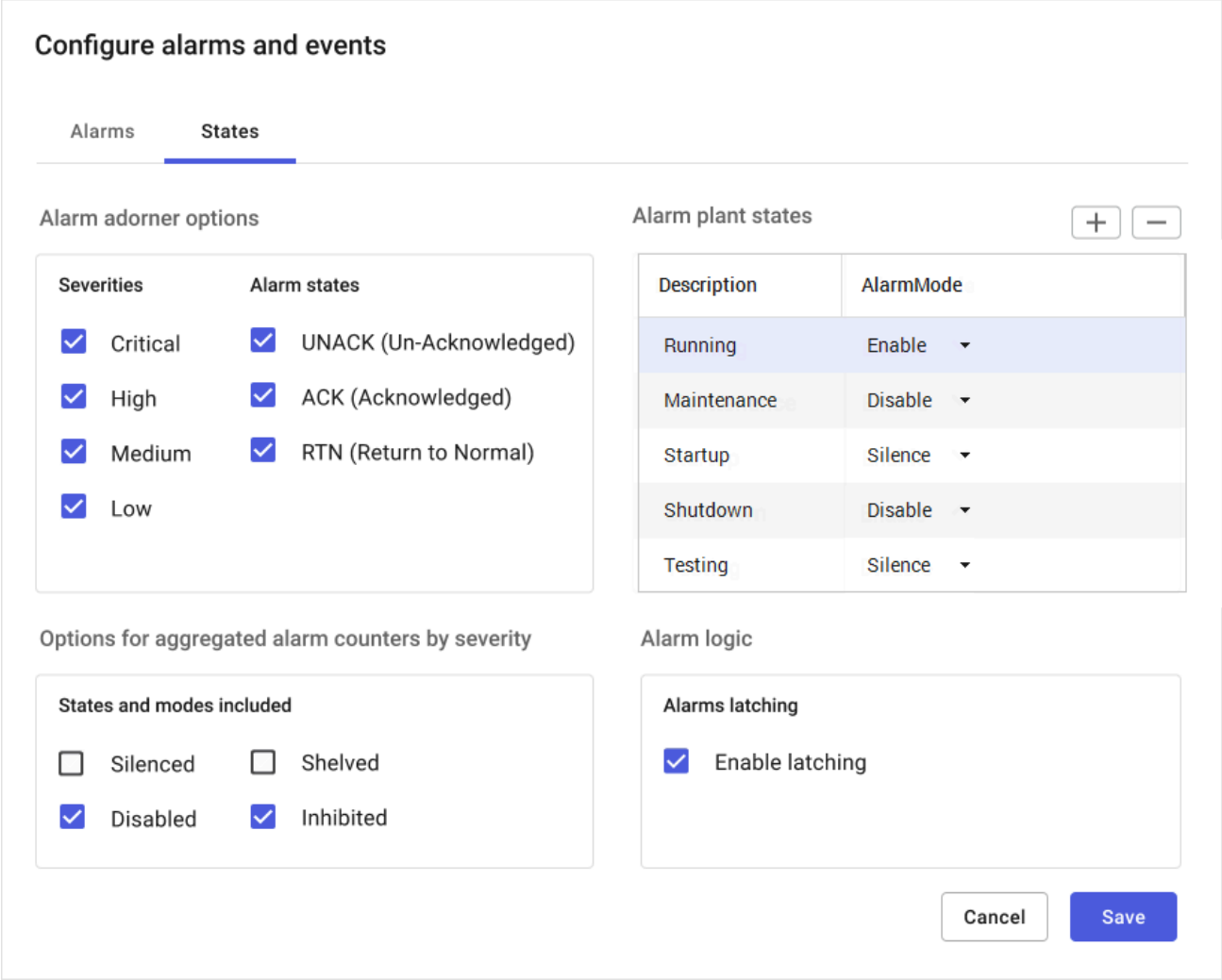
Enable alarm latching

Alarm latching is enabled or disabled on a Galaxy-wide basis. The default is disabled.

To enable alarm latching for the Galaxy, select **Galaxy** from the menu bar to open the backstage and then select the **Galaxy** tab. From the Galaxy tab, select **Alarms and events**, and then select **States**.

- To enable alarm latching for the Galaxy, set the checkmark for **Enable latching**.
- The alarm latching state is not present in the Galaxy if the checkbox is not enabled.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.



When alarm latching is enabled for the Galaxy, a fourth alarm state called LATCHED is created. A latched alarm is one that has LATCHED status. The latched alarm remains visible at run time until an operator explicitly issues a "Dismiss" command.

About templates and instances

Before you start modeling your application using the Application Server, you must understand templates and object instances.

Templates are elements in Application Server that contain common configuration parameters for object instances that you use multiple times in your application.

For example, you might create a template for valves. You configure the template with all the unique attributes for valves. You use that template to make object instances of valves. You can further configure and customize each object instance to represent a specific valve.

Object instances are the specific devices in your environment, such as diaphragm valves or very complex devices, like a reactor. You create an instance from a template and then customize the specific instance as needed.

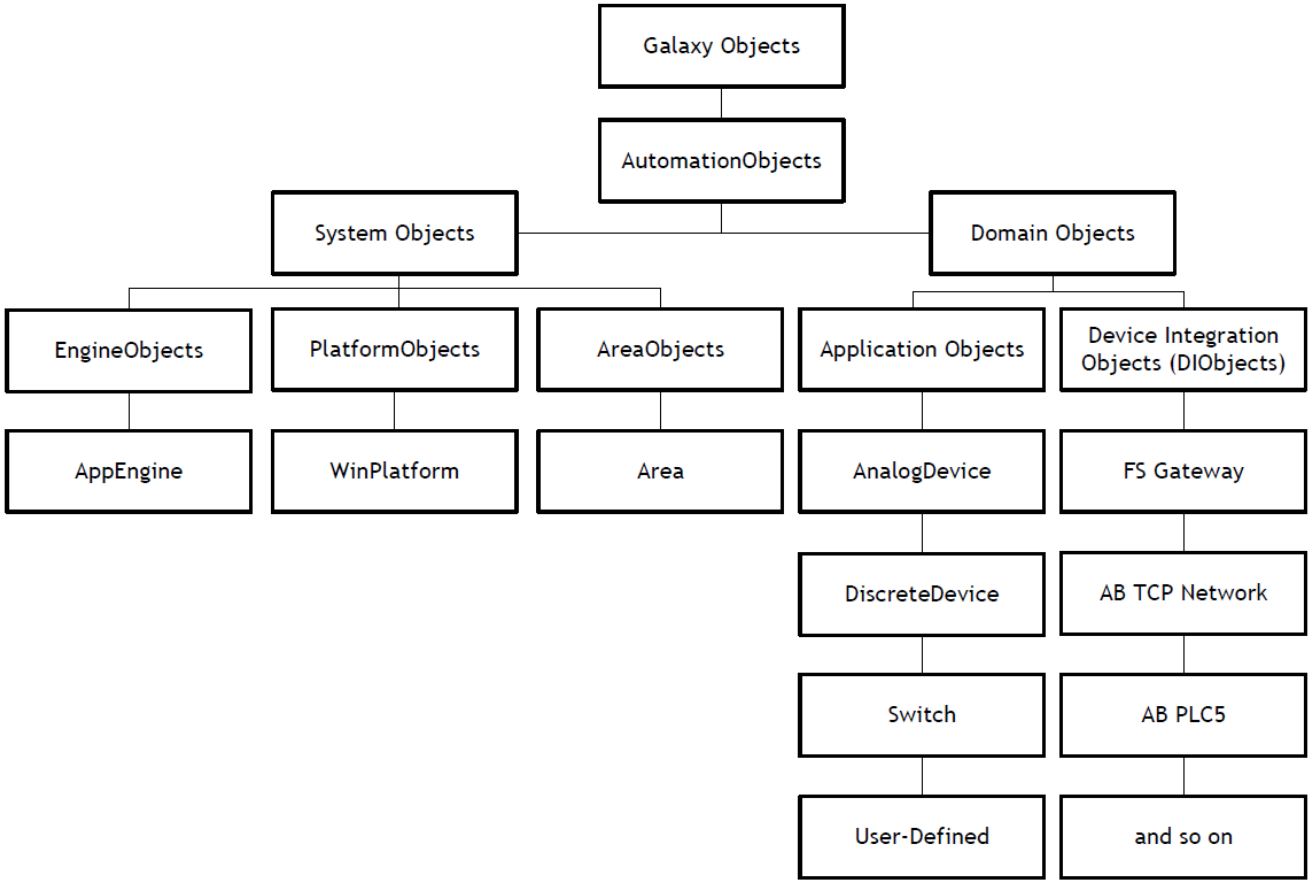
Instances are deployed to the runtime environment. Templates exist in the development environment and

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

cannot be deployed.

Creating templates and instances is very similar to object-oriented programming. For example, templates and instances have a parent/child relationship that involves inheriting attributes. There are differences, however, between object-oriented programming and creating templates and instances in Application Server.

Collectively, templates and instances are called objects. The following graphic shows the different kinds of objects and how they are organized.



If you are new to this kind of programming, the next section explains the basic concepts you need to know before you start. If you are familiar with object-oriented programming, the concepts in the next section may be familiar to you, but notice the important differences between object-oriented programming and Application Server.

Instances

Instances are the run-time objects created from templates in Application Server. Instances are the specific things in your environment like processes, valves, conveyor belts, holding tanks, and sensors. Instances can get information from sensors on the real-world device or from application logic in Application Server. Instances exist during run time.

In your environment, you may have a few instances or several thousand. Many of these instances may be similar or identical, such as valves or holding tanks. Creating a new valve object from scratch when you have several thousand identical valves is time-consuming. That’s where templates come in.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Templates

Templates are high-level definitions of the devices in your environment. Templates are like a cookie cutter from which you can make many identical cookies.

You define a template for an object, like a valve, one time and then use that template when you need to define another instance of that item. Template names have a dollar sign ($) as the first character of their name.

A template can specify application logic, alarms, security, and historical data for an object.

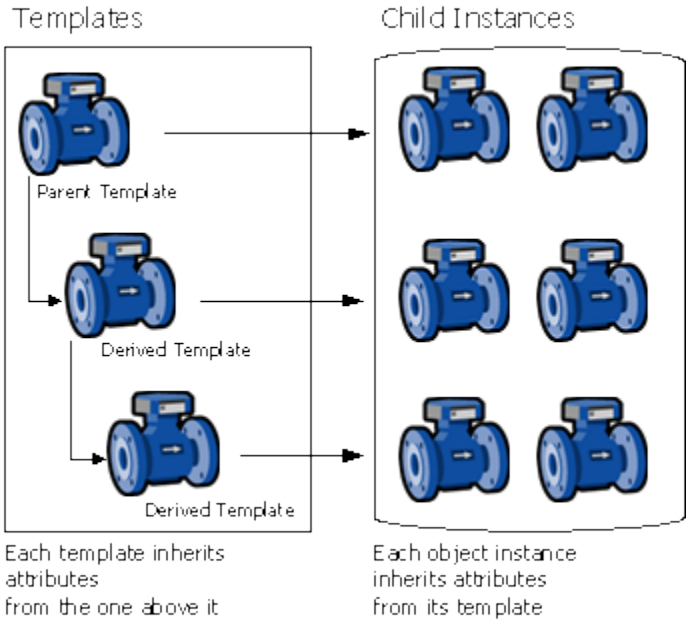
A template can also define an area of your environment. You can extend and customize a template by adding attributes, scripts, or features to meet the specific needs of your environment. Objects inherit attributes, scripts and features from their parents.

Application Server comes with predefined templates, called base templates. You cannot modify base templates. All templates you create are derived from base templates.

You can also nest templates, or contain them. Contained templates consist of nested object templates that represent complex devices consisting of smaller, simpler devices, including valves. A reactor is a good candidate for containment.

Templates only exist in the development environment.

Using the Diaphragm valve template, you can quickly create a Diaphragm valve instance when you need another Diaphragm valve in your application.



Propagation

If you need to change something about all diaphragm valves, you can change the template for the Diaphragm valve and all diaphragm valves in your application inherit the changes, assuming the attributes are locked in the parent template. This makes it easy to maintain and update your application.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

About derived templates

All templates you create within the IDE are derived templates.

When creating your Galaxy application, plan ahead and create derived templates for devices of a certain type so you can use the templates to create the individual instances.

A new derived template is an exact copy of its parent template with the possible exceptions of locking and security and modified attribute values. You can lock attributes to prevent them from being modified in child templates.

After you create a new derived template, you can customize it. For more information about customizing and extending templates, see Create derived templates.

Every template has a set of attributes and default values. When you create an instance, attributes are inherited by the instance. In the instance, you can reconfigure many of the attributes inherited from the parent template if they are not locked on the parent template.

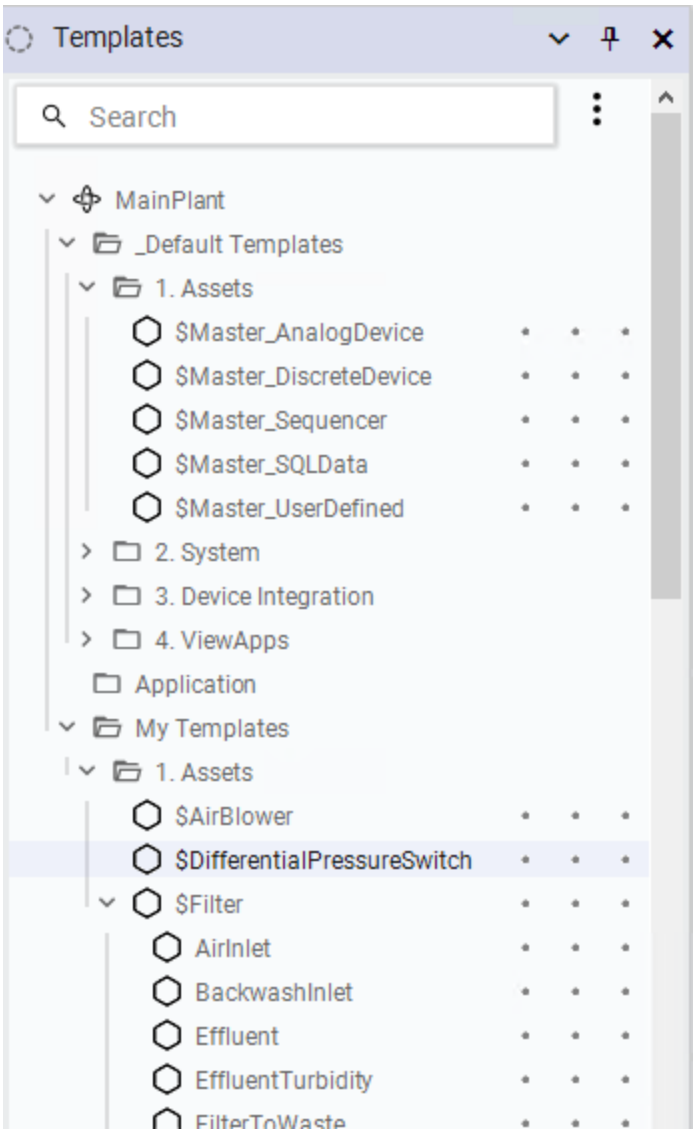
For more information about customizing instances, see Edit objects.

About base templates

When you first open the IDE, you see a set of read-only base templates, which are indicated by an orange lock symbol. If you based your Galaxy on the Default.cab, you will also see a set of first-level derived templates, separated into subfolders, in the **Template folder**.

You cannot modify base templates, but you can use them to create derived templates, which are copies of the base templates. We recommend that you use the first-level derived templates that are provided under the **_Default Templates** folder to create additional templates. Most of these first-level derived templates have the prefix **Master_**. You can modify derived templates and use them to create instances of them for your applications.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.



The Template folder includes:

- Asset templates

  Use these templates to represent real devices in your Galaxy, such as pumps and valves.
- Device Integration templates

  Use these templates to create instances that communicate with external devices, such as PLCs.
- System templates

  Use these templates to define system instances, like other computers.

Application templates

These base templates let you easily create devices in your Galaxy. They contain the properties you need to set for each kind of device. For example, a DiscreteDevice device contains all the settings you need to specify for an on/off device. Of course, you can extend and customize any device by using attributes, scripts, and features.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Device integration templates

These base templates provide communication with external devices. External devices run on the application engine.

For example:

- DINetwork object – Refers to the object that represents the network interface port to the device through the Data Access Server. The object provides diagnostics, and configuration for that specific card.
- DIDevice object – Refers to the object that represents the actual external device (such as a PLC or RTU), which is associated to the DINetwork Object.

System templates

These objects represent the parts of a Galaxy and not the domain they are monitoring/controlling. These base templates let you create more system level grouping and computers, such as areas you add objects to or another host AppEngine.

WinPlatform object

The WinPlatform platform object is a key base object because you need a platform to host the objects you are modeling. This object:

- Calculates various statistics for the node it is deployed to. These statistics are published in attributes.
- Monitors various statistics related to the node it is deployed to. These monitored attributes can be alarmed and historized.
- Starts and stops engines which are deployed to it, based on each engines' startup type .
- Monitors the running state of engines deployed to it. If the platform detects an engine failed, it can, optionally based on the value of the engine’s restart attribute, restart the engine.

AppEngine object

The AppEngine object must have a Platform on which to run. This object:

- Hosts ApplicationObjects, Device Integration objects and areas.
- Contains the logic to set up and initialize objects when they are deployed.
- Contains the logic to remove objects from the engine when they are undeployed.
- Determines the scan time which all objects within that particular engine run.

Area object

All application objects belong to an area. Areas can contain sub-Areas. Areas provide a key organizational role in grouping alarm information and providing that information to those who use alarm/event clients to monitor their areas of responsibility.

The values of three Area object alarm attributes can be saved to the historian:

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

- Active alarm counter
- Unacknowledged alarm counter
- Disabled (or silenced) alarm counter

ViewEngine object

The ViewEngine object must have a Platform on which to run. This object:

- Hosts ViewApp objects which will be used through the desktop client.
- Hosts InTouchViewApp objects.
- Contains the logic to set up and initialize objects when they are deployed.
- Contains the logic to remove objects when they are undeployed.
- Determines the scan time which all objects within that particular engine run.
- Determines if the ViewApp objects that it hosts will be read/write or read-only.

  Note: InTouch applications are not affected by the "ViewApps are Read Only" setting of the ViewEngine. This setting is used for the ViewApp object only.

WebViewEngine object

The WebViewEngine object must have a Platform on which to run. This object:

- Hosts OMI ViewApp objects which will be used through the OMI web client.
- Sets up a web server and content server to serve the OMI ViewApp and its content files to the clients.
- Sets up a telemetry endpoint allowing OMI clients running in browsers to subscribe and receive telemetry values from the Galaxy.
- Sets up authentication mechanisms allowing users to log in to the OMI ViewApp based on the Galaxy configuration.

There are no configurable attributes or other settings for WebViewEngine objects. No data is historized for them and no alarms can be configured for any of their attributes.

InTouchViewApp object

The InTouchViewApp object must have a ViewEngine on which to run. This object:

- Manages the synchronization and delivery of files required by the associated InTouch® application.
- Provides runtime access to tags on the associated InTouch application.
- Starts WindowMaker for the associated InTouch application when edited.

  Note: InTouch applications are not affected by the "ViewApps are Read Only" setting of the ViewEngine that hosts the InTouchViewApp object. This setting is used for the ViewApp object only.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

ViewApp object

The ViewApp object must have a ViewEngine (for access from the Application Manager) or WebViewEngine (for OMI web client access) on which to run. This object:

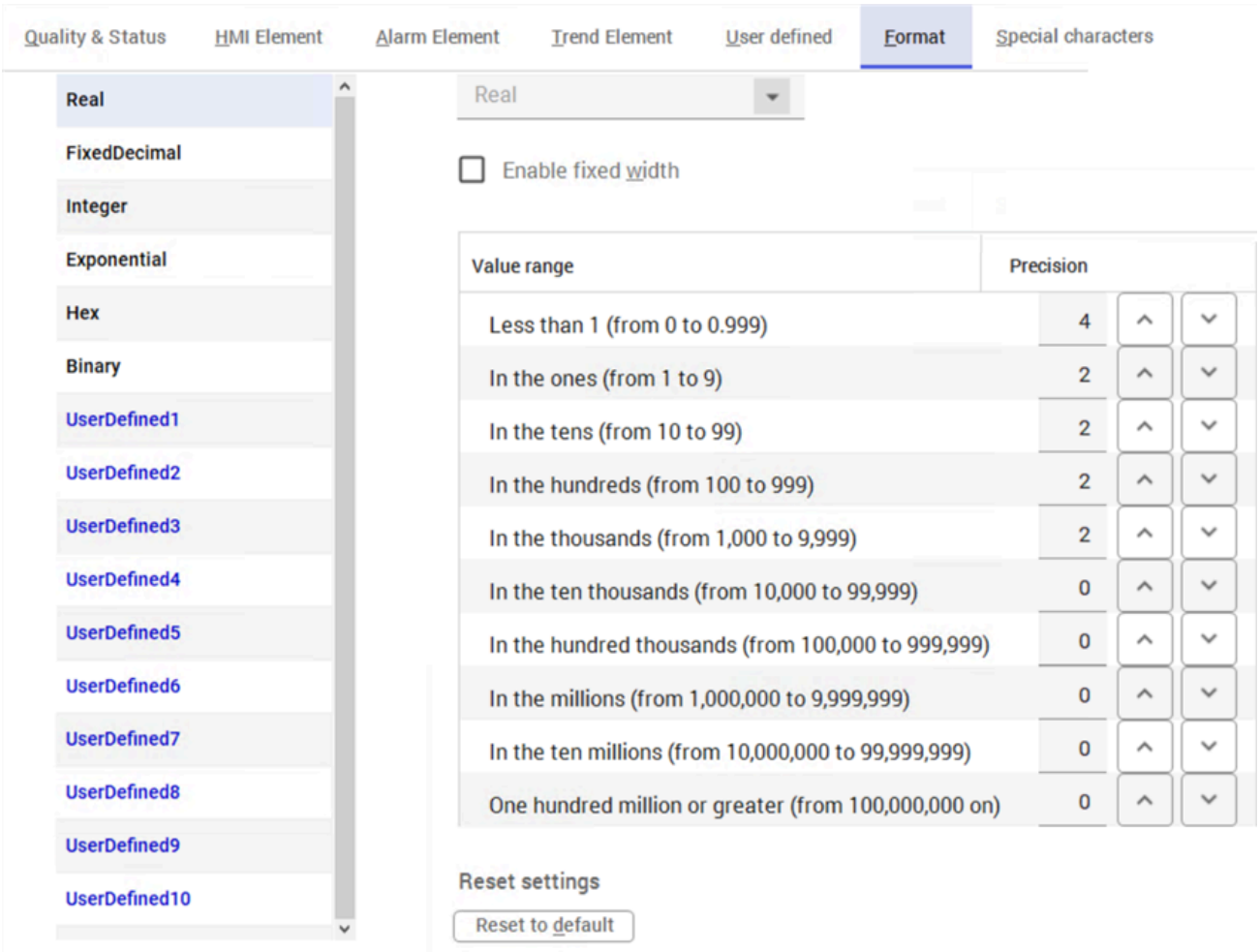
- Provides visualization at runtime for an AVEVA OMI application.
- Incorporates a Screen Profile and the content defined in an associated Layout.
- Provides runtime access to your production environment.
- A ViewApp deployed on a ViewEngine (as opposed to a WebViewEngine) can be read/write or read-only, depending on whether the ViewEngine hosting the ViewApp object has its "ViewApp are Read Only" setting enabled (ViewAppsReadOnly attribute set to true).

About the galaxy style library

A Galaxy Style Library applies a set of visual and numeric styles to ViewApps in the current Galaxy. Styles can be set for text, lines, fill colors, outlines, etc. The Galaxy Style Library contains the following style elements:

- The **Quality and Status** tab sets the physical appearance of quality and status icons. Also, the **Quality and Status** tab includes a Status group to assign the visual characteristics to error condition status icons.
- The **HMI Element** tab sets the appearance of various HMI aspects, such as background color colors, bad values, default fonts, etc.
- The **Alarm Element** tab sets the appearance of alarms.
- The **Trend Element** tab sets the appearance of the pens and background grid used to show historical trends.
- The **User defined** tab can be used to create custom style elements.
- The **Format** tab specifies the format of numbers that appear in a ViewApp.
- The **Special characters** tab sets characters to be shown at runtime for bad quality/no value and values too large for a fixed field.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.



Open the galaxy style library

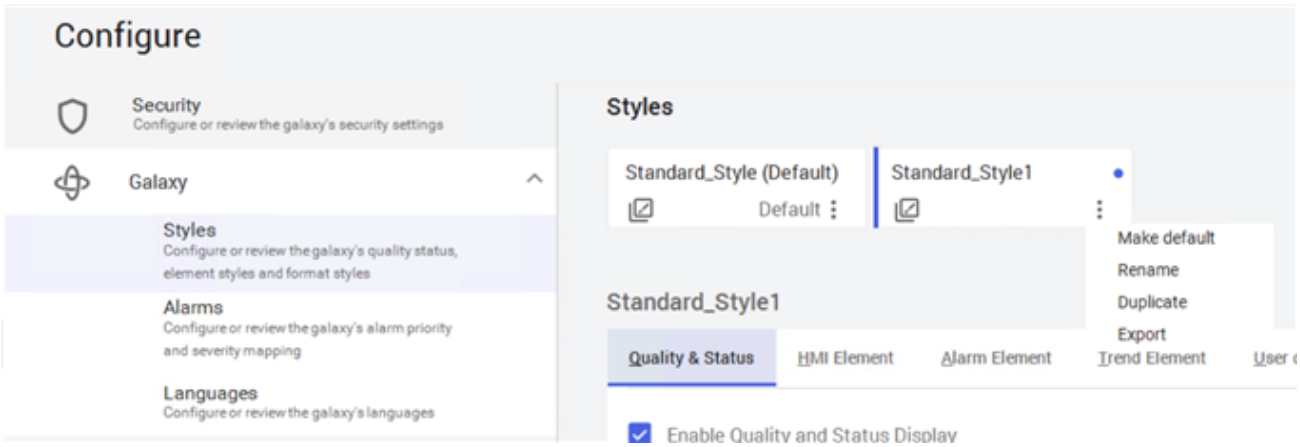
1. Select the **Galaxy** tab.
2. Select the **Configure** option, then select **Galaxy**.
3. Select the **Styles** option.

Manage galaxy style libraries

To manage your collection of Galaxy Style Libraries, select the vertical ellipse above the **Galaxy Style Libraries** field to show a list of management commands.

- Make default (enabled only if more than one Galaxy Style Library is available)
- Rename
- Duplicate
- Export

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.



Duplicate a galaxy style library

You can duplicate an existing Galaxy Style Library to add it to the list of other Galaxy Style Libraries defined for the Galaxy.

To duplicate a galaxy style library

1. Select the Galaxy Style Library to be duplicated.

   A blue background indicates the Galaxy Style Library is selected.
2. Select the vertical ellipse below the Galaxy Style name.
3. Select the **Duplicate** command from the context menu.
4. Click **Save** to save the duplicated library.

The duplicated Galaxy Style Library appears with a default name of StandardStyle*n*, where *n* is a single digit. A blue dot next to the Galaxy Style is shown until the new style is saved.

Set the default galaxy style library

The default Galaxy Style Library shown from the list is active when a ViewApp starts in a Galaxy.

To set a default galaxy style library

1. Select the Galaxy Style Library to be set as the default.

   A blue background indicates the Galaxy Style Library is selected.
2. Select the vertical ellipse below the Galaxy Style name.
3. Select the **Make default** command from the context menu.
4. Default is shown in parentheses next to the name of the Galaxy Style Library, and no longer appears next to the previous default library name.

Rename a galaxy style library

Galaxy Style Libraries should be named using a convention that suggests the theme of the element styles or their purpose. Also, a new Galaxy Style Library should be renamed from its default name after it has been added.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

To rename a galaxy style library

1. Select the Galaxy Style Library to rename.

   A blue background indicates the Galaxy Style Library is selected.
2. Select the vertical ellipse below the Galaxy Style name.
3. Select the **Rename** command from the context menu.
4. Enter the new name for the library.

   A warning message is shown if you type an invalid name.
5. Click in the Galaxy Style Libraries area to deselect the Galaxy Style Library and verify if you typed a valid name. You can also press **Enter** to validate the new name.

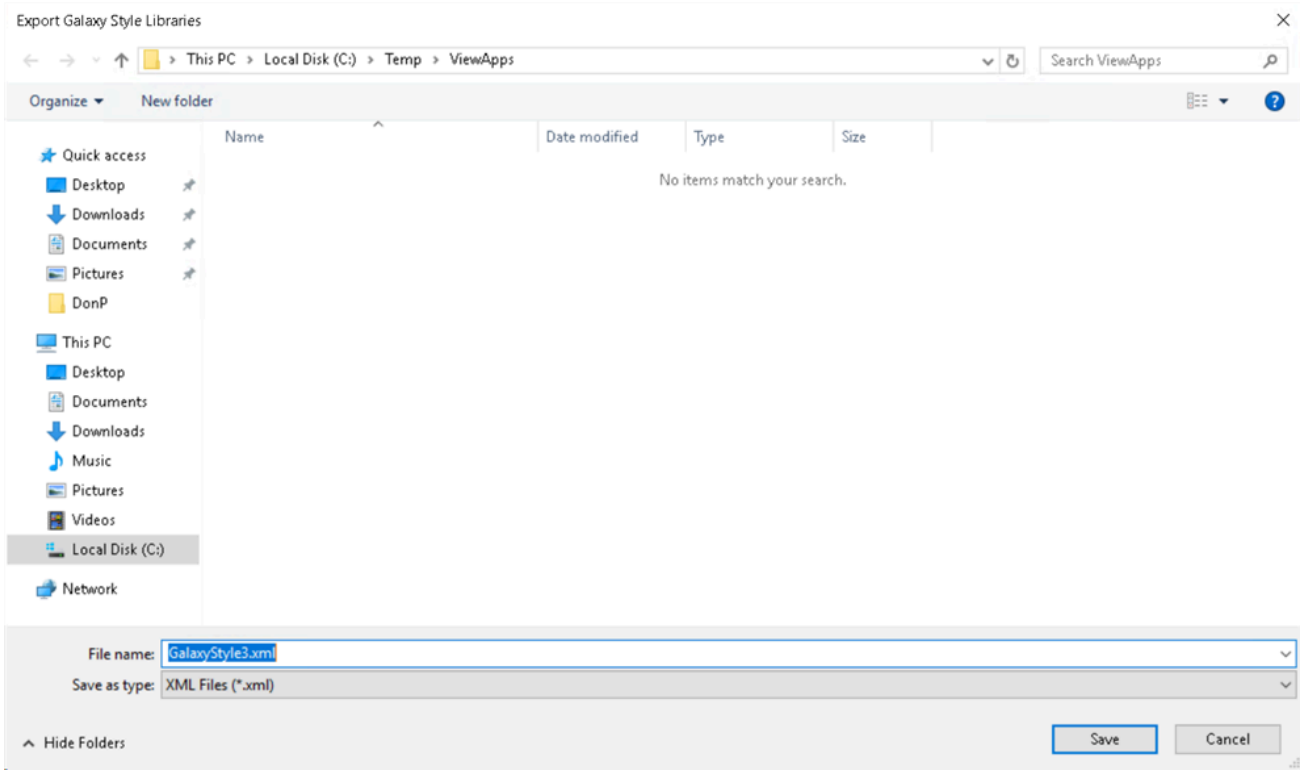
Export a galaxy style library

A single Galaxy Style Library can be exported as an xml file to a designated folder location.

To export a galaxy style library

1. Select the Galaxy Style Library you want to export.

   A blue background indicates the Galaxy Style Library is selected.
2. Select the vertical ellipse below the Galaxy Style name.
3. Select the **Export** command from the context menu.
4. In the **Export Galaxy Style Libraries** dialog, select the folder where you want to save the xml file.



5. The default name of the exported xml file is the name of the exported Galaxy Style Library. You can edit this in the dialog.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

6. Click **Save**.

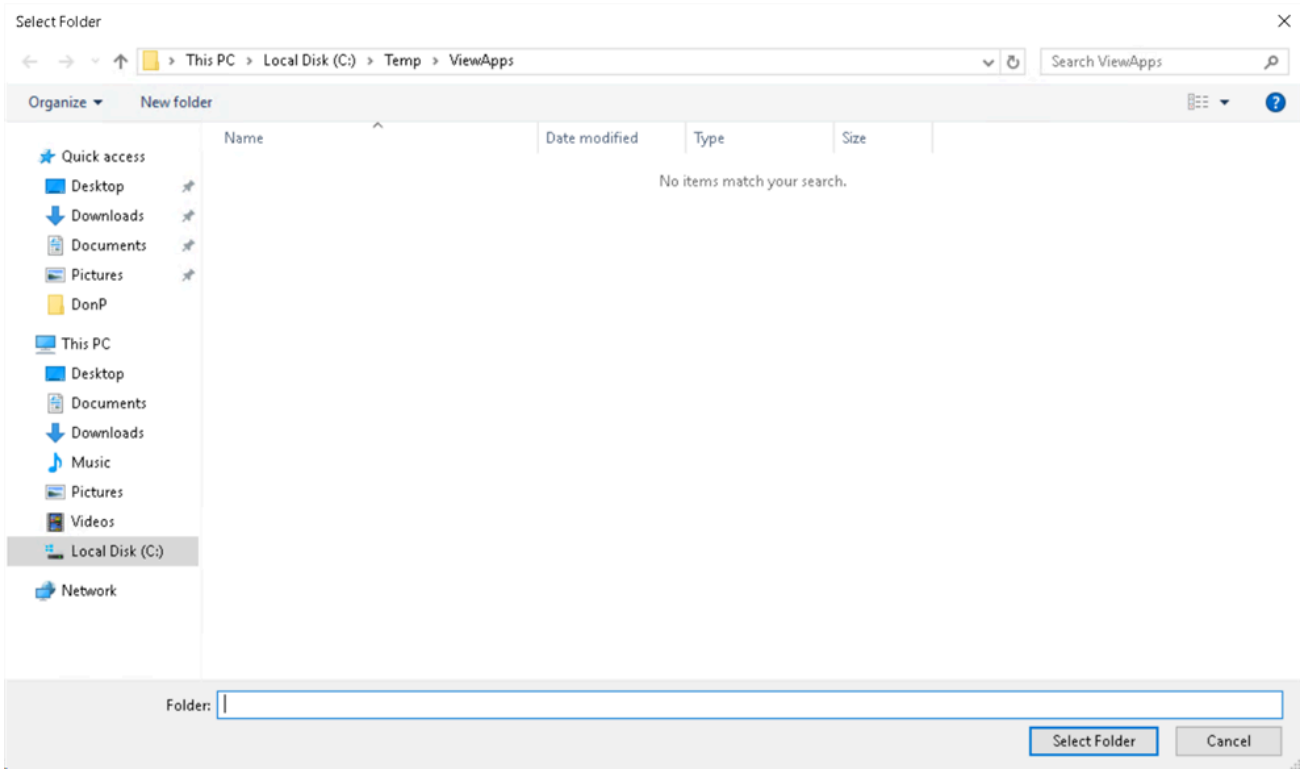
Export all galaxy style libraries

All Galaxies shown in the **Galaxy Style Libraries** list can be exported together to a common folder location.

To export all all galaxy style libraries

1. Select the **Galaxy** tab in the System Platform IDE.
2. Select the **Export** option, then select **All Style libraries**.

   The **Export all Styles** dialog opens.
3. Select the folder where you want to save the exported xml files and click **Select Folder**.



An xml file for each Galaxy Style Library in the Galaxy is created in the folder you selected. The name of each exported xml file is the name of the exported Galaxy Style Library.

Import a galaxy style library

You can import a Galaxy Style Library from an XML file.

Optional Galaxy Style Library XML files are installed with System Platform. These optional Galaxy Style Libraries are located in the ...\Program Files (x86)\ArchestrA\FrameWork\Bin\AdditionalElementStyles folder.

The names of the optional Galaxy Style Library XML files suggest the primary color schemes of the element styles within each optional Galaxy Style Library.

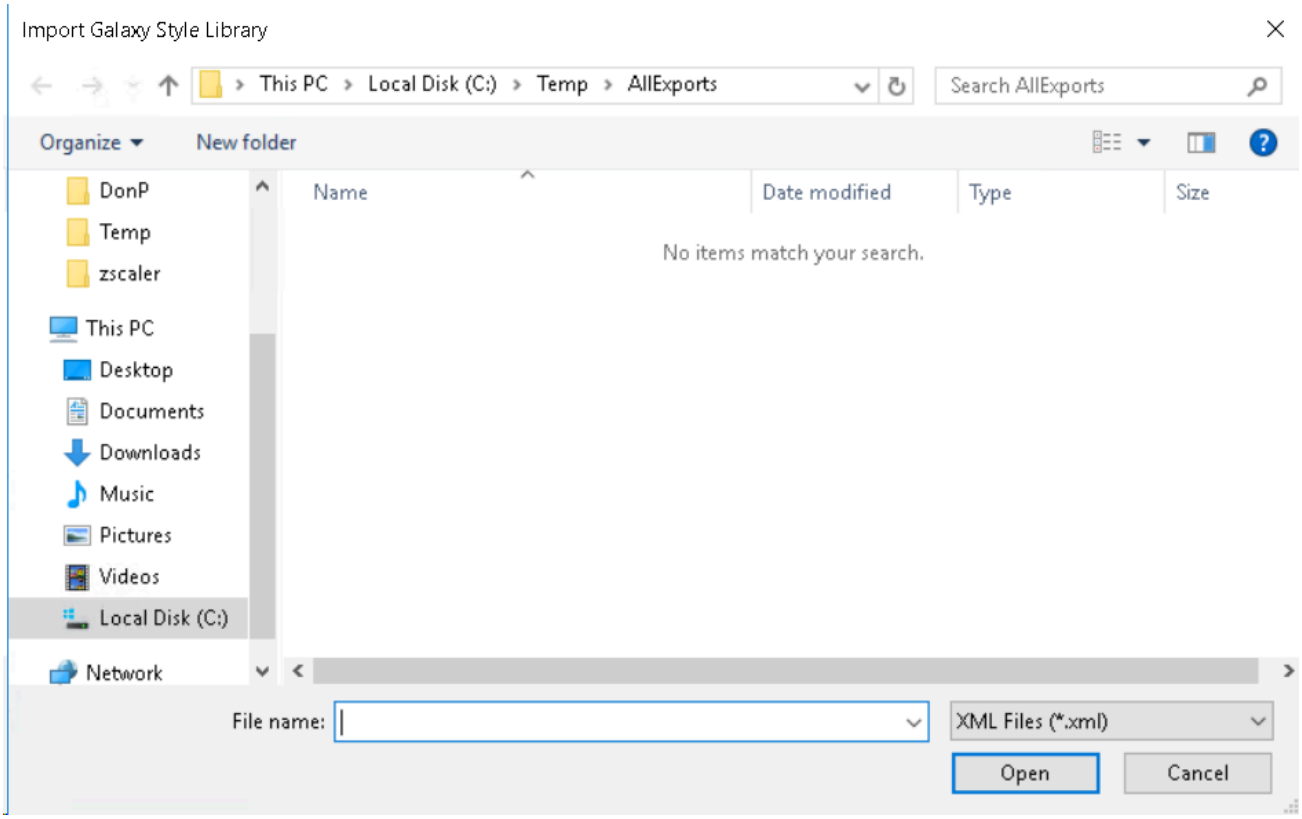
To import a galaxy style library

1. Select the **Galaxy** tab in the System Platform IDE.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

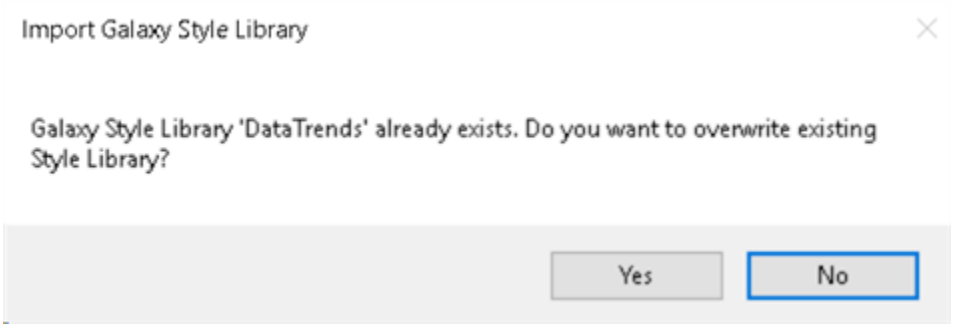
2. Select the **Import** option, then select **Style Libraries**.

   The **Import Style Libraries** dialog opens.



3. Select the folder containing the Galaxy Style Library to be imported.
4. Select the Galaxy Style Library xml file you want to import, then click **Open**. You can select multiple files for import.

   If the Galaxy Style LIbrary to be imported is a duplicate of a Galaxy Style Library that is already listed, a warning message appears with options to overwrite the existing Galaxy Style Library or not.



After successfully importing a Galaxy Style LIbrary, its name appears in the **Galaxy Style Libraries** list.

About element styles

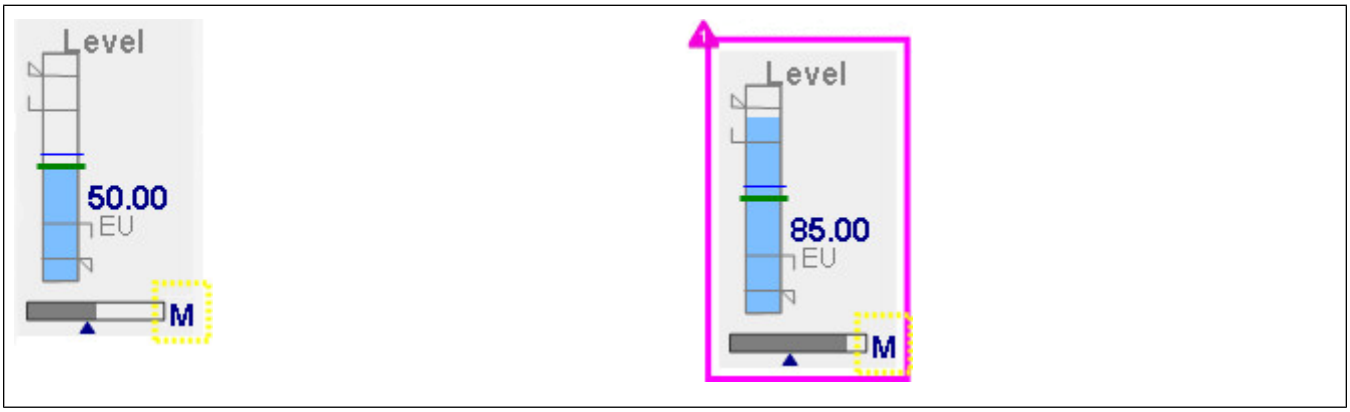
An element style defines a set of visual properties that determine the appearance of text, lines, graphic outlines, and interior fill shown in Industrial Graphics or graphic elements. An element style can be applied to a graphic

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

that sets pre-configured visual property values that take precedence over a graphic's native visual properties.

Element styles provide the means for developers to establish consistent visual standards in their AVEVA OMI ViewApps. An element style can define the same visual properties of text, lines, fill, and outlines for all graphics included in a ViewApp.

Likewise, element styles can be applied to a graphic during runtime to show the changing status of a piece of equipment in response to events or alarms. The following example shows an alarm border animation element style around a meter when the process value shown by the meter transitions into an alarm state.



Element styles details

A Galaxy Style Library includes pre-defined and user-defined element styles, plus numerical format styles and special characters used to show exceptions at runtime.

- Pre-defined element styles

  Pre-defined element styles have been configured with default values. You can modify each element style's font, fill, line, and outline properties.

  Pre-defined element styles are organized in groups belonging to the **Element Style Overrides** list by their functional purpose:

  - **Quality and Status**
  - **HMI Elements**
  - **Alarm Elements**
  - **Trend Elements**

    Many of the element styles apply to the graphic elements that comprise Situational Awareness Library graphics. An element style can define the visual properties of a graphic like its background color. An element style can also indicate a graphic's current operating state. For example, there is a set of pre-defined element styles that set the visual characteristics of machinery alarm states represented by a graphic.
- User-defined element styles

  User-defined element styles do not contain preset font, fill, line, and outline properties. You must specify each pre-defined element style's properties to meet the needs of your applications.
- **Format** styles are applied to numerical values, such as set points and process variables. There are pre-defined styles for real, Fixed Decimal, Integer, Exponential, Hex and Binary numbers. You can also configure user-defined formats. Format styles define range and precision for values.
- **Special characters** define two different special characters that are shown in alarm mode at runtime for the

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

following conditions:

- Bad quality with no value: default character is "!" (exclamation point)
- Value too large for fixed field: default character is "*" (asterisk)

You can create your own Galaxy Style Library by overriding the default style configurations and exporting the library. For information about setting element style overrides, see Change the visual properties of an element style (,).

If you want to restore the original default visual properties for your graphic elements, each of the styles pages includes a **Reset to Default** button. The UI Theme page does not have a button to restore the default setting.

Change the visual properties of an element style

You can modify the visual properties of any element style in the currently loaded Styles Library. You modify properties by setting overrides on the **Element style overrides** dialog box for the style you want to change.

In the **Element style overrides** dialog box, you can:

- Modify the appearance of text by setting overrides for the text font, text size, text style, text color, and blinking.
- Modify the appearance of graphic fill by setting overrides for fill color and blinking.
- Override the appearance of the line pattern, weight, color, and blinking.
- Override the appearance of the outline line pattern, weight, color, and blinking.
- Preview the appearance of an element style.
- Reset element style visual properties to their default values.

To show the Element style overrides dialog box

1. Navigate to the Style Library configuration dialog.

   The navigation path will differ according to your HMI/SCADA software. Typically, the path will be through menu items to configure your project or application components.
2. Click the tab for the type of element style you want to modify.
3. Click the edit icon for the desired style.

   The **Element style overrides** dialog box includes the following:
   - The style name.
   - The preview area at the top right showing what the element style looks like with its current settings.
   - The **Reset to default** button returns the element style to its default values.
   - Property tabs where you can set values for each property defined in the selected element style.

Override element style text properties

You can modify an element style’s text visual properties by setting alternative values for text font, text color, text style, and blink rate.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

To change the appearance of text in an *element style*

1. Navigate to the Style Library configuration dialog.

   The navigation path will differ according to your HMI/SCADA software. Typically, the path will be through menu items to configure your project or application components.
2. Select the edit icon for an element style to show the **Element Style Override** window.
3. Click the **Text** tab.
4. To change the font, select **Font Override**, click the browse button, and select a font from the **Font** dialog box.
5. To override the font color:
   a. Select **Font Color Override**.
   b. Click the color box.
   c. Select a color from the **Select Font Color** dialog box.
6. To override the text blink behavior:
   a. Select **Blink**.
   b. Select a blinking speed from the **Speed** list.
   c. Click the color box to show the **Select Blink Color** dialog box.
   d. Select the color, gradient, pattern, and texture for the blink style.
7. Click **OK**.

Override element style fill properties

You can modify an element style's fill visual properties by setting alternative values for fill color and blink rate.

To override the fill appearance of an element style

1. Navigate to the Style Library configuration dialog.

   The navigation path will differ according to your HMI/SCADA software. Typically, the path will be through menu items to configure your project or application components.
2. Select the edit icon for an element style to show the **Element Style Override** window.
3. Click the **Fill** tab.
4. To override the fill style:
   a. Select **Fill Color Override**.
   b. Click the color box.
   c. Select a style from the **Select Fill Color** dialog box.
5. To override the fill blink behavior:
   a. Select **Blink**.
   b. Select a blink speed from the **Speed** list.
   c. Click the color box.
   d. Select a style from the **Select Fill Color** dialog box.
6. Click **OK**.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Override element style line properties

You can modify an element style's line visual properties by setting alternative values for line color, line pattern, and line weight

To override the line appearance of an element style

1. Navigate to the Style Library configuration dialog.

   The navigation path will differ according to your HMI/SCADA software. Typically, the path will be through menu items to configure your project or application components.
2. Select the edit icon for an element style to show the **Element Style Override** window.
3. Click the **Line** tab.
4. To override the line pattern, select **Line Pattern Override** and select a line pattern from the adjacent list.
5. To override line weight, select **Line Weight Override** and enter a new line weight in the adjacent box.
6. To override line color properties:
   a. Select **Line Color Override**.
   b. Click the color box.
   c. Select a color style from the **Select Line Color** dialog box.
7. To override the line blink behavior:
   a. Select **Blink**.
   b. Select a blinking speed from the **Speed** list.
   c. Click the color box.
   d. Select a style from the **Select Blink Color** dialog box.
8. Click **OK**.

Override element style outline properties

You can modify an element style's outline visual properties by setting alternative values for text font, text color, text style, and blink rate.

To override the outline appearance of an element style

1. Navigate to the Style Library configuration dialog.

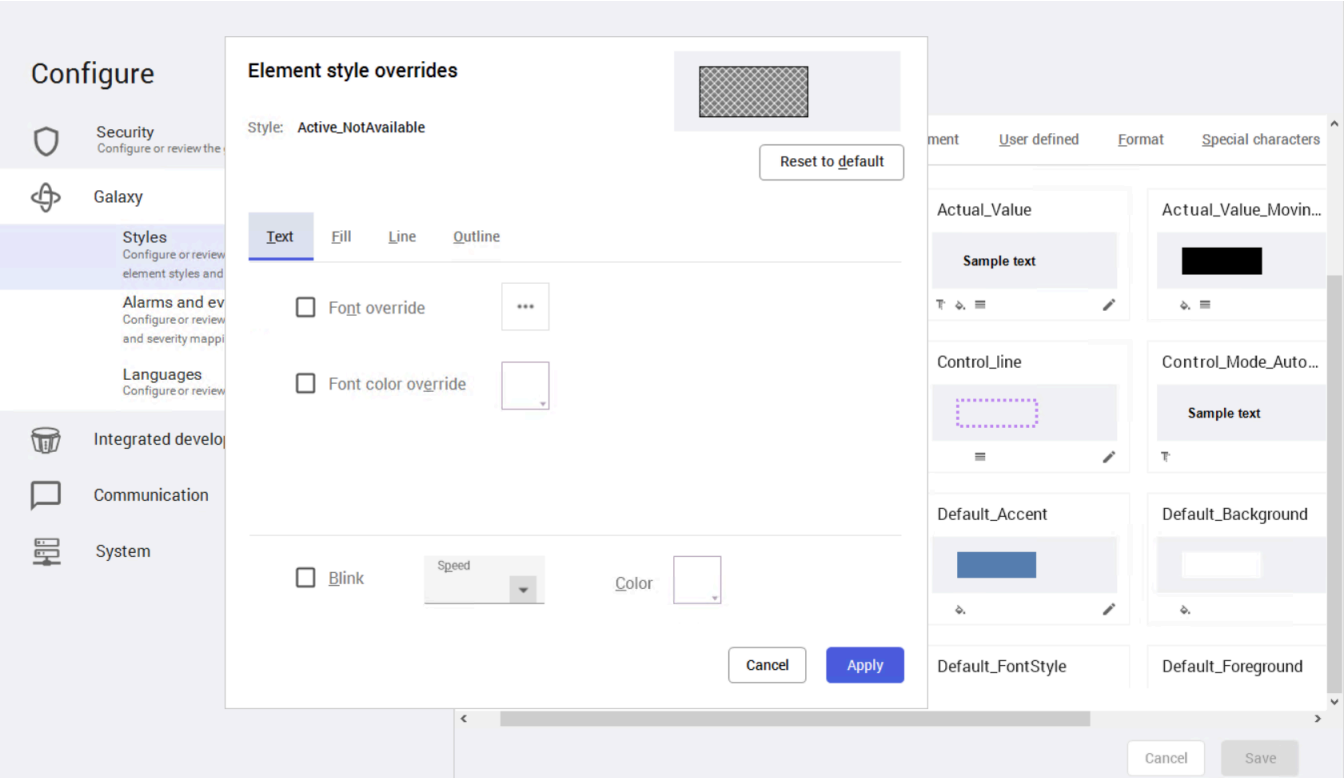
   The navigation path will differ according to your HMI/SCADA software. Typically, the path will be through menu items to configure your project or application components.
2. Select the edit icon for an element style to show the **Element Style Override** window.
3. Click the **Outline** tab.
4. Select **Show Outline.**
5. To set the line pattern, select **Line Pattern** and select a line pattern from the adjacent list.
6. To set the line weight, select **Line Weight** and type a line weight in the adjacent box.
7. To set the line style:
   a. Click the color box next to **Line Color.**
   b. Select a style from the **Select Line Color** dialog box.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

8. To set the line blink behavior:
   a. Select **Blink.**
   b. Select a blinking speed from the **Speed** list.
   c. Click the color box.
   d. Select a blink style from the **Select Blink Color** dialog box.

Preview an element style

The **Preview** area shows the appearance of an element style's current assigned visual styles.



To preview an element style

1. Navigate to the Style Library configuration dialog.

   The navigation path will differ according to your HMI/SCADA software. Typically, the path will be through menu items to configure your project or application components.
2. Select the **Styles** tab.
3. Select an element style to show the **Element Style Overrides** window.

   The **Preview** field updates to show the appearance of the selected element style.

Reset an element style to its default values

You can reset an element style to its original default visual styles.

Resetting an element style resets visual styles to their original default styles, not to any previous override settings.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

To reset an element style

1. Navigate to the Style Library configuration dialog.

   The navigation path will differ according to your HMI/SCADA software. Typically, the path will be through menu items to configure your project or application components.
2. Select the **Element Styles** tab.
3. Select one or more element styles from the **Element Style Overrides** list.
4. Click **Reset to Default**. The selected element style properties are reset to their default styles.

Change the visual properties of user-defined element styles

The Style Library includes a set of 25 user-defined element styles. User-defined element styles appear towards the bottom of the list of the **Element Style Overrides** field and are named User_Defined_01 to User_Defined_25.

All visual properties of user-defined element styles are initially set to default values. Visual properties can be individually configured for each user-defined element style by setting overrides for text, fill, line, and outline as other predefined element styles.

Apply element styles to elements

You can apply element styles to one or more graphic elements. Unlike setting element style overrides that change the appearance of an element style's properties, applying an element style to a graphic element overrides the element's native properties.

Applying element styles to graphic elements can help standardize the appearance of those elements and show the current state of an object represented by a graphic. For more information, see Change the visual properties of an element style.

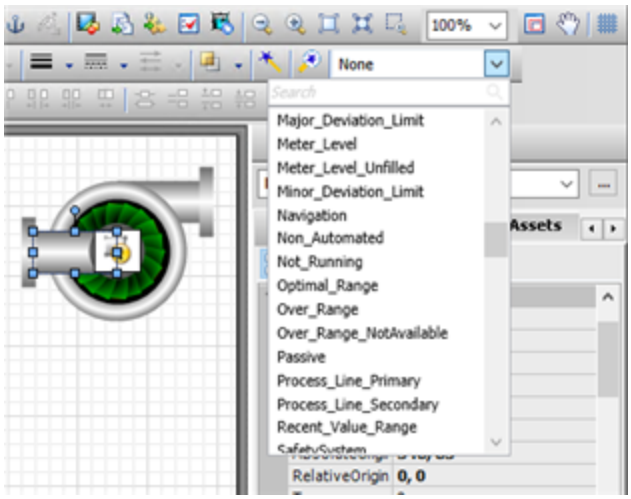
Element style list

The Industrial Graphics Editor menu bar contains an **Element Style** list to select an element style and apply it to a selected element of a graphic.

To apply an element style to a graphic element

1. Open the graphic in the Industrials Graphic Editor.
2. Select one or more elements from the graphic.
3. Select an element style from the **Element Styles** list to apply to the selected elements.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.



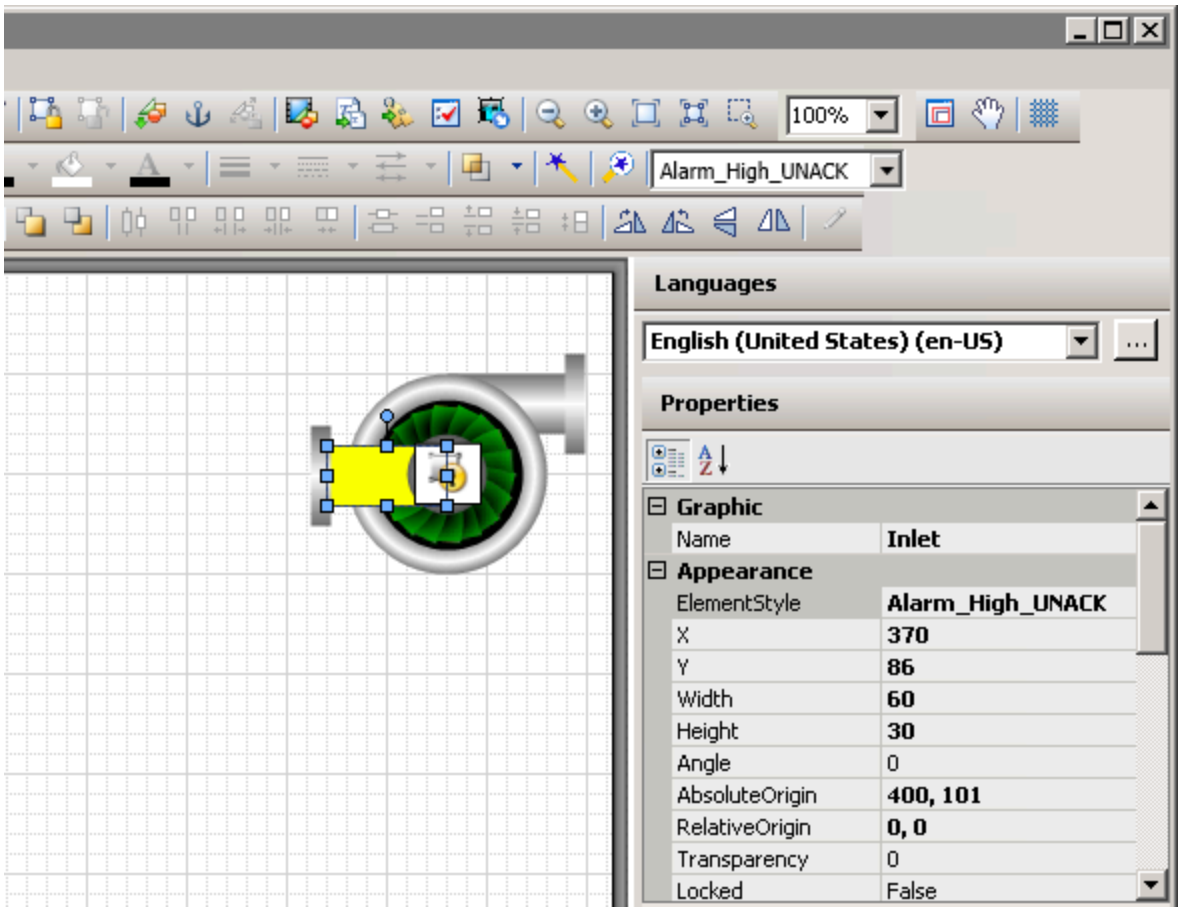
Properties grid

The Industrial Graphics Editor **Properties** view contains an **ElementStyle** Appearance item to select an element style and apply it to a selected element of a graphic.

To apply an element style from the properties editor

1. Open the graphic in the Industrial Graphic Editor.
2. Select one or more elements from the graphic.
3. In the **Appearance** category of the **Properties** Editor, select an element style from the **Element Style** list.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.



Format painter

You can use the Format Painter in the Industrial Graphics Editor to copy an element style from one graphic element to another.

To apply an element style using format painter

1. Open a graphic in the Industrial Graphics Editor.
2. Select the element with the element style you want to copy.
3. On the **Edit** menu, click **Format Painter**. The pointer appears as the Format Painter cursor.
4. Select the elements you want to apply the element style to. The element style is applied to the selected elements.

Clear an element style

When an element style is applied to an element, you cannot edit the element's styles that are controlled by the applied element style. However, you can clear the application of the element style so that all of the styles can be edited.

To clear an element style

1. Select the element.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

2. Select **None** in the element style list.

Set an element style as a default for a canvas

You can select an element style at the canvas level of the Industrial Graphics Editor. The selected element style is applied to any graphic element or groups that you create on the canvas.

Apply element styles to groups of elements

You can apply an element style on a group of elements in the same way that you apply an element style to a single graphic element. To enable group support for element styles, set the group's run-time behavior to TreatAsIcon.

Set run-time behavior for a group to TreatAsIcon

To apply an element style to a graphic element group, set the group's **TreatAsIcon** property to True. Otherwise, the element style lists are disabled when an element group is selected.

To set the TreatAsIcon property to true

1. Select the element group to which the element style will be applied.
2. On the **Properties** menu, click **Run-time Behavior** and click **TreatAsIcon**.
3. Select **True** from the drop-down list.

Element style behavior with a group of elements

- The element style applied to a group has higher precedence than the property styles applied to individual graphic elements in the group.
- If the element style applied to a group of elements has undefined property styles, then the element continues to use its element style or element-level settings for undefined property styles.
- If the element style that is applied to a group of elements has defined property styles, then those property styles override the property styles defined at the element level for elements in the group.
- An element style cannot be applied to a nested element group.
- If you add an element to a group that has a group-level element style applied, the group element style is applied to it.

Use element styles to configure an animation

You can configure an element or a group of elements with a:

- Boolean animation that applies element styles based on a binary True/False condition.
- Truth table animation that applies element styles based on a range of possible values.
- Bit State animation that applies element styles based on a bit state based conditions.

The truth table animation that applies element styles:

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

- Associates expressions of any data type supported by Application Server or InTouch to an element style.
- Defines as many conditions as required and applies a separate element style for each condition
- Defines the conditions to apply an element style by specifying a comparison operator (=, >, >=, <, <=) and a breakpoint, which itself can be a value, an attribute reference, or an expression.
- Arranges conditions in the order that element styles are processed.

Use element styles to configure a boolean animation

You can configure an element or a group of elements with a Boolean animation that uses only two element styles.

To configure an element or a group of elements with an element style that uses boolean animation

1. Open the graphic in the Industrial Graphics Editor.
2. Select the element or element group.
3. On the **Special** menu, click **Edit Animations**. The **Edit Animations** dialog box appears.
4. Click the **Add** icon and select **Element Style**. The Element Style animation is added to the Animation list and the **Element Style** state selection panel appears.
5. Click the **Boolean** button. The **Boolean Element Style** configuration panel appears.
6. In the **Boolean** text box, enter a Boolean numeric value, attribute reference, or an expression.
7. Clear **ElementStyle** in the **True, 1, On** area or **False, 0, Off** area if you don't want a different element style for the true or false condition than the default element style that is shown in the **Element Style** list.
8. In the **True, 1, On** area, select the element style to use when the expression is true.
9. In the **False, 0, Off** area, select the element style to use when the expression is false.
10. Click **OK**.

Use element styles to configure a truth table animation

You can configure an element or a group of elements with a Truth Table animation that can select multiple element styles based on a set of evaluated values or expressions.

Truth Table animation is typically used to set element styles for the different states of an object. For example, you can set Truth Table conditions to show different element styles that represent the following alarm conditions:

- When the attribute TankLevel_001.PV is 0 then no element style is applied.
- When the attribute TankLevel_001.PV is less than 20, then the element style is Alarm_Minor_Dev.
- When the attribute TankLevel_001.PV is greater than the attribute Standards.TankMax then the element style is Alarm_Major_Dev.

To configure an element or a group of elements with an element style that uses truth table animation

1. Open the graphic in the IDE Industrial Graphics Editor.
2. Select the element or group.
3. On the **Special** menu, click **Edit Animations**. The **Edit Animations** dialog box appears.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

4. Click the **Add** icon and select **Element Style**. The Element Style animation is added to the Animation list and the **Element Style** state selection panel appears.
5. Click the **Truth Table** button. The **Truth Table Element Style** configuration panel appears. The element style that is applied to the element is shown in the **Element Style** list at the bottom of the panel.
6. In the **Expression Or Reference** area:
   - Select the data type of the expression from the list.
   - Type a value, attribute reference or expression in the text box.
7. If the data type of the expression is string or internationalized string, you can specify to ignore the case by selecting **Ignore Case.**
8. In the **Truth Table**, select the **Element Style** check box and select the element style for one of the conditions to be defined in the truth table.
9. In the **Operator** column, select a comparison operator.
10. In the **Value or Expression** column, type a value, attribute reference, or expression.
11. To add other conditions:
    a. Click the **Add** icon. An additional condition is added to the truth table.
    b. Select the **Element Style** check box, select the element style for the condition, select an operator, and enter the condition value or expression.
12. After adding all truth table conditions, click **OK**.

Delete a condition from an animation truth table

You can delete a condition from an animation Truth Table to remove the associated element style from the animation.

To delete a condition from a truth table animation that uses element styles

1. Open the **Edit Animations** dialog box, **Truth Table Element Style** panel.
2. Select the condition you want to delete.
3. Click the **Remove** icon. The condition is removed.

Change the processing order of element styles in a truth table animation

You can change the processing order of element styles by moving the conditions up or down in the Truth Table list. The element style at the top of the Truth Table list is processed first. The remaining element styles are processed in order based on their position from the top of the list.

To change the processing order of element style conditions

1. Open the **Edit Animations** dialog box, **Truth Table Element Style** panel.
2. Select the condition you want to move up or down the condition list in order for it to be processed sooner or later.

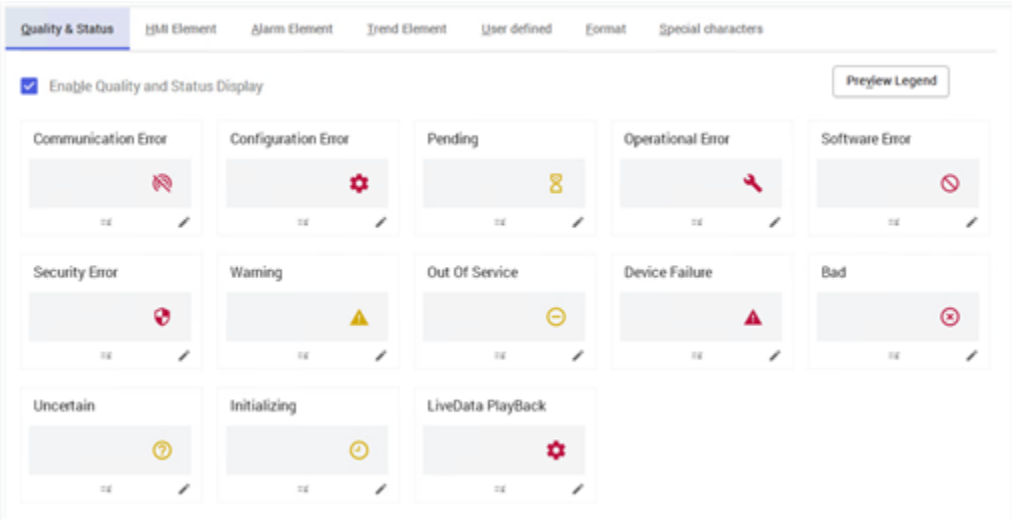
© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

3. Click the:
    - **Arrow up** icon to move the condition up in the truth table.
    - **Arrow down** icon to move the condition down in the truth table.

About the quality and status tab

Data quality and status can be represented by a set of icons that appear in a running ViewApp. The physical appearance of icons indicates current quality and status of the data associated with an icon. The **Quality and Status** tab shows a list of status style overrides that can be modified to set the visual characteristics of quality and status states at runtime.

Quality and Status (QS) indicators are listed by order of priority (left to right, with the first row listing the highest priority indicators). At runtime, if more than one QS indicator is active, only the highest priority indicator is shown.



The appearance of text, fill, line, outline, blink can be set as overrides of the following **Quality and Status** icons:

- Communication Error (highest priority)
- Configuration Error
- Pending
- Operational Error
- Software Error
- Security Error
- Warning
- Out of Service
- Device Failure
- Bad
- Uncertain
- Initializing
- LiveData Playback (lowest priority)

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Show data status

To show a specified status or quality at runtime, you can:

- Use a Status element that shows an icon. It indicates the status or quality of specified attributes directly or those used indirectly in elements.
- Change the appearance of animated elements based on the status and quality of AVEVA attributes they use.

Show quality and status with the status element

The Status element cannot monitor attributes of:

- Elements that are not in the same hierarchy level in the Elements List.
- Elements that use the attributes in scripts.
- Elements that are invisible at runtime.

Show quality and status by overriding the appearance of an animation

You can override the appearance of an animation depending on its configured attributes by:

- Overriding the animation or changing the appearance of the element.
- Drawing an outline around the element.

Overriding the appearance of animations also applies to:

- Elements contained in groups.
- Elements in graphics embedded in other graphics.

Overriding the appearance of animations does not apply to:

- Elements that use the monitored attribute in scripts.
- Elements that are invisible at runtime.

For more information, see *Overriding Element Appearance Depending on Quality and Status of its Attributes* in the Industrial Graphics Editor help.

Quality and status styles

You can configure your graphic to show non-good status and quality of attributes in different ways:

- A status element shows a specific icon depending on the quality and status of configured attributes or elements. For more information see Use status elements.
- The text, fill, or line appearance of elements can be overridden depending on the quality and status of the attributes they reference.
- Elements are drawn with an outline depending on the quality and status of the attributes they reference.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Use status elements

Status elements show a specified graphic depending on the quality and status of:

- Attributes and tags configured for specific animated elements.
- One or more specified attributes or tags.

You can assign status elements to an animation in three steps:

1. Draw the status element on the canvas.
2. Associate the status element with animated elements on the canvas and/or attributes that provide the quality and status data to be monitored.
3. If needed, configure the appearance of the status element.

Draw the status element on the canvas

You can easily place a status element on the canvas to show an icon that indicates quality and status of attributes or tags contained in selected animated elements and/or specified attributes and tags.

You do this as you would with any other element. For more information, see *Drawing and Dragging Elements* in the Industrial Graphic Editor help.

Configure the status element

You can associate the status element with:

- Animated elements that use attributes and tags that provide the quality and status that is to be monitored.
- Attributes and tags that provide the quality and status to be monitored.

In both cases, the appearance is set by the settings in your application. For example, in AVEVA ViewApps it is set in the **Quality and Status** tab of the **Configure Galaxy Style Library** dialog box.

For more information on how to configure this animation, see *Configuring Animation for a Status Element* in the Industrial Graphic Editor help.

Set the appearance of a status element

You can set the appearance of a status element depending on the quality and status of its referenced attributes and tags and/or attributes and tags used in its referenced elements.

You can also preview the appearance of a status element. For more information, see Previewing all Status Appearances.

You can reset the appearance of a status element to its default. For more information, see Reset an override appearance to its default.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Override element appearance depending on quality and status of its attributes

You can configure any animated element to appear differently depending on the quality and status of its associated attributes and tags if your HMI application supports it.

For animated elements, you can:

- Override the appearance of the text font, style, and blinking.
- Override the appearance of the fill style and blinking.
- Override the appearance of the line style, weight, pattern, and blinking.
- Preview all status appearances in one dialog box.
- Reset the status appearances to their defaults.
- Use an outline to indicate a specified status or quality.

Note: Instead of overriding the appearance of elements on the canvas, you can use a status element. The status element shows an icon representing quality and status of monitored attributes and tags.

Override the text appearance of elements to indicate non-good status or quality

If your application supports it, you can configure the text appearance of animated elements with attributes that have non-good status or quality are overridden with a specific text appearance.

Override the fill appearance of elements to indicate non-good status or quality

If your HMI application supports it, you can configure the fill appearance of animated elements with attributes that have non-good status or quality are overridden with a specific fill appearance.

Override the line appearance of elements to indicate non-good status or quality

You can configure a Galaxy so that the line appearance of animated elements with attributes that have non-good status or quality are overridden with a specific line appearance.

To override the line appearance of elements specified by a status element

1. On the IDE ribbon, select **Galaxy**, then select **Configure**.
2. Select **Galaxy**, then **Styles**. The **Style Library** opens.
3. Select the **Quality and Status** tab.
4. Select **Enable Quality and Status Display**.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

5. Select the edit icon for a status or quality.
6. Click the **Line** tab.
7. To override the line pattern, select the **Line Pattern** checkbox and select a line pattern from the adjacent list.
8. To override the line weight, select the **Line Weight** checkbox and type a new line weight in the adjacent box. Alternatively, you can increment or decrement the line weight by clicking the adjacent boxes.
9. To override the line color:
    a. Select **Line Color** checkbox.
    b. Click the color box.
    c. Select a line color from the **Select Line Color** dialog box. For more information, see *Setting Style* in the Industrial Graphic Editor help.
10. To override the line blink behavior:
    a. Select **Blink**.
    b. Select a blinking speed from the **Speed** list (Slow, Medium, or Fast).
    c. Click the color box.
    d. Select a line blink color from the **Select Blink Color** dialog box. For more information, see *Setting Style* in the Industrial Graphic Editor help.
11. Click **Apply**.

Add outlines to elements to indicate non-good status or quality

You can configure the Galaxy so that animated elements with attributes that have non-good status or quality are shown with an outline.

To add outlines to elements to indicate non-good status or quality

1. On the IDE ribbon, select **Galaxy**, then select **Configure**.
2. Select **Galaxy**, then **Styles**. The **Style Library** opens.
3. Select the **Quality and Status** tab.
4. Select **Enable Quality and Status Display**.
5. Click the **Outline (Ol)** tab.
6. Select the edit icon for a status or quality.
7. Select **Show Outline**.
8. To set the line pattern, select **Line Pattern** and select a line pattern from the dropdown list.
9. To set the line weight, select **Line Weight** and enter a line weight. Alternatively, you can increment or decrement the line weight by clicking the adjacent boxes.
10. To set the line style:
    a. Click the color box next to **Line Color**.
    b. Select a line color from the **Select Line Color** dialog box. For more information, see *Setting Style* in the Industrial Graphic Editor help.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

11. To set the line blink behavior:
    a. Select **Blink**.
    b. Select a blinking speed from the **Speed** list (Slow, Medium, or Fast).
    c. Click the color box.
    d. Select a line blink color from the **Select Blink Color** dialog box. For more information, see *Setting Style* in the Industrial Graphic Editor help.
12. Click **Apply**.

Preview all status appearances

You can preview the appearance of all status overrides by showing the **Status Legend** dialog box.

To preview all override appearances

1. On the IDE ribbon, select **Galaxy**, then select **Configure**.
2. Select **Galaxy**, then **Styles**. The **Style Library** opens.
3. Select the **Quality and Status** tab.
4. Click **Preview Legend**. The **Status Legend** dialog box appears.
5. Click **Close**.

Reset an override appearance to its default

For any status, you can reset the default appearance:

- Text, fill, and line overrides.
- Outline settings.
- Status element settings.

To reset a status or quality to its default appearance

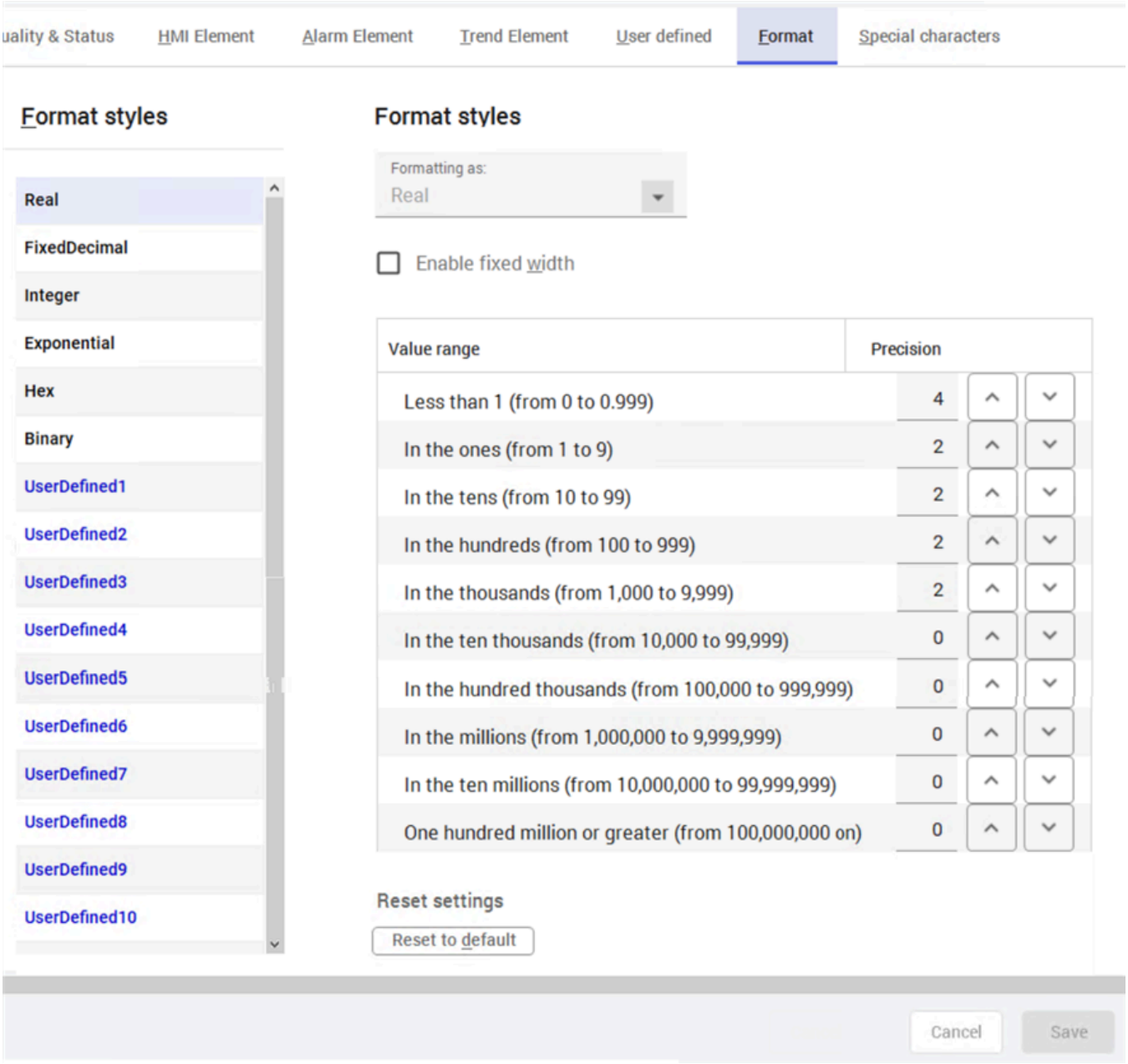
1. On the IDE ribbon, select **Galaxy**, then select **Configure**.
2. Select **Galaxy**, then **Styles**. The **Style Library** opens.
3. Select the **Quality and Status** tab.
4. Select the edit icon for a status or quality.
5. Click **Reset to default**. All text, fill, and line overrides, status element icons, and outline settings are reset to their defaults.

Format Styles tab

The **Style Library** page of the backstage includes the **Format Styles** tab. **Format Styles** provides options to individually configure Galaxy-wide styles for common types of numbers used in industrial applications.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Each global number style is assigned a unique name, which cannot be changed. A number style can be applied by name in design time and runtime for an analog data type in User Input and Value Display animations. Also, grouped elements support global number styles.



Configure a global numeric format

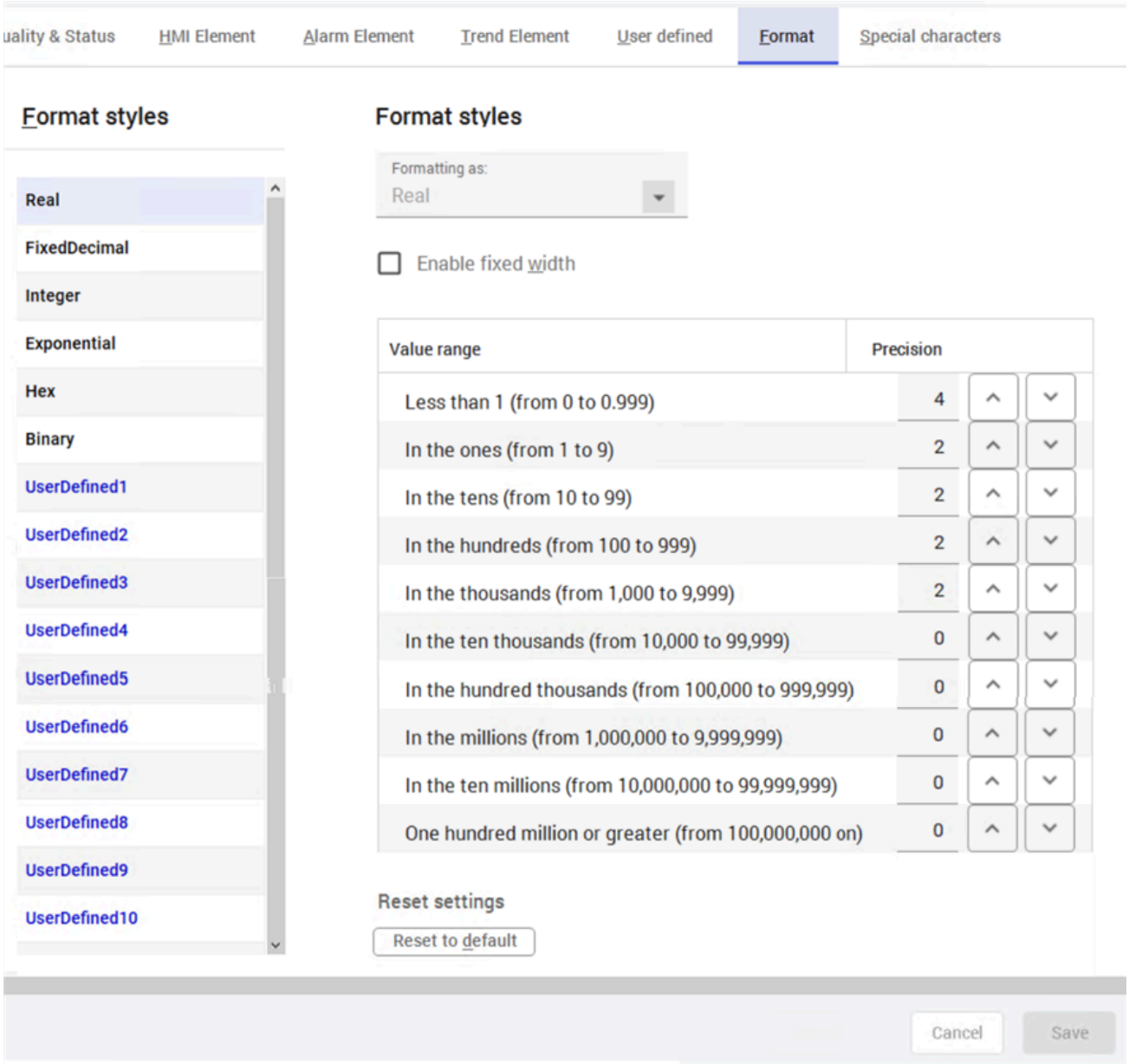
You can configure a global number style by changing the values assigned to the properties for each type of number.

To configure a global numeric format style

1. On the IDE ribbon, select **Galaxy**, then select **Configure**.
2. Select **Galaxy**, then **Styles**. The **Style Library** opens.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

3. Select the **Format** tab to show numeric format options.



4. Select a numeric format style from the **Format styles** list.

   The **Format** tab updates to show options for **Fixed Width**, **Precision**, and/or **Bits From**, or **To** based on the selected format style.
5. Update the fields shown for the selected format style.
6. Click **Save** to save your changes.

Format list

The **Format** list includes a set of styles that can be individually configured to create customized number styles. The Real number format is the default for the user defined styles.

Based on the selected number style, the following fields appear on the **Format** dialog box to change the default properties.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

- **Fixed Width**

  Appears for every number format style. When selected, the length of a number cannot exceed the text length of the text element (Text, TextBox, or Button) in design time. Numbers that exceed design time text length will show the special character specified in the **Value too large for Fixed Field**. The character is defined in the **Special character** tab of the **Styles** dialog.

  For the Real number style, the length of the fractional part of the number is truncated to fit the design time length. If the length of the number is still too large after removing the entire fractional part, then the number will show the special character specified in the **Value too large for Fixed Field**.

- **Precision**

  Appears for the **Real**, **FixedDecimal, and Exponential** number styles. **Precision** sets the possible number of digits in the fractional part of a number to the right of the decimal point and can be set from 0 to 8.

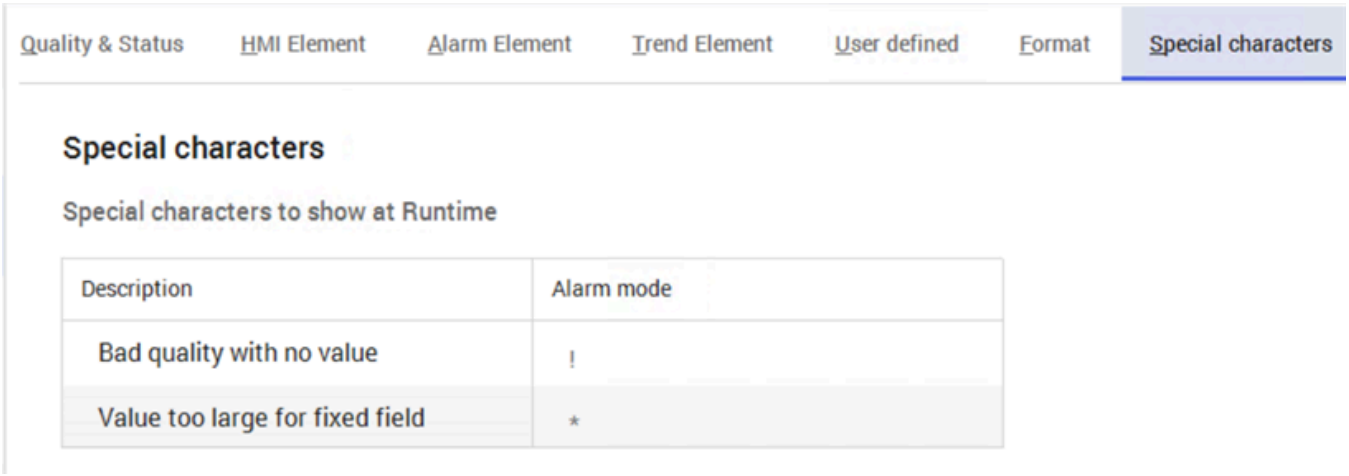
- **Bits From** and **To**

  Appear for the **Hex** and **Binary** number formats. **Bits From** sets the starting bit position (0-31) of a hex or binary number shown during runtime. **To** sets the ending bit position of a hex or binary number shown during runtime.

Special characters styles

The **Format** dialog box includes fields to specify characters that appear during run time that indicate bad quality data or data that exceeds a fixed width field.

Industrial Graphics and InTouch HMI graphics will display the current value when the data quality is bad but presents with a value. It will display the special character configured under Galaxy-Styles when there is no data and the data quality is bad.



User-defined global number styles

The Galaxy Style Library includes a set of 25 user-defined number styles. User-defined styles appear towards the bottom of the list of the **Format Styles** list and are named UserDefined1 to UserDefined25.

Rename user-defined global number styles

Rename a style to provide a descriptive name, to provide a group of related styles names, or to suit any other

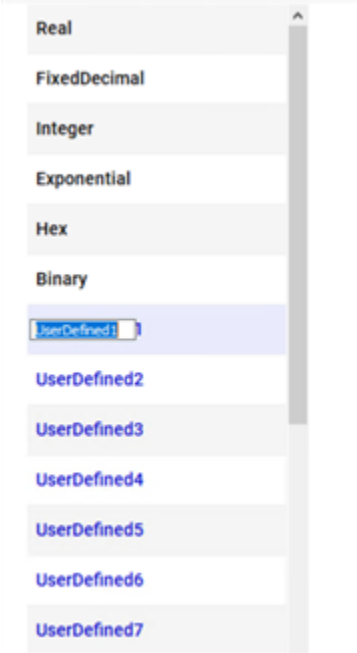
© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

specific application needs. The renamed style will appear during configuration. For example, the renamed style will appear in the Industrial Graphic Editor when configuring an animation using Number Styles.

The renamed style will function the same during runtime as it did prior to the name change.

To rename a user-defined element style

1. On the IDE ribbon, select **Galaxy**, then select **Configure**.
2. Select **Galaxy**, then **Styles**. The **Style Library** opens.
3. On the **Format** tab, navigate to the user-defined style you want to rename.
4. Right-click the UserDefined format style to select the style name and enable the text box for editing.



5. Enter a new style name.

Import and export user-defined global number styles

You can import and export your renamed user-defined Number Style as a normal part of the Galaxy Style Library. For more information, see Import and export graphics.

Set number formats by regional locales

The format of numbers varies by country. In the United States, a period represents the decimal point of an analog number and a comma is the thousand separator. In other countries, the purpose of the characters may be different. Germany uses a comma to represent the decimal point and a period to represent the thousand separator.

Industrial graphic numeric values can display thousands and decimal separators that match the numeric format of the country specified as the Home location of the computer running an HMI/SCADA application.

Numeric formatting by regional locale applies only to Industrial graphic number displays and input numbers included as part of your applications.

Numeric formatting by regional locale can be applied to an application containing:

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

- User Input animation
- Value Display animation
- Tooltip animation
- Windows Client Controls (RadioBox, ComboBox, and ListBox)
- SecuredWrite dialog
- VerifiedWrite dialog
- SignedWrite dialog

Design time considerations for numeric formats

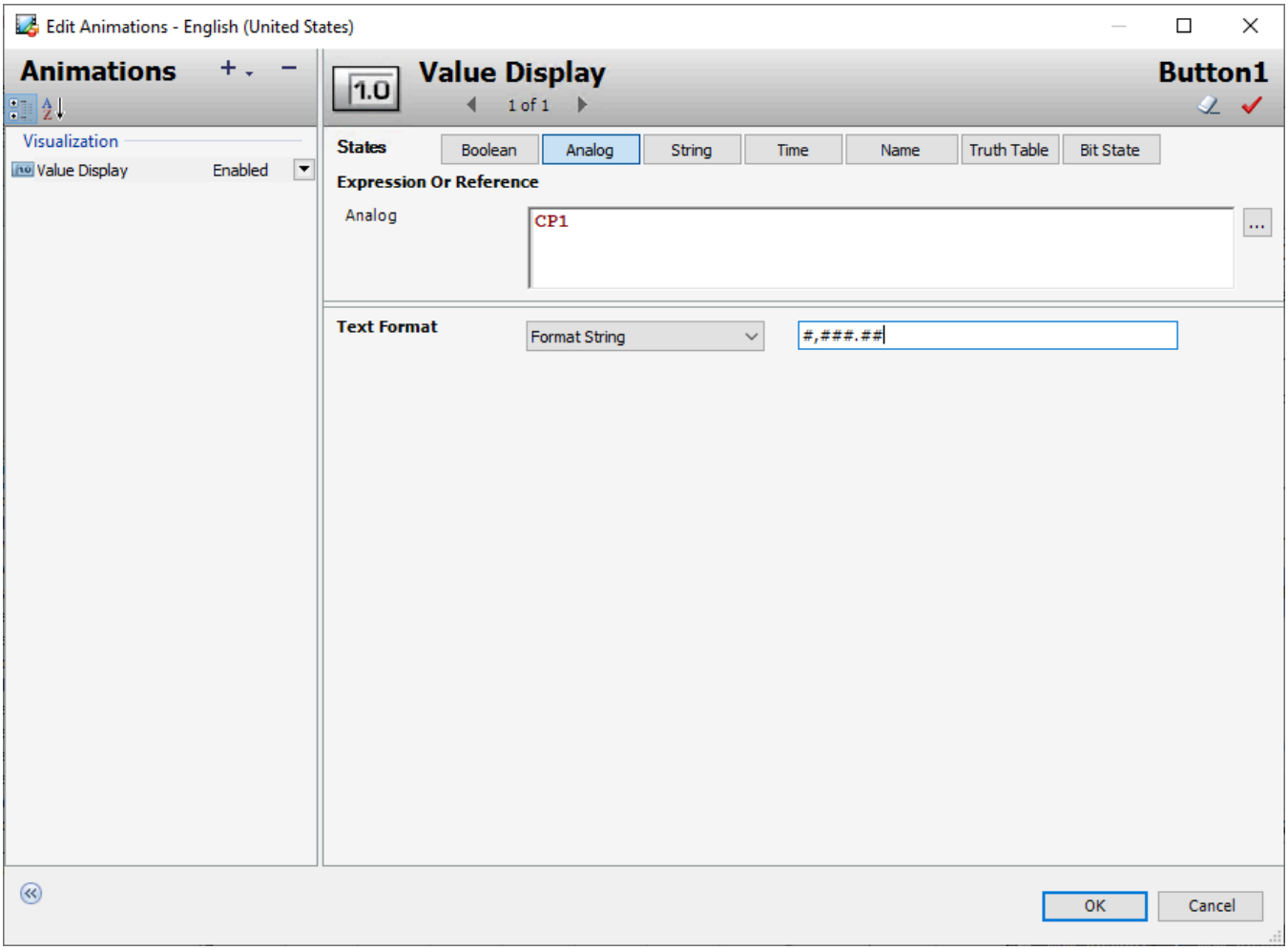
During design time, make the following preparations to show numbers in a regional locale:

- Enter numbers according to the the U.S. format in design time, e.g. #,###.##
- Set the regional locale of the computer running an HMI/SCADA application.

Enter input numbers in U.S. format

The Industrial Graphics Editor enables you to specify a number format during design time using a format string for Value Display animations.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.



A Value Display animation includes a **Text Format** field. When **Text Format** is set to **Format String**, enter a text string that represents the format of numbers shown during run time.

During design time, a numeric format string or numeric value follows the United States number format.

- The decimal point in a number is a period.
- The thousand separator in a number is a comma.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Value in String (no format)	56000456.56

Format String	User Input	Value Display	
#,###.##	56000.456,56	56000.456,56	
#,###,###.##	56.000.456,56	56.000.456,56	
####.##	56000456,56	56000456,56	
#.###,##	56000456,560000	56000456,560000	(wrong format)

Numeric Format Styles

	Value Display
Custom, Real	56000457
Custom, Fixed Decimal	56000456,56
Custom, Fixed Decimal, 4 precision	56000456,5600
Real	56000457
Fixed Decimal, 3 precision	56000456,56
Integer	56000457
Exponential, 3 precision	5,60e+007
Hex	4C559FF2
Binary	01001100010101011001111111110010

In the example above, the design-time numeric format strings within the box at the left comply with the U.S. number format. But during run time, User Input and Value Display animation show numbers in the correct format of the country specified by the computer's Regional setting.

Set the regional locale of the computer hosting the HMI/SCADA application

To enable numeric formatting by regional locale, set the region of the computer running an HMI/SCADA application to the country in which you want Industrial graphic numbers to be formatted.

The **Region** setting is accessible from the Windows Control Panel. If you want to display Industrial graphic numbers in a non-U.S. format, select the **Formats** tab and select a country in the **Format** field.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Runtime considerations for formatting numbers

The following analog number format styles support numeric formatting by regional locale during runtime:

- Custom
- Real
- FixedDecimal
- Integer
- Exponential

During runtime, numbers are entered in the format of the selected country.

The following figure shows the analog number 123456.558857 formatted to the German regional locale during runtime by the different numeric format styles:

Numeric Format Styles

	Value Display
Custom, Real	123.457
Real	123.457
Fixed Decimal	123.456,56
Integer	123.457
Exponential	1,23e+005

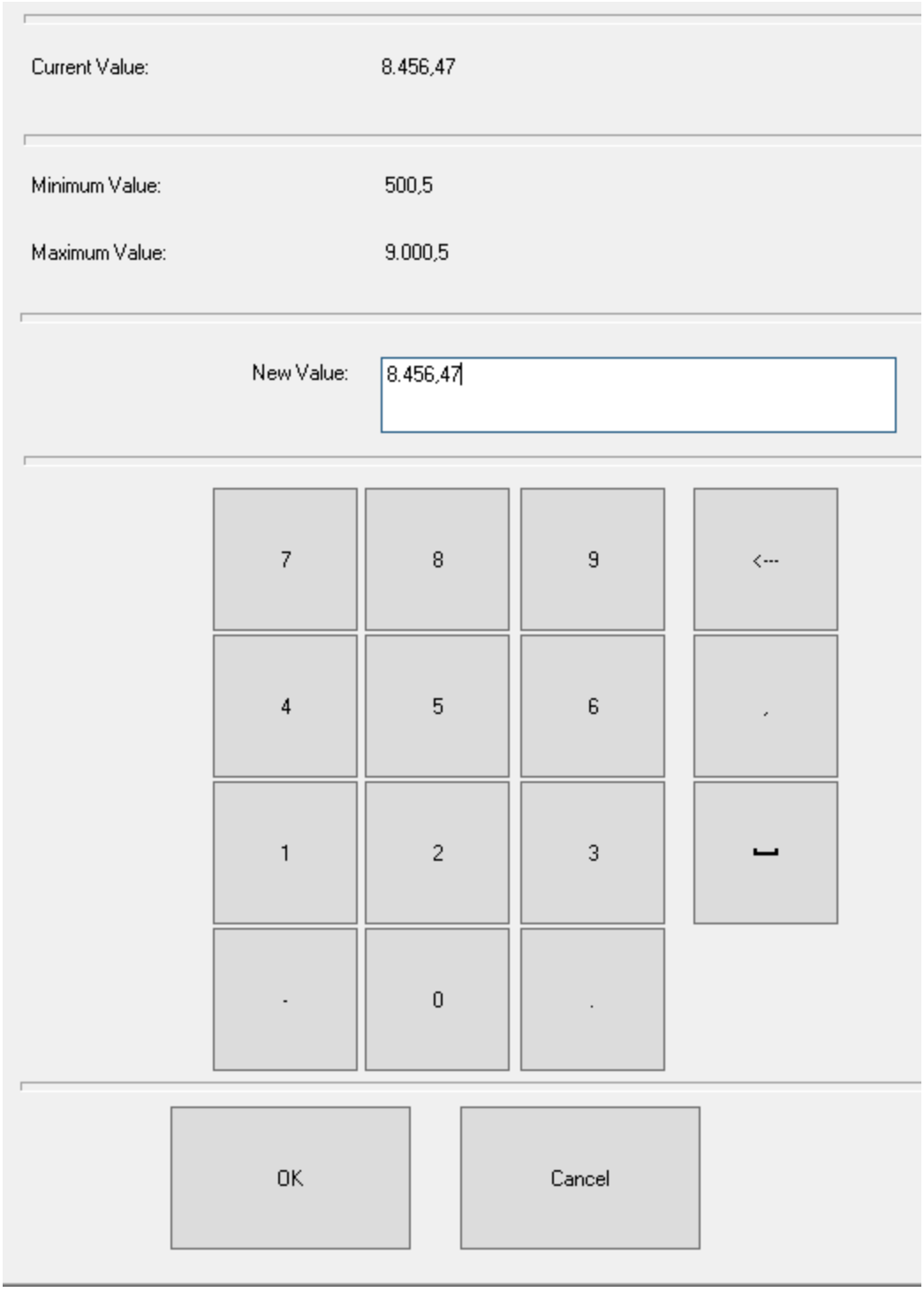
A thousand separator is not required when entering a number or specifying a format string. But, if a thousand separator is used during runtime, place it in the correct location in a specified number or with User Input animation.

The thousand separator appears in the following numeric format styles:

- Real
- Fixed Decimal
- Integer
- Custom configured as Real, Fixed Decimal, or Integer

During runtime, if your HMI/SCADA software provides software keypads, you can include comma, period, and thin space keys to enter numbers in a User Input animation data entry field that match the format of the computer's regional locale. The following illustration shows a typical software keypad. The software keypad, if supported by your HMI/SCADA software, might vary in appearance.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.



Numeric format restrictions by regional locale

There are several restrictions of numeric formatting by regional locale to consider for the visualization applications you build with your HMI/SCADA software.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Numeric strings enclosed within quotation marks

Numeric strings enclosed within quotation marks

A numeric string enclosed within quotation marks cannot be converted to the number format of another regional locale because of the ambiguity interpreting the thousand separator character.

Example:

"4000.654" in the U.S. regional setting is four thousand.

"4000.654" in the Germany regional setting is over four million.

A numeric string is not supported and cannot be converted to the number format of another regional locale.

Numbers passed as script parameters

Numbers passed as script parameters

Scripts containing the SignedWrite() function experience similar problems interpreting the thousand separator character when a numeric string is passed as a parameter within quotation marks.

Example:

SignedWrite("AO1.PV1","8.456,56","",true,1,null);

The numeric value is enclosed within quotation marks as a string data type and the comma thousand separator character is interpreted as a parameter delimiter. If the computer running a visualization application regional locale is set to Germany, the script incorrectly writes 8.46 to an attribute.

Alternative Solution:

Use a custom property with an analog data type instead of a string.

SignedWrite("AO1.PV1",CP1,"",true,1,null);

where CP1=8.456,56 is set by the user at run time.

Double-byte character languages

Double-byte character languages

Double-byte character languages like Chinese or Japanese provide narrow or wide character sets. The Windows default setting is to show narrow characters in Chinese or Japanese languages. The decimal point and digital grouping characters can be shown with a narrow double-byte character set. However, the comma or period characters cannot be shown with a wide double-byte character set.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Switch to another galaxy style library during runtime

AVEVA OMI includes the Style namespace with two attributes.

- CurrentStyle is a string that is the name of a Galaxy Style Library. CurrentStyle can be included in a script to change to another Galaxy Style Library during runtime.

  CurrentStyle is retentive. A ViewApp starts with the Galaxy Style Library that was active when the ViewApp was previously stopped.

  The CurrentStyle attribute operates in the MyViewApp.Style namespace. CurrentStyle is specified in a script with the MyViewApp.Style prefix in the form MyViewApp.Style.CurrentStyle.
- Styles.CSV is a comma-separated string that is a list of the defined galaxy styles. The value is read-only and is non-retentive. It is defined by the list of galaxy styles defined in the Galaxy Style Library. This list is configured in the IDE.

Note: The CurrentStyle and Styles.CSV attributes are available for use only in AVEVA OMI ViewApps. The CurrentStyle attribute cannot be used in an InTouch HMI application because the MyViewApp. prefix is regarded as a configuration error.

Attribute	Data Type	Read/Write	Initial Value	Description
CurrentStyle	String	Yes	Retentive	Loads the name of a Galaxy Style. During runtime, the value of CurrentStyle can be changed to the name of another Galaxy Style. The appearance of graphics in the running ViewApp are updated with the element styles of the new, active Galaxy Style.
Styles.CSV	String	Read Only	No initial value	Gets the list of available Galaxy Styles in CSV format. During runtime, shows the styles as a comma separated string. The value is automatically updated when new styles are added to the Galaxy Style

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

				Library in the IDE.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Build

This section provides information about building your application.

Configure objects

Enhance and extend derived templates and instances by adding and configuring features, attributes, scripts, and graphics. These elements can be associated with an Object Wizard in a derived template to simplify the task of creating instances.

The **Object Editor** includes a separate page for adding scripts. All other elements (graphics, attributes, Object Wizards, and external content) are added and configured from the Object Editor's main page. For an overview of the **Attributes** page, see Attributes page.

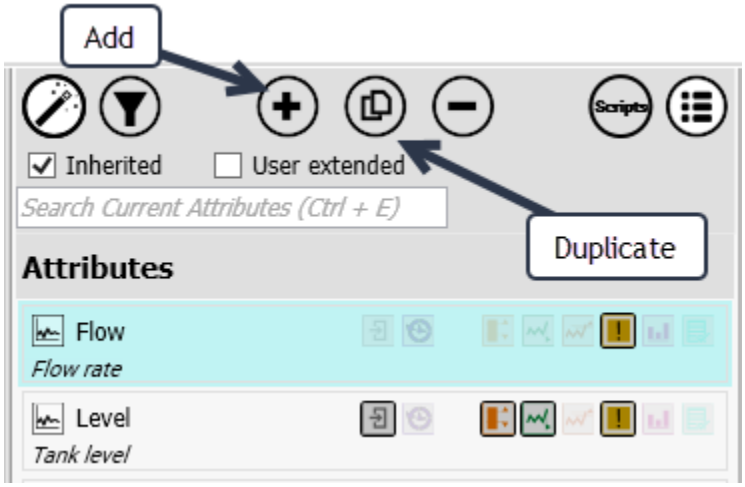
Add attributes to an object

There are two ways to add attributes to an object. These are:

1. Select the **Add Attribute** button. The first attribute you add to an object has the default name "Attribute001" and the default data type "Boolean.

   Subsequent attributes added with the **Add Attribute** button use the previous attribute's name and increments the number at the end of the name, or appends "1" at the end of the name if the previous name did not end in a number. It also retains the previous attribute's description, Data type, and Writeability.
2. To duplicate an existing attribute, select the **Duplicate Attribute** button. Selecting **Duplicate Attribute** retains any features that were configured for the selected attribute, in addition to name, description, Data type and Writeability.

   Note: When you add an attribute to a derived template, the attribute, its data type, and writeability are automatically locked in derived objects.



If attribute parameters such as initial values and security classifications are locked in the template, they cannot

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

be changed in child instances. If these parameters are unlocked in the template, the initial value and security are editable and lockable in derived templates. When unlocked in either the base or derived template, the value is editable in instances.

After you add an attribute to an instance, it appears in the **Attribute Browser** list for use with the scripting and attribute configuration functions. For more information about using the **Attribute Browser**, see Use the galaxy browser to reference objects .

In the **Attributes** page, you can define the following initial information and parameters for the attribute:

- Add a new attribute to an object.
- Name the attribute and provide a description.
- Configure its data type.

  For a Boolean data type, you can specify different text strings for the **'False' label** and **'True' label**. For example, if a Boolean attribute is associated with the status of a motor, you can specify the states as "**Stopped**" and "**Running**". Text boxes appear for you to enter these strings when you select a Boolean data type.

  These labels are also shown in the **Value** and **Limit** columns of the Alarm and Event database and InTouch AlarmView control.
- Specify the attribute writeability.
- Set initial values if the attribute is user writeable.
- Enable and set locks and security on the new attribute.
- Set whether the new attribute is an array and how many elements are in the array.

You can then add Features to the attribute. For more information, see Add features to an attribute.

Configure an attribute

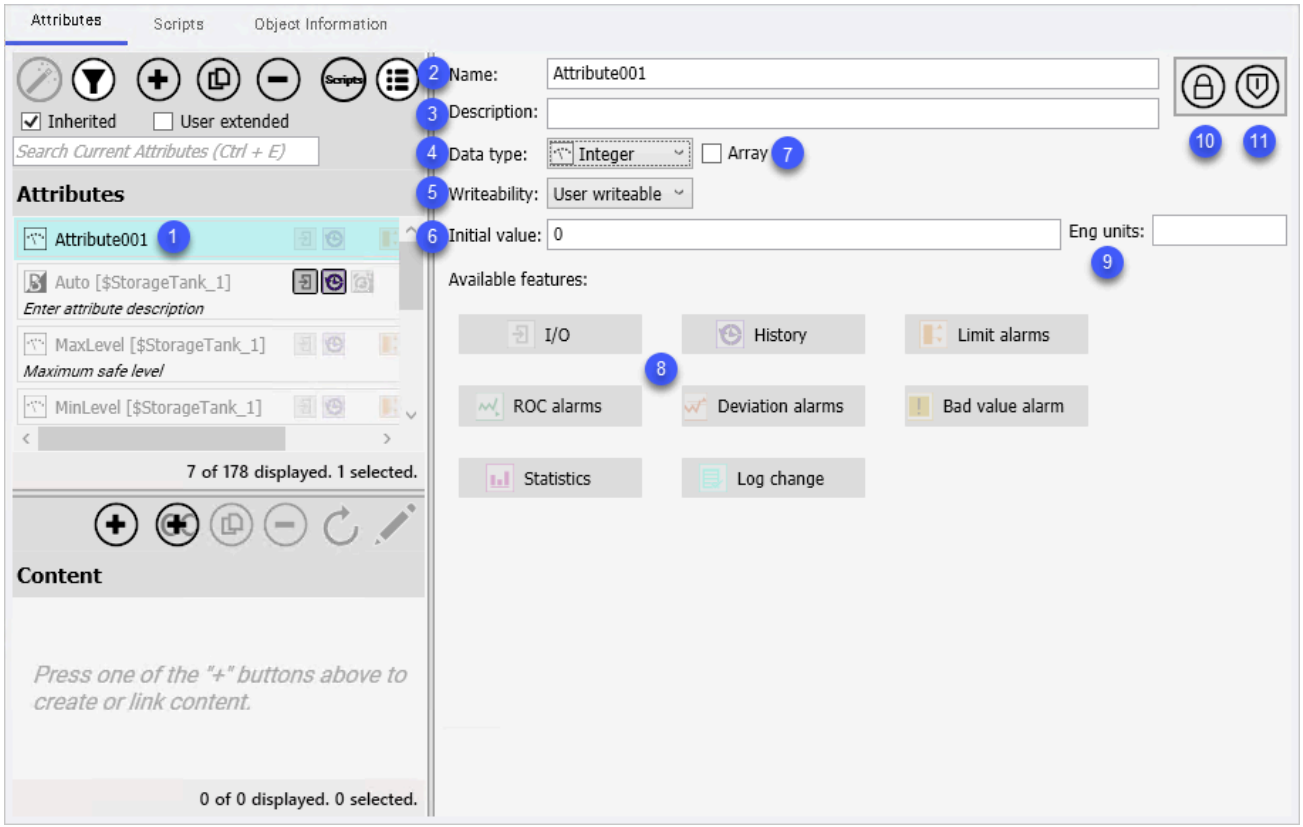
To configure attributes

1. Highlight the attribute you want to configure.
2. Edit the **Name** of the attribute.
3. Enter a **Description**.
4. Select a **Data type.** See Data type for additional information.
   - Boolean: Enter False and True labels (if not False and True).
   - Integer: Enter Initial value and Engineering units.
   - Float: Enter Initial value and Engineering units.
   - Double: Enter Initial value and Engineering units.
   - String: Enter Initial value.
   - Time: Enter Initial value.
   - ElapsedTime: Enter Initial value.
   - InternationalizedString: Enter Initial value.
   - BigString: Enter Initial value.
5. Select Writeability type:

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

- Calculated
- Calculated retentive
- Object writeable
- User writeable

6. Enter the Initial value for the attribute.
7. Select the checkbox if the attribute is part of an array.
8. Enter the Engineering unit type (numeric data types only).
9. Select Features and configure them. See Add features to an attribute for additional information.
10. Lock the Initial value, if you do not want users to be able to modify the value in derived objects. You can show or hide the lock icon by toggling the **Show/Hide Lock icons** button. See Lock and unlock template attributes for additional information.
11. Configure the attribute's security state. You can show or hide the security icon by toggling the **Show/Hide Security icons** button. See Set object security for additional information.



Data type definitions

Application Server data types are defined below.

Data Type	Definition
Boolean	The Boolean data type provides a choice between two values, typically True and False. Other common Boolean values are On and Off, Enabled and Disabled,

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

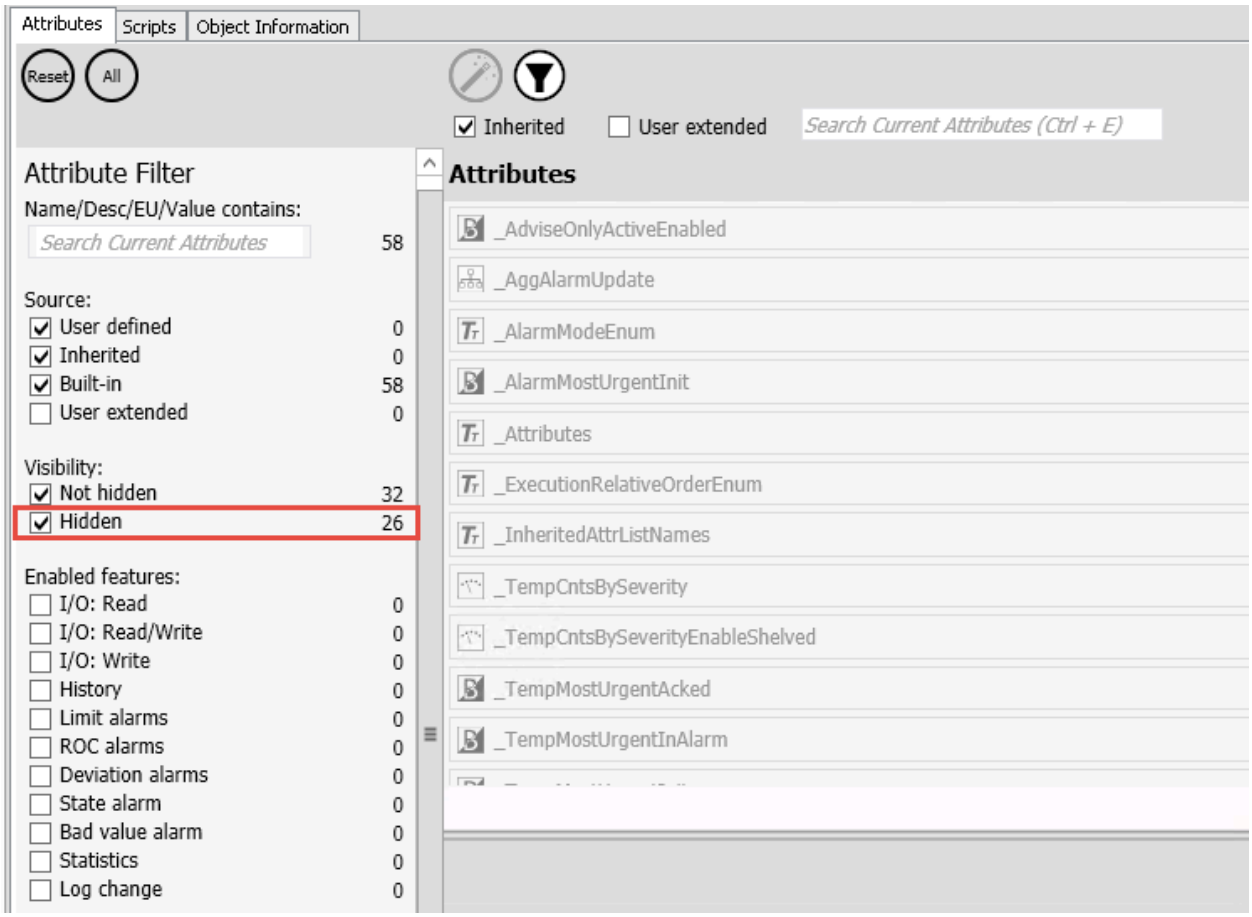
	and 0 and 1.
Integer	The Integer data type is a 32-bit signed number, and is used for positive and negative whole numbers. Valid values range from -2147483648 to 2147483647 (-2,147,483,648 to +2,147,483,647). Note that commas are invalid characters and cannot be contained within the values.
Float	The Float data type is a 32-bit floating point number, and is used for positive and negative decimal values. It can contain up to seven significant digits. The range of values is $\pm 1.18\times10^{-38}$ to $\pm 3.4\times10^{38}$.
Double	The Double data type is a 64-bit floating point number, and is used for positive and negative decimal values. It can contain up to 15 significant digits. The range of values is $\pm 2.23\times10^{-308}$ to $\pm 1.80\times10^{308}$.
String	The String data type is used for strings of ASCII characters, such as those that can be entered by a standard US or international keyboard. Strings can be up to 1024 characters long.
Time	The Time data type is used to encode the date and time. It uses the operating system locale to format the appearance of the date. For example, on a U.S.-locale operating system, the date is shown as MM/DD/YYYY. In operating systems for other locales, the date is shown as DD/MM/YYYY. Time is shown as HH:MM:SS. AM or PM is shown when time is shown in 12 hour format.
ElapsedTime	The ElapsedTime data type is used to measure time from a start time to an end time. Elapsed time is shown as HH:MM:SS.fffffff, where fffffff represents fractions of a second (potentially to 1/10,000,000 of a second).
InternationalizedString	The InternationalizedString data type is used for text that will be localized.
BigString	A BigString data type can be used to hold very long character strings, such as the contents of json, csv, or xml files. The BigString data type has a virtually unlimited length (more than two billion characters).

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Attribute naming conventions

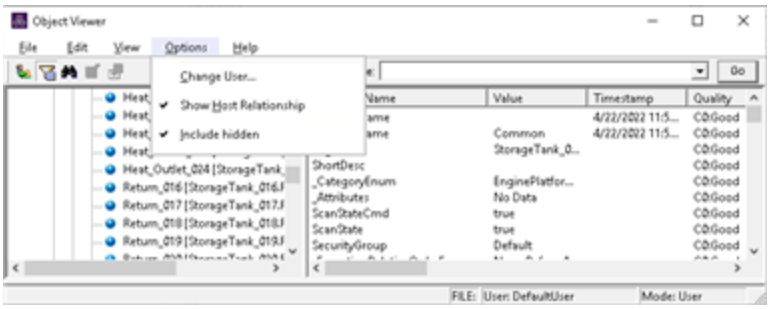
Attribute names can be from one to 329 characters long. Valid characters are alphanumeric characters and the following special characters: . (period), _ (underscore), $, and #. The maximum name length includes extensions to an attribute that result from the addition of features, such as historization, alarms, and I/O that can be associated with an attribute. These additions are categorized as "User extended" attributes. See Add features to an attribute for more information.

- An attribute name that starts with an underscore (_) as the first character of the name is a hidden attribute. By default, hidden attributes are not shown in the **Attribute Editor**, the **Properties>Attributes** dialog box, or the **Object Viewer**.
  - To view hidden attributes in the **Attribute Editor**, select the **Hidden** filter option.



  - To view hidden attributes in the **Properties** dialog box, select **Include Hidden**.
  - To view hidden attributes in the **Object Viewer**, select **Include Hidden**.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.



Note: We recommend that you do not use hidden attributes in scripts.

- Using the word "quality" as an attribute name is not supported. "Quality" is used by InTouch HMI in a set of dotfields to show the reliability of data values assigned to an I/O tag. An attribute named "Quality" cannot be accessed through FS Gateway or InTouch due to a naming conflict.

Important: After creating an attribute, it is available for adding Features or additional attributes. If you extend an attribute you have added to an object or another attribute, and then delete or rename that attribute, all associated Features and additional attributes added to the object are lost.

To create and associate a attribute with an object

1. On the **Attributes** page of the Object Editor, click the **Add** (**+**) button. An attribute is added to the **Attributes** list.
2. Type the new attribute name and add an optional description.
3. In the **Data type** list, select the **Data type** for the new attribute. The available features and basic options change depending on your selection in the **Data type** list.
4. Set the remaining parameters as needed.

Note: For detailed information about each item on the **Attributes** page, see Attributes page.

5. Lock the values, if needed. The lock is available only when you are working with a template. If you are working with an instance, it shows the lock status for the value in the parent object.
6. Set any security you need. For more information about setting security, see Set object security.
7. Save and close the Object Editor when you are done.

Attributes and scripting

Follow these guidelines and best practices when using attributes in scripts:

- If you use **Calculated** and **Calculated retentive** attributes as counters, they must be manually initialized. For example, if you use `me.Attribute=me.Attribute+1` as a counter in a script, you must also initialize the attribute with something like `me.Attribute=1` or `me.Attribute=<some attribute value>`.
- **Calculated** attributes can be initialized in scripts with **Execution type** triggers of On Scan and Execute, but not initialized in Startup scripts.
- If you use an attribute **array** in a **startup script**, value and quality may become bad when the array is initialized in the startup script. To prevent this, set the dimension after initializing the array as shown below. This will ensure that the value is set and the attribute quality is good.

```
Me.ArrayAttribute[]="0,0,0,0,0,0,0,0,0,0";
Me.ArrayAttribute.Dimension1 = 10;
```

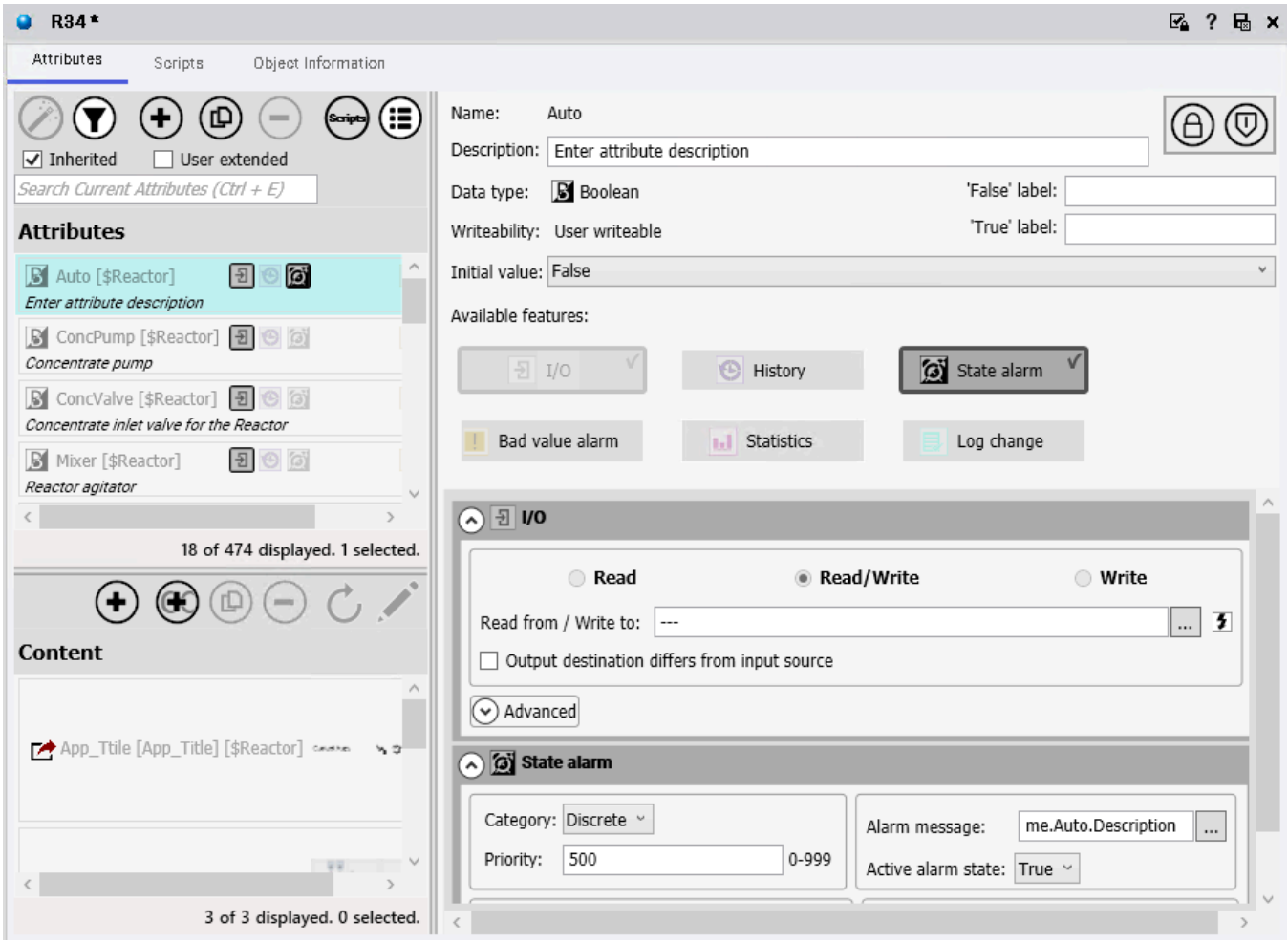
© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

- You must initialize **Calculated retentive** attributes in Startup scripts and you can initialize these attributes in On Scan and Execute scripts. A **Calculated retentive** attribute retains the attribute's current value after a computer restart, redundancy-related failover, or similar situation in which valid checkpoint data is present.

  Your Startup script should contain a statement testing the Boolean value of the Engine.StartingFromCheckpoint attribute on the object's AppEngine. If the value is TRUE, do not initialize the attribute. If the value is FALSE, initialize the attribute. For more information about Engine.StartingFromCheckpoint, see the help for the AppEngine object, available from the AppEngine object editor.
- We recommend that you **do not use hidden attributes** (attribute names that begin with an underscore) in scripts.

Add features to an attribute

The **Attributes** page allows you to configure an existing attribute for I/O, alarms, and history functionality not embedded in the original object.



Features inheritance

You can add Features to attributes that are in either derived templates or in instances. You cannot add Features

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

to attributes in Base templates. The following parent-child object characteristics also apply to Features added to objects:

- If you add a Feature to an attribute in a derived template that has objects derived from it, all child objects inherit the Feature.
- You cannot add a Feature to attributes on derived objects that duplicate parent object Features in name and type.
- You cannot add a Feature with the same name as an existing Feature.
- Renaming a Feature on an attribute in the template to which it was originally added renames all other objects derived from the template. This change happens when the template is checked in.
- You can check in a template with an attribute configured with a new Feature with the same name as an existing Feature on an attribute in a derived object. The template definition of the Feature overrides the Feature in the derived object.
- If you remove a Feature on an attribute from a template, that Feature is removed from any child object. You see the change when you check in the template.

To create and associate a Feature with an object

1. On the **Attributes** page, select an attribute from the **Attributes List**. The available Features change to the Features allowed for the attribute.
2. Click the button for the Feature you want to apply to the selected attribute. The associated parameters for each kind of Feature become available. For detailed information about each item on the **Attributes** page, see Attributes page.
3. Select the parameters for the Feature you have added. Available Features, which vary depending on the attribute's Data Type and Writeability, are the following:
   - **I/O**: For information about adding the I/O Feature, see Enable I/O for an attribute.
   - **History**: For information about adding the History Feature, see Enable history for an attribute.
   - **Limit alarms**: For information about adding a Limit alarm Feature, see Enable limit alarms for a numeric attribute.
   - **ROC alarms**: For information about adding a Rate of Change (ROC) alarm Feature, see Enable rate of change (ROC) alarms for a numeric attribute.
   - **Deviation alarms**: For information about adding a Deviation alarm Feature, see Enable deviation alarms for a numeric attribute.
   - **State alarm**: For information about adding a State alarm, see Enable a state alarm for a boolean attribute.
   - **Bad Value alarm**: For information about adding a Bad Value alarm, see Enable a bad value alarm for an attribute.
   - **Statistics**: For information about adding a Statistics Feature, see Enable statistics for a numeric attribute.
   - **Log change**: For information about adding a Log Change Feature, see Enable log change for an attribute.
4. Lock the values, if needed. The lock symbol is available only when you are working with a template. If you are working with an instance, it shows the lock condition of the value in the parent object.
5. Set any security for the attribute. For more information about setting security, see Set object security.
6. Save and close the Object Editor to include the new Features in the configured object.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Enable I/O for an attribute

The I/O Feature allows you to configure all aspects of data input and output for an attribute.

You can configure I/O type and you can specify input sources and output destinations. The I/O types you can specify are:

- Read (Input): See Configure I/O as read-only.
- Read/Write (InputOutput): See Configure I/O as read/write.
- Write (Output): See Configure I/O as write-only.

You can also configure advanced properties, described in the following table. The attribute's data type and I/O type determine what Advanced I/O properties are available.

I/O Feature Advanced Property	Description
Buffered	Enable buffered data to propagate to data subscribers the entire subset of values accumulated within a single scan cycle. Buffering data ensures that if a given attribute changes its value several times during a single scan cycle, there is no folding of data, which occurs when an accumulation of values of multiple data changes within a single scan are overwritten and only the latest value is stored. **Note:** If Galaxy-wide buffering is enabled, you cannot enable (or disable) buffering for individual attributes. With Galaxy-wide buffering, all attributes are buffered. See Auto-buffering for additional information.
Deadband	Specify the minimum amount by which a value must vary in order for the attribute to register a change, for example, by triggering an alarm, historizing an alarm or event, or triggering a script. The Deadband property is not available for string data types.
Reflect input to output	Enable to propagate an input value to an output destination. Enabling automatically disables the **Output destination differs from input source** option. Typically, set this option when you want to read an input from one source, manipulate its value in a script, and send the manipulated value to a different destination address, all during a single scan of an object. For more information, see Use read/write I/O in scripts. Available only when the I/O type is Read/Write. You

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

I/O Feature Advanced Property	Description
	must set separate **Read from** and **Write to** properties.
Output every scan	Available only when **Reflect input to output** is selected. Write to the specified output destination occurs even when there has been no state or data change since the previous scan. The timestamp is not updated if there has been no state or data change since the previous scan.
Enable I/O scaling	Enables scaling between the raw value and the Engineering Units (EU) value. Scaling is the process of taking raw data from a device and presenting it as an appropriate value for your application. Application Server supports two types of data scaling: linear and square root, described in this table. Available only for integer, float, and double data types.
Maximum	Available only when I/O scaling is enabled. Raw: The raw input maximum to be used in the scaling equation. EU value: The maximum value in engineering units to be used in the scaling equation. Extended EU range: The highest value allowed for the attribute before it is clamped, if clamping is enabled, or set to NaN. This value must be greater than or equal to the specified maximum EU value.
Minimum	Available only when I/O scaling is enabled. Raw: The raw input minimum to be used in the scaling equation. EU value: The minimum value in engineering units to be used in the scaling equation. Extended EU range: The lowest value allowed for the attribute before it is clamped, if clamping is enabled, or set to NaN. This value must be less than or equal to the specified minimum EU value.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

I/O Feature Advanced Property	Description
Conversion mode	Available only when I/O scaling is enabled. Select Linear or Square Root conversion mode from the list. Linear: Typically converts directly from one value scale and type (raw) to another value scale and type (EU). *ScaledValue = ((RawValue-RawMin)/(RawMax-RawMin) * (EngUnitsMax-EngUnitsMin)) + EngUnitsMin* Square Root: Provides more precise scaling, typically used when the raw value is too large to be usefully scaled to an EU. *ScaledValue = sqrt((RawValue-RawMin)/(RawMax-RawMin))* *If RawValue < RawMin, then ScaledValue = (sqrt(RawMin) * (EngUnitsMax-EngUnitsMin)) + EngUnitsMin*
Clamp input to EU range	Available only when I/O scaling is enabled. If enabled, the scaling calculation result is clamped at either the maximum EU range value or the minimum EU range value, and the attribute quality is set to Uncertain. If not enabled, and the attribute value exceeds the EU range, then the value will continue to scale out of range, and the quality is set to Bad.

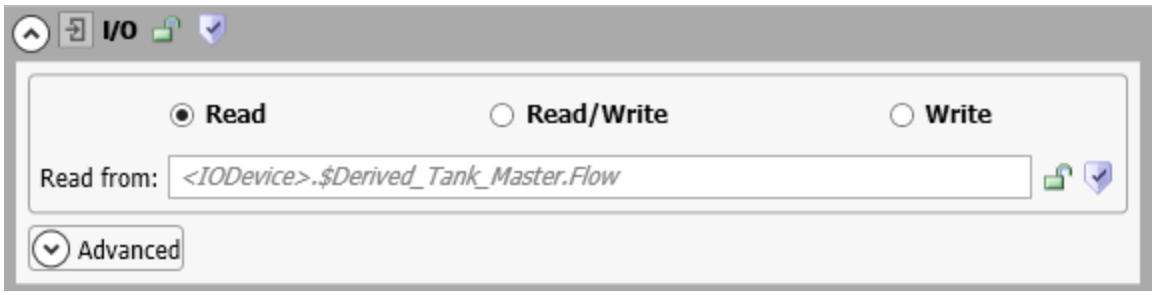
Configure I/O as read-only

Select **Read** in the I/O parameters area. Define an input **source** by using I/O auto-assignment, by typing in the reference string, or by clicking the **Attribute Browser** button at the right.

- Use I/O auto-assignment to prepare the input source for automatic assignment to a configured Device Integration (DI) object or other data source. For information about using I/O auto-assignment, see I/O auto assignment.

  Important: If the InputSource attribute is locked in the parent template, the attribute cannot be updated with the resolved reference when the object is deployed, and the runtime value will be "---Auto---".
- Use the **Attribute Browser** to select an attribute and automatically insert the correct reference string for that attribute. For more information about using the Attribute Browser, see Use the galaxy browser to reference objects .
- **Telemetry Server:** See ATS-configure I/O for use with AVEVA Telemetry Server Communication Drivers for additional information.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.



You can add multiple Read (input) I/O Features to an object. However, you cannot add a Read/Write (InputOutput) I/O Feature to an attribute that already has either a Read or a Write (output) I/O Feature. Arrays are not supported.

Note: Lockable attributes can be configured with a Read I/O Feature, but they only function correctly during runtime if the configured attribute is unlocked.

If the data types of the attribute and its I/O source attributes are the same, they are set to equal values according to the object's execution rate. If the two attributes are different data types, coercion rules are applied.

If coercion fails or the input value is out of the attribute's range, quality for the configured attribute is set to Bad. Otherwise, the configured attribute quality matches the **source** attribute. When the object is Off Scan, quality is always Bad and user sets are accepted.

Attributes configured with a Read I/O Feature are not protected by their security classification. The only enforced security specifies if an IDE user can edit or add features to the object. A Read I/O Feature can be added to a template or instance. If added to a template, the existence of the Read I/O Feature is automatically locked in derived objects.

Configure I/O as read/write

Select **Read/Write** in the I/O parameters area. Define an input **source** by using I/O auto-assignment, by typing in the reference string, or by clicking the **Attribute Browser** button at the right.

- Use I/O auto-assignment to prepare the input source for automatic assignment to a configured Device Integration object or other data source. For information about using I/O auto-assignment, see I/O auto assignment.

  If InputSource or OutputDest attributes, or both, are locked in the parent template, the attributes cannot be updated with the resolved reference when the object is deployed, and the runtime value will be "---Auto---".
- Use the **Attribute Browser** to select an attribute and automatically insert the correct reference string for that attribute. For more information about using the Attribute Browser, see Use the galaxy browser to reference objects .
- If you are configuring an I/O attribute for use with the Telemetry Server, see ATS-configure I/O for use with AVEVA Telemetry Server Communication Drivers for additional information.

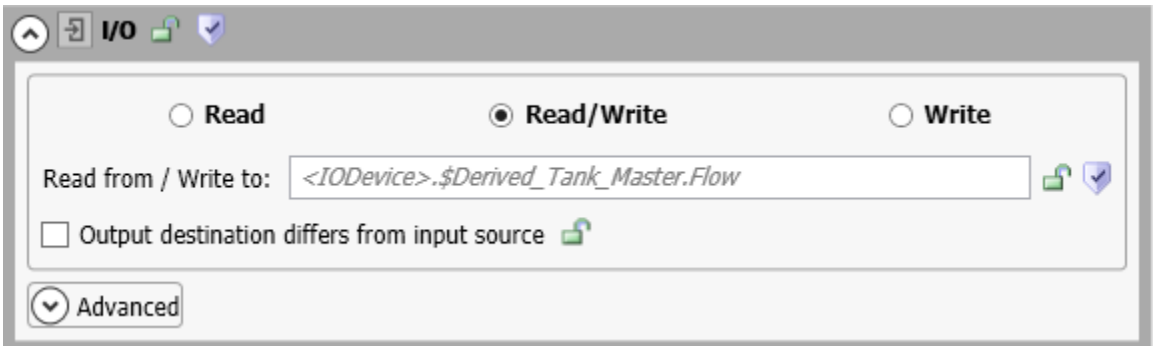
If the output destination and the input source are not the same, click **Output destination differs from input source**. Enter a **Destination** attribute by using I/O auto-assignment, by typing in the reference string, or by using the **Attribute Browser** to search for the reference string in an object.

Important: If you clear the **Output destination differs from input source** check box, the **Write to** text box automatically shows "---". In the runtime environment, "---" is the same reference as the **Read from** value entered during configuration time. During runtime, you can change the **Read from** reference. During

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

configuration, do not lock the **Write to** parameter if you clear the **Output destination differs from input source** check box.

A Read/Write I/O Feature allows an attribute in a template or an instance to be configured so that its value is both read from and written to an external reference. The Read/Write I/O Feature monitors the value/quality of an input and sends outputs on state change.



The **Write to** (output) destination can be the same or different from the **Read from** (source). The references are always to another acceptable attribute type in the Galaxy.

You can add multiple Read/Write I/O Features to an object. However, you cannot add a Read/Write I/O Feature to an attribute that already has a Read or Write I/O Feature.

Note: You can add a Read/Write I/O Feature to lockable attributes, but they only function correctly during runtime if the configured attribute is unlocked.

When objects are on scan

When an object is On Scan, the value and quality of the attribute configured with a Read/Write I/O Feature mirrors the quality of the externally referenced attribute during a successful read. The data quality of the attribute is set to Bad when reads fail. Reads can fail because of communication errors or datatype conversion failures.

While the object is On Scan, the data can change quality. If an external set (for example, from a user) to an attribute changes either the value or quality, then a write of the attribute's value to the destination occurs during the next execute phase. The quality must be Good or Uncertain for a write to occur. For writes to occur because of a quality change, the quality change must be a transition from Bad or Initializing to Good or Uncertain.

The attribute called WriteValue is publicly exposed and plays an important role in driving outputs. When the object is Off Scan, quality is always Bad and user sets are accepted.

Use read/write I/O in scripts

Two common types of scripts can be written on an attribute configured with a Read/Write I/O Feature: One can look at the input side and one can look at the output side.

The input side script uses the current value coming from the input source location and performs logic or calculations on it. This script refers directly to the attribute in its expressions. For example, if the attribute is `"me.attribute1"`, the script refers directly to `"me.attribute1"` for data change conditions and for expressions within the script.

The output side script can modify output or validate a new requested output value. This script refers to the "WriteValue" attribute configured on the attribute: `"me.attribute1.WriteValue"`.

To validate a new requested value to the attribute1, for example, a data change condition expression is written

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

on "me.attribute1.WriteValue". In addition, if the script wants to do clamping or validation, it can manipulate the "me.attribute1.WriteValue" directly to clamp the output value. For example:

```
If (me.attribute1.WriteValue > 100.0 ) then
Me.attribute1.WriteValue = 100.0;
Endif;
```

The data change expression for this script is "me.attribute1.WriteValue" because this value changes when a new value is about to be written to the field.

The script can intercept this value just before output and manipulate it. To prevent WriteValue from being written out, its data quality can be set to Bad with the SetBad() function.

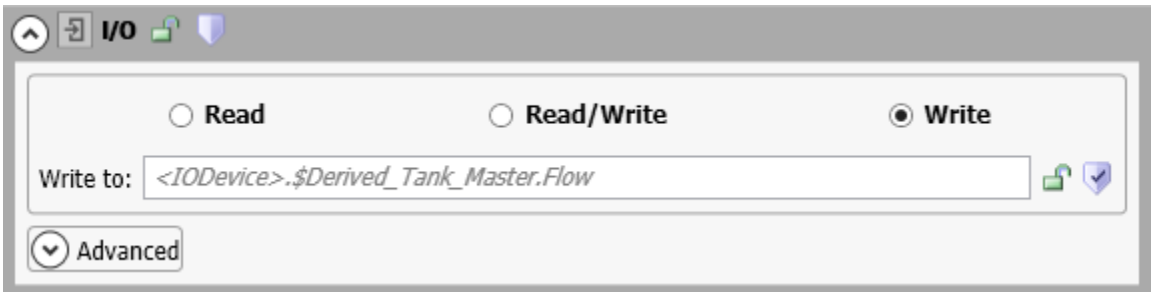
For more information, see I/O outputs.

Configure I/O as write-only

Select **Write** in the I/O parameters area. Define a **Write to** (output) **destination** by using I/O auto-assignment, by typing in the reference string, or by clicking the **Attribute Browser** button at the right.

- Use I/O auto-assignment to prepare the output destination for automatic assignment to a configured Device Integration (DI) object or other data source. For information about using I/O auto-assignment, see I/O auto assignment.
- If the OutputDest attribute is locked in the parent template, the attribute cannot be updated with the resolved reference when the object is deployed, and the runtime value will be "---Auto---".
- Use the **Attribute Browser** to select an attribute and automatically insert the correct reference string for that attribute. For more information about using the Attribute Browser, see Use the galaxy browser to reference objects .

Select the **Output Every Scan** check box if you want the attribute to write to the **Destination** attribute every scan period of the object. Otherwise, the write executes only when the value is modified or when quality changes from Bad or Initializing to Good or Uncertain.



Writeable and Calculated attributes can be configured with a Write I/O Feature. Arrays are not supported.

A Write I/O Feature can be added to a derived template or to an instance. If added to a template, the existence of the Write I/O Feature is automatically locked in derived objects. The output **Destination** attribute is separately lockable in templates.

If the data types of the configured and destination attributes are the same and only when the quality of the extended attribute is good, the two attributes are set to equal values according to the configured object's execution rate. If the two attributes are different data types, coercion rules are applied. If coercion fails, the extended attribute is placed into a configuration error and type mismatch state.

An attribute that is enhanced with a Write I/O Feature has the following characteristics:

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

- A value can be output only when quality is Good or Uncertain. The quality is not output, only the value is output, because quality is not output on sets.
- When the quality changes from Bad or Initializing to Good or Uncertain, the value is output, even if the value is not modified.
- When the quality changes from Good to Uncertain, with no value modification, the value is not output.
- When the object goes Off Scan, no output is done.
- When the extended object is Off Scan, quality is always Good and user sets are accepted.

I/O outputs

The following information applies to the functionality of Read/Write and Write Features as well as to the output function of AnalogDevice objects.

If a single set request is made to a destination attribute during a single scan cycle, that value is sent to the destination. During a single scan cycle, though, more than one set request to the same destination is possible. In that case, folding occurs and the last value is sent to the destination.

During a single scan cycle, only the last value requested during a scan cycle is sent to its destination when the object executes. Its status is marked as Pending as it waits for write confirmation from the destination object. All other set requests during that scan cycle are marked as successfully completed.

If one or more new sets are requested during the next scan cycle, then the second scan cycle's value is determined as described in the preceding paragraphs. It is then sent to the destination when the object executes again and the value sent to the destination during the previous scan cycle is marked with successful completion status even if write confirmation is not received.

Within a single scan cycle, data is folded and only the last set requested is sent to the destination, unless buffering is enabled. For example, an {11,24,35,35,22,36,40} sequence of set requests results in a value of 40 being sent to the destination object. All other values result in successful completion status.

The exception to this behavior is when you enable buffering, either on a Galaxy-wide basis or for an individual attribute. For more information, see I/O Feature advanced properties information in Enable I/O for an attribute.

Boolean data types are an exception to folding behavior when buffering is not enabled. This allows an unknown user input rate (for example, repeated button pushes) with a consistent object scan rate for outputs, and creates reproducible results.

In this case, a combination of folding as described plus maintenance of a queue of one element deep better meets the expectation of users. To begin with, the first value set after the object is deployed (the default True or False) is always written to its destination.

Subsequently, the following occurs during a single scan cycle: A two-tiered caching scheme of a Value to be Sent and a Next Value to be Sent is implemented. The Value to be Sent is based on data change as compared to the last value sent to the destination object. The Next Value to be Sent is based on data change as compared to the Value to be Sent value.

When the first data change occurs, the new value is cached in the Value to be Sent queue. Folding occurs if the same value is requested again. If another value change occurs, this second value is cached in the Next Value to be Sent queue. Again, folding occurs if the same value is requested again.

The Value to be Sent value is sent during the next scan cycle, and the Next Value to be Sent value is sent during the following scan cycle.

Note: In the case of Boolean data types used in Supervisory sets (sets between ApplicationObjects) or a mixture of Supervisory and User sets during a single scan cycle, the behavior is the same as the other data types.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

For Boolean data types and User sets, the following examples apply:

Previous Scan Cycle Value Sent	Scan Cycle Set Requests	Value to be Sent	Next Value to be Sent
0	1,0,0,1,1	1	none
1	1,0,0,1,1	0	1
0	1,1,0,0	1	0
1	1,1,0,0	0	none

When the same attribute is extended with an Input extension and an Output extension, writes to the Output extension's **Destination** occur every scan regardless of whether the extended attribute has changed.

This behavior occurs even when the **Output Every Scan** check box is cleared, which may add more network traffic. The behavior does not apply to an Input extension.

I/O data quality

When the object is On Scan, the value and quality of an attribute configured with a Read I/O Feature mirrors the quality of the externally referenced attribute in the case of successful reads. The data quality of the configured attribute is set to Bad when reads fail because of communication errors or datatype conversion failures.

While the extended object is On Scan, it behaves as follows: If an external set (for example, from a user) to the configured attribute causes either the value or quality to change, then a write of the configured attribute's value to the destination occurs during the next execute phase.

The quality must be Good or Uncertain for a write to occur. For writes to occur because of a quality change, the quality change must be a transition from Bad or Initializing to Good or Uncertain. The attribute called WriteValue is publicly exposed.

When the configured object is Off Scan, quality is always Bad and user sets are accepted.

ATS-configure I/O for use with AVEVA Telemetry Server Communication Drivers

If you are using AVEVA Telemetry Server Communication Drivers to send and receive data from Application Server, you will need to configure I/O attributes, in accordance with the applicable communications protocol. The Telemetry Server communicates with outstations via the DNP3, IEC-60870, and/or Modbus protocols. Use the reference syntax that corresponds with the protocol you are using. Since some outstations may send data to the Telemetry Server only once per day, enable buffering for the Galaxy to ensure that data is not lost. See Auto-buffering for information about enabling buffering on a Galaxy-wide basis.

Note: If multiple attributes are configured for buffered data from the same data point in the Telemetry Server, or if auto-buffering is enabled, only one attribute will receive the buffered data from that data point at runtime.

Dynamic References

You can configure Telemetry Server data points using dynamic references. When you create and deploy an attribute with a dynamic reference in Application Server, the point is automatically generated on the Telemetry Server, and eliminates the need to manually create the same reference in two places.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

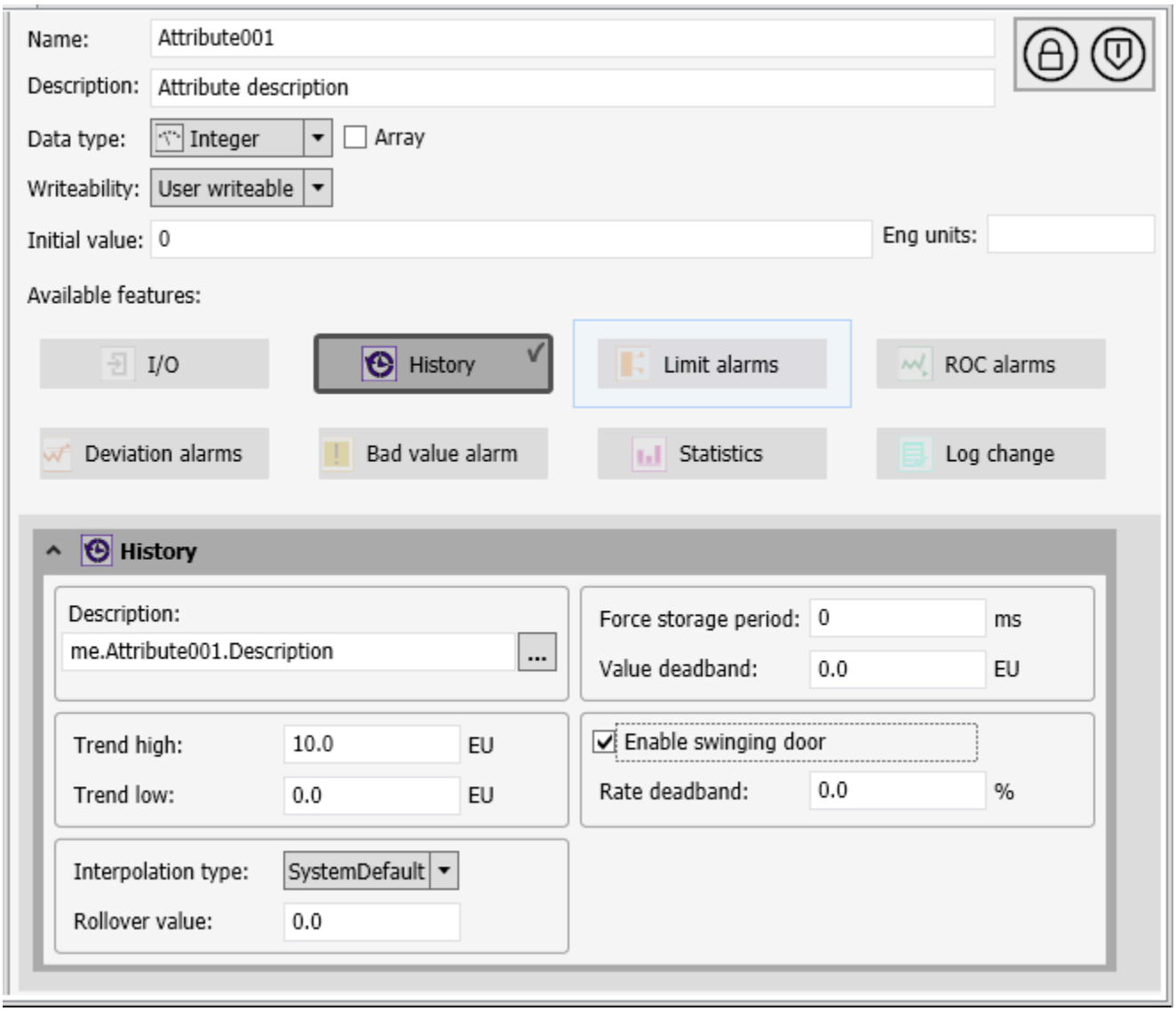
Additional Information

For more information about enabling buffered data and creating Telemetry Server references, see the following sections:

- Auto-buffering
- Enable buffered data
- Telemetry server data references
- Telemetry server dynamic tags

Enable history for an attribute

Any attribute that exists at runtime and is not already historized can be configured with a history Feature.



A history Feature can be added to a template or an instance attribute. If added to a template attribute, the existence of the history Feature is automatically locked in derived objects.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

You can configure Writeable and Calculated attributes of the following data types with a history Feature:

- Float, Double (stored as a Float)
- Integer
- Boolean
- String stored as Unicode, 512 character limit
- Custom Enumeration stored as an Integer
- ElapsedTime stored as seconds

You can configure the following attributes for a history Feature:

History Feature Attributes	**Description**
Description	The attribute containing the description string, if any, associated with the attribute being configured. Enter the attribute manually, or use the browse button to open the Galaxy Browser to find a specific attribute. Example: For the description associated with Attribute001 that has been added to object Reactor31, you would enter or browse to "Reactor31.Attribute001.description". The description also can be a simple string without reference to an attribute.
Force storage period	The Historian normally stores values by exception. The force storage period is designed for situations where data is expected to change very infrequently, compared to the engine scan rate, for examples, setpoints and values which only toggle in abnormal cases. The force storage period sets a length of time, in milliseconds, after which the value must be historized, even if the value has not changed, or has not changed more the the **Value deadband**. Setting the value to 0 disables the parameter. Example: A setting of 3600000 (60 seconds/minute x 60 minutes/hour x 1000) will force the value to be stored once per hour (if the value remains unchanged), as measured from the time the object was last put onscan. The force storage period should be long relative to the AppEngine scan cycle. If the force storage period is shorter than the scan cycle, forced storage will occur at every scan period, as the value change is not recorded until the scan takes place.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

History Feature Attributes	Description
	When the storage period is set to a value greater than the scan period and there is no change in value, or if the change in value is less than the **Value deadband**, the forced storage period artificially updates the timestamp on the previous (unchanged) value, and allows the Historian to store that value again with the artificial timestamp. In most cases, that is, in cases where a change in value occurs, values are associated with the timestamp from the source system (e.g., PLC), and that time is applied to the historized values. On the other hand, if a value is historized because of the force storage period, the value is instead stored with the system time of the engine scan when the force storage period was exceeded. Factors such as communications latency, clock differences, and detailed engine timing can result in force storage values being historized out of order or overlapping with those values historized due to value changes. It is also possible that the timeout for the forced storage period may coincide exactly with the timestamp of an actual exception update from the instrumentation. If this occurs, both values are stored with the associated timestamp. To avoid this, disable (set to 0) the force storage period in cases where the value normally updates frequently. In cases where the value rarely changes, set the force storage period to a much longer time period, for example, to a value between 1 and 48 hours (3600000 to 172800000 ms).
Value deadband	The threshold value, measured in engineering units, that the absolute value of the difference between the new and last-stored values must differ before storing the new value to history. The value deadband is intended to filter out low-level noise in the signal so that only significant changes are historized. When a value deadband is set, value changes that are within the value deadband will not be historized. Therefore, you should only specify a value deadband when the reduction of processing and storage overhead is more important than completeness of the data. If you choose to set a value deadband, the value should be set to as small a value as possible. The value should be based on how large (or small) the changes in value that are expected.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

History Feature Attributes	Description
	A value of zero (0) is valid and means that any level of change results in the new value being stored. A change in Quality always causes a new record to be stored, regardless of whether the Value has changed. When combined with the **Force storage period**, new values are compared with the last historized value, regardless of whether the the previous value was historized because it exceeded the value deadband or exceeded the force storage period.
Trend high	The default top of a trend scale. This value must be greater than or equal to the low value for the trend. If this value is changed at runtime, the maximum engineering unit change is not reflected in the HIstorian until you redeploy the object. This attribute applies only to numeric data types.
Trend low	The default bottom of a trend scale. This value must be less than or equal to the high value for the trend. If this value is changed at runtime, the minimum engineering unit change is not reflected in the HIstorian until you redeploy the object. This attribute only applies to numeric data types.
Enable swinging door	Enable and provide a valid swinging door rate deadband and the force storage period becomes the deadband override period. Boolean or string data types cannot be configured with a swinging door deadband.
Rate deadband	Available only if swinging door is enabled. The percentage rate of change deadband based on the change in the slope of incoming data values to the HIstorian. Example: A swinging door rate deadband of 10 percent means that data is saved to the HIstorian if the percentage change in slope of consecutive data values exceeds 10 percent. Default is 0.0, which indicates a swinging door rate deadband is not applied. Any percentage greater than 0.0 can be assigned to the rate deadband.

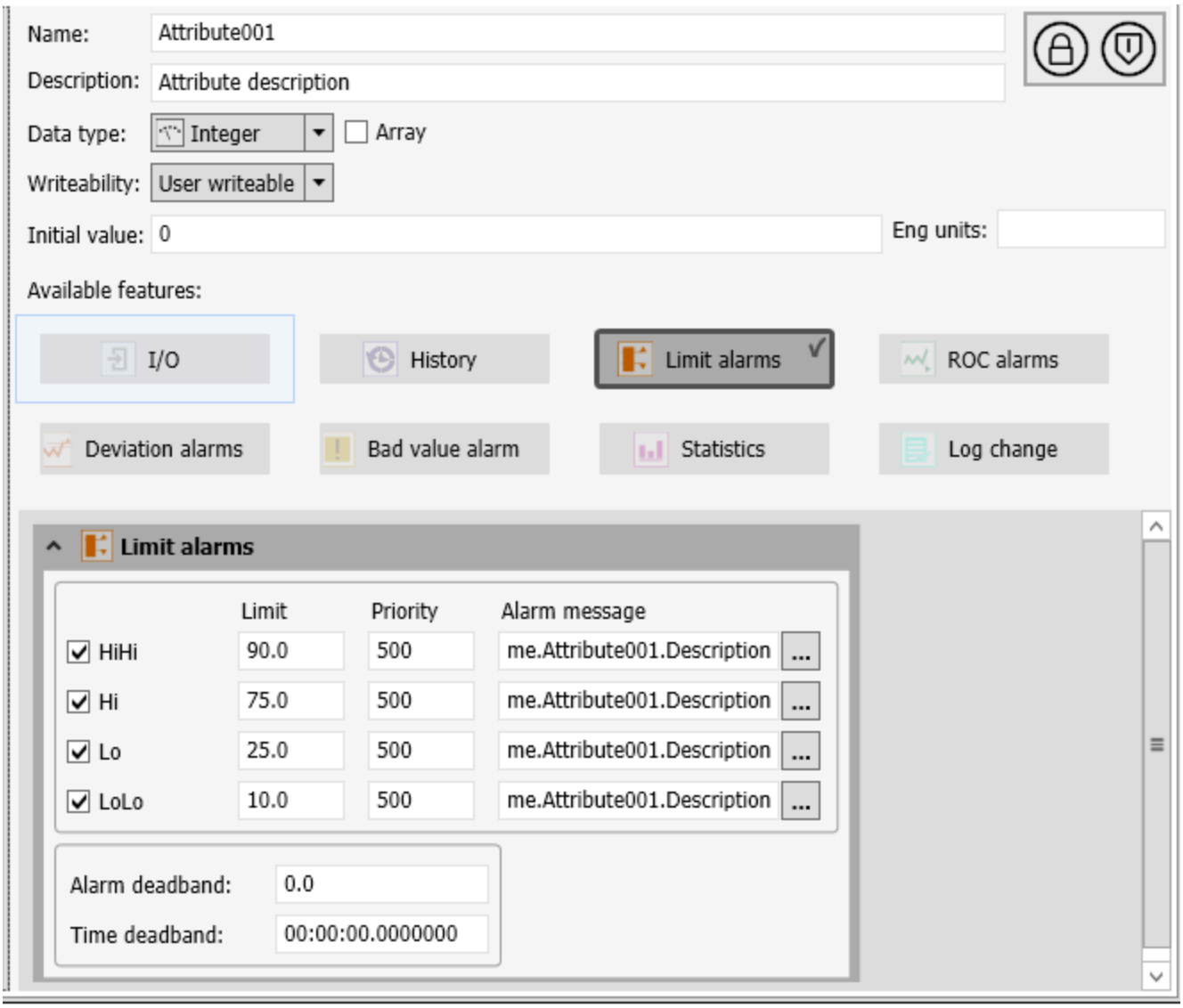
© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

History Feature Attributes	Description
Interpolation type	The method used by the Historian to interpolate analog historical data. The interpolation type determines which analog value is selected during a Historian ata retrieval cycle. Select an Interpolation type: **System Default**: The HIstorian system-wide interpolation setting is used. The system-wide setting must be either stairstep or linear interpolated. **Stairstep**: The last known value is returned with the given cycle time. If no valid value can be found, a NULL is returned to the HIstorian. **Linear**: HIstorian calculates a new value at the given cycle time by interpolating between the last known value prior to the cycle time and the first value after the cycle time.
Rollover value	A non-negative numeric value that represents a tag's reset limit when the Historian operates in counter retrieval mode. In counter retrieval mode the Historian uses a tag's rollover value to calculate and return the delta change between consecutive retrieval cycles. The default value is 0.0. Boolean and string data types cannot be configured with a rollover value.

Enable limit alarms for a numeric attribute

Select the **Limit alarms** Feature to add and configure a Limit Alarm on an attribute of Integer, Float, or Double data type. You can add a Limit alarm Feature to a template or instance. If added to a template attribute, the Limit alarm Feature is automatically locked in derived objects. Limit alarm Features cannot be added to attribute arrays.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.



You can enable up to four categories of Limit alarms. When enabled, alarms will be triggered when the value reaches the configured limit.

- HiHi
- Hi
- Lo
- LoLo

For each Limit alarm, specify the following parameters:

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

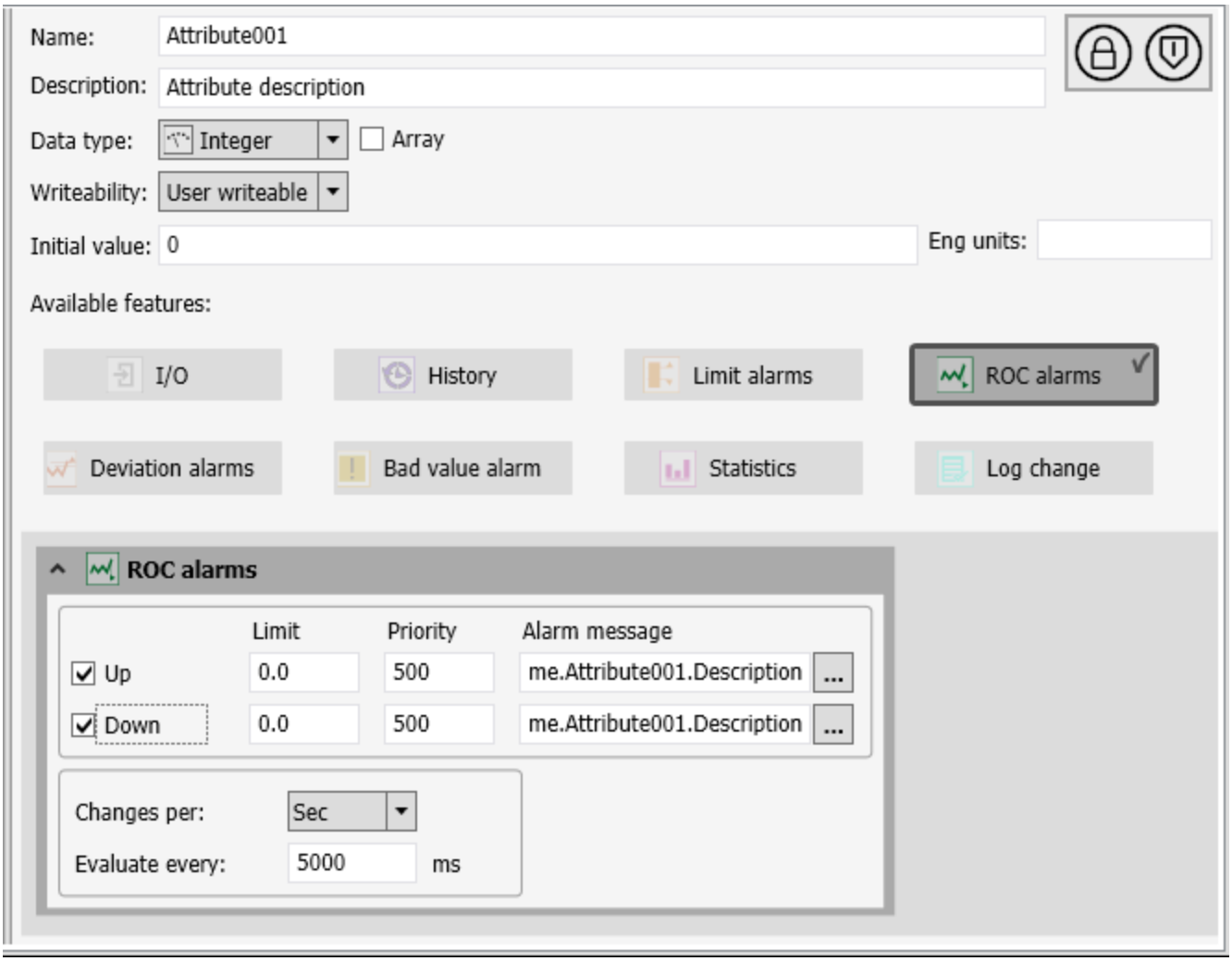
Limit Alarm Feature Parameters	Description
Limit	The limit that the attribute value must exceed to trigger an alarm. Enter the limit value for each enabled Limit alarm category.
Priority	A numeric value for the urgency of the alarm. Valid values are 1 through 999, with 1 being the most urgent. For more information about pre-configuring alarm priorities, see Configure priority ranges for alarm historization, mapping, and shelving .
Alarm message	Browse and select an existing attribute or type a text string as an alarm message. This text string appears in the InTouch alarm view.
Alarm deadband	The amount, in engineering units, that the attribute value must exceed the configured HiHi, Hi, Lo, or LoLo limit before an alarm is triggered.
Time deadband	The time, in seconds, that must elapse after the attribute value exceeds an alarm limit before the alarm is triggered.

For additional information, see Limit alarms.

Enable rate of change (ROC) alarms for a numeric attribute

Select the **ROC alarms** Feature to add and configure a Rate of Change (ROC) alarm on an attribute of Integer, Float, or Double data type. You can add a ROC alarm Feature to a template or instance. If added to a template attribute, the ROC alarm Feature is automatically locked in derived objects. ROC alarm Features cannot be added to attribute arrays.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.



You can enable up to two categories of ROC alarms. When enabled, alarms will be triggered when the value reaches the configured limit.

- Up: Rate of increase
- Down: Rate of decrease.

For each ROC alarm, specify the following parameters:

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

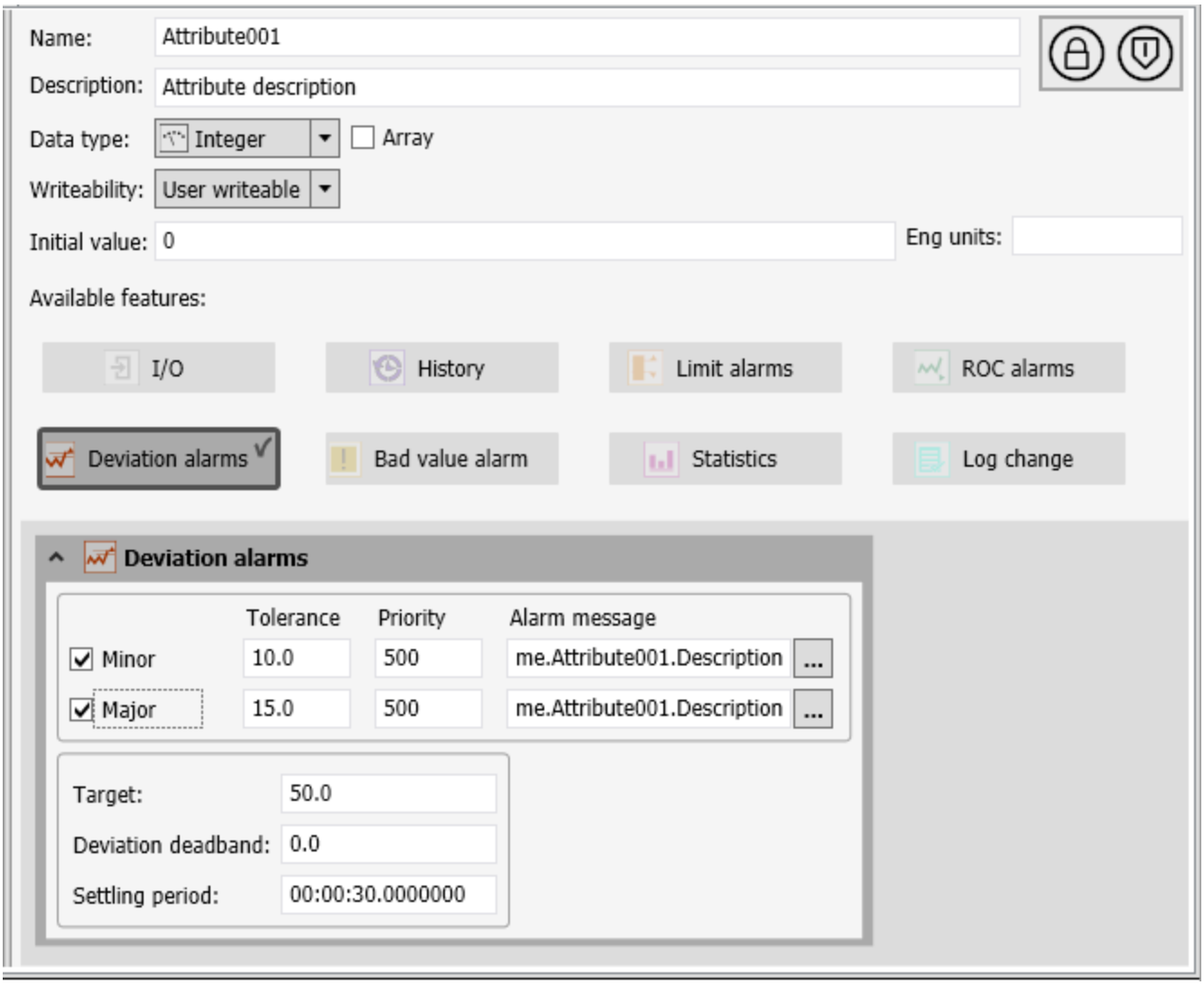
ROC Alarms Feature Parameters	Description
Limit	The limit that the attribute rate of change value must exceed to trigger an alarm. Enter the limit value for each enabled alarm category. The down limit is a positive value, but indicates a negative rate of change, or a descending rate Hi limit.
Priority	A numeric value for the urgency of the alarm. Valid values are 1 through 999, with 1 being the most urgent. For more information about pre-configuring alarm priorities, see Configure priority ranges for alarm historization, mapping, and shelving .
Alarm message	Browse and select an existing attribute or type a text string as an alarm message. This text string appears in the InTouch alarm view.
Changes per	Select a unit of time for the rate of change calculation – seconds, minutes, hours, or days.
Evaluate every	The time interval, in milliseconds, at which the rate of change calculation and detection is performed.

For additional information, see Rate of change alarms.

Enable deviation alarms for a numeric attribute

Select the **Deviation alarms** Feature to add and configure a Deviation alarm on an attribute of Integer, Float, or Double data type. You can add a Deviation alarm Feature to a template or instance. If added to a template attribute, the Deviation alarm Feature is automatically locked in derived objects. Deviation alarm Features cannot be added to attribute arrays.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.



You can enable up to two categories of Deviation alarms. When enabled, an alarm will be triggered when the attribute level deviates from a target value by a configured amount.

- Major: The major alarm deviation tolerance.
- Minor: The minor alarm deviation tolerance.

For each Deviation alarm, specify the following parameters:

Deviation Alarms Feature Parameters	**Description**
Tolerance	The major or minor deviation tolerance that the attribute value must exceed to trigger an alarm.
Priority	A numeric value for the urgency of the alarm. Valid values are 1 through 999, with 1 being the most urgent. For more information about pre-configuring alarm

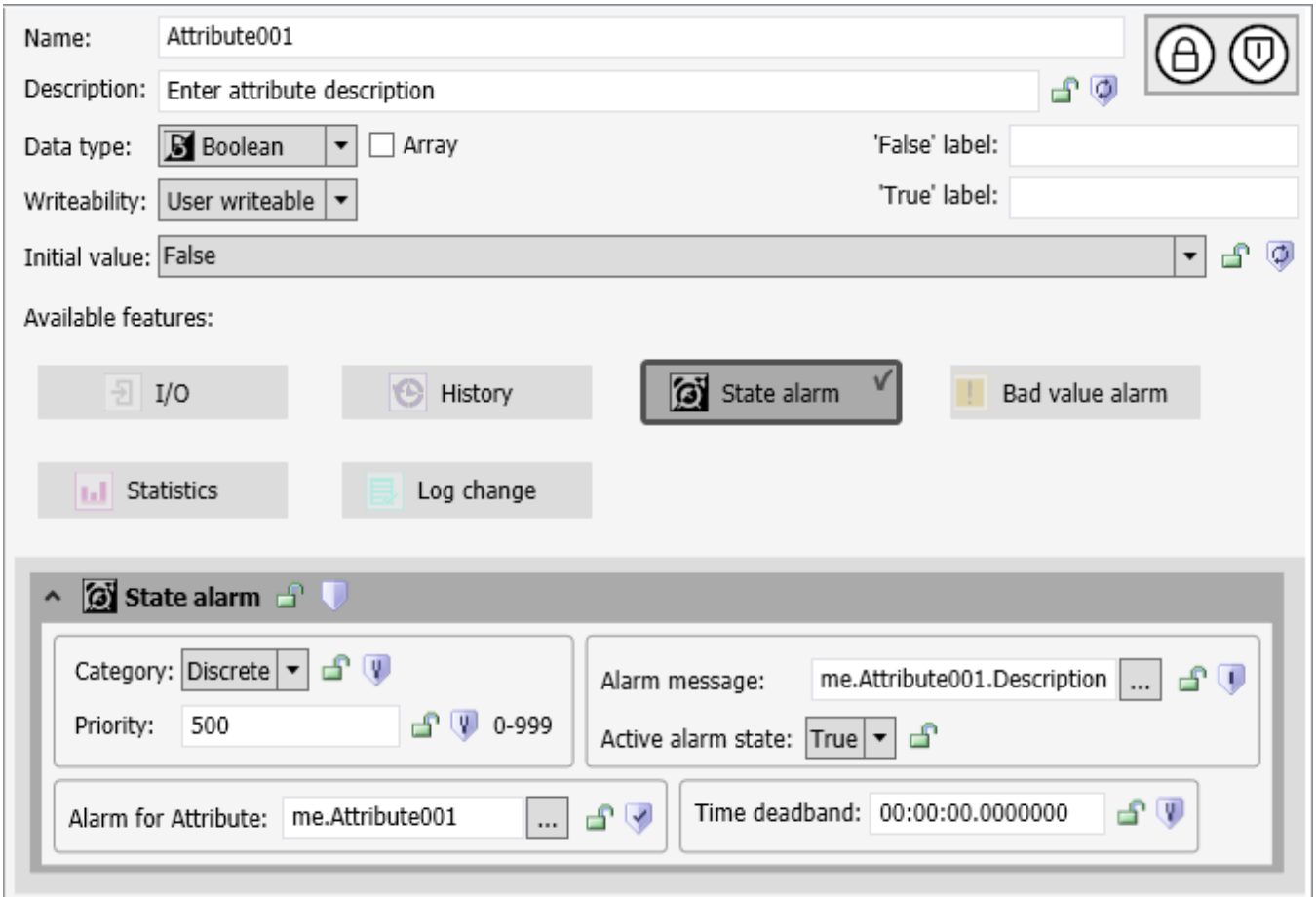
© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Deviation Alarms Feature Parameters	Description
	priorities, see Configure priority ranges for alarm historization, mapping, and shelving .
Alarm message	Browse and select an existing attribute or type a text string as an alarm message. This text string appears in the InTouch alarm view.
Target	The setpoint value for deviation calculation.
Deviation deadband	The amount the attribute value must drop (measured in engineering units) below any of the deviation limits before the respective condition is set to FALSE.
Settling period	The time, in seconds, allowed for the attribute value to return within allowable range after a Target change is made before the deviation alarm is triggered. A deviation alarm cannot occur again until the alarm clears. The value you specify will be converted to the following format: days hh:mm:ss.fffffff.

Enable a state alarm for a boolean attribute

Select the **State alarm** Feature to add and configure a State alarm on an attribute of boolean data type. You can add a State alarm Feature to a template or instance. If added to a template attribute, the State alarm Feature is automatically locked in derived objects. State alarm Features cannot be added to attribute arrays.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.



You can set an alarm message and Priority for a state alarm. The time deadband is used to filter out rapid, transitory value spikes, and the active alarm state determines the trigger value (True or False) that will activate the alarm. You can also link the State alarm to an external attribute to capture alarm conditions detected in the PLC, rather than in the Application Server object.

For each State alarm, specify the following parameters:

State Alarms Feature Parameters	**Description**
Category	The alarm category. Select a category from the list box. See State alarms for additional information.
Priority	A numeric value for the urgency of the alarm. Valid values are 1 through 999, with 1 being the most urgent. For more information about pre-configuring alarm priorities, see Configure priority ranges for alarm historization, mapping, and shelving .
Alarm message	Browse and select an existing attribute or type a text string as an alarm message. This text string appears in the InTouch alarm view.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

State Alarms Feature Parameters	Description
Active alarm state	The state of the active alarm, True or False, when the alarm is triggered.
Alarm for Attribute	This lets you link to an external attribute, such as a PV attribute, to provide historical state alarm data from the PLC. This allows the PLC to pass alarm conditions it detects directly to Application Server. You can use the Attribute Browser to select which attribute to link. The Alarm for Attribute parameter is for use only with Historian Clients and is not compatible with InTouch HMI clients. For more information, see Configure an external alarm source for a boolean attribute.
Time deadband	The time, in seconds, that must elapse after the attribute value exceeds an alarm limit before the alarm is triggered.

Configure an external alarm source for a boolean attribute

You can only configure an external alarm source for use with Historian Clients, not InTouch clients.

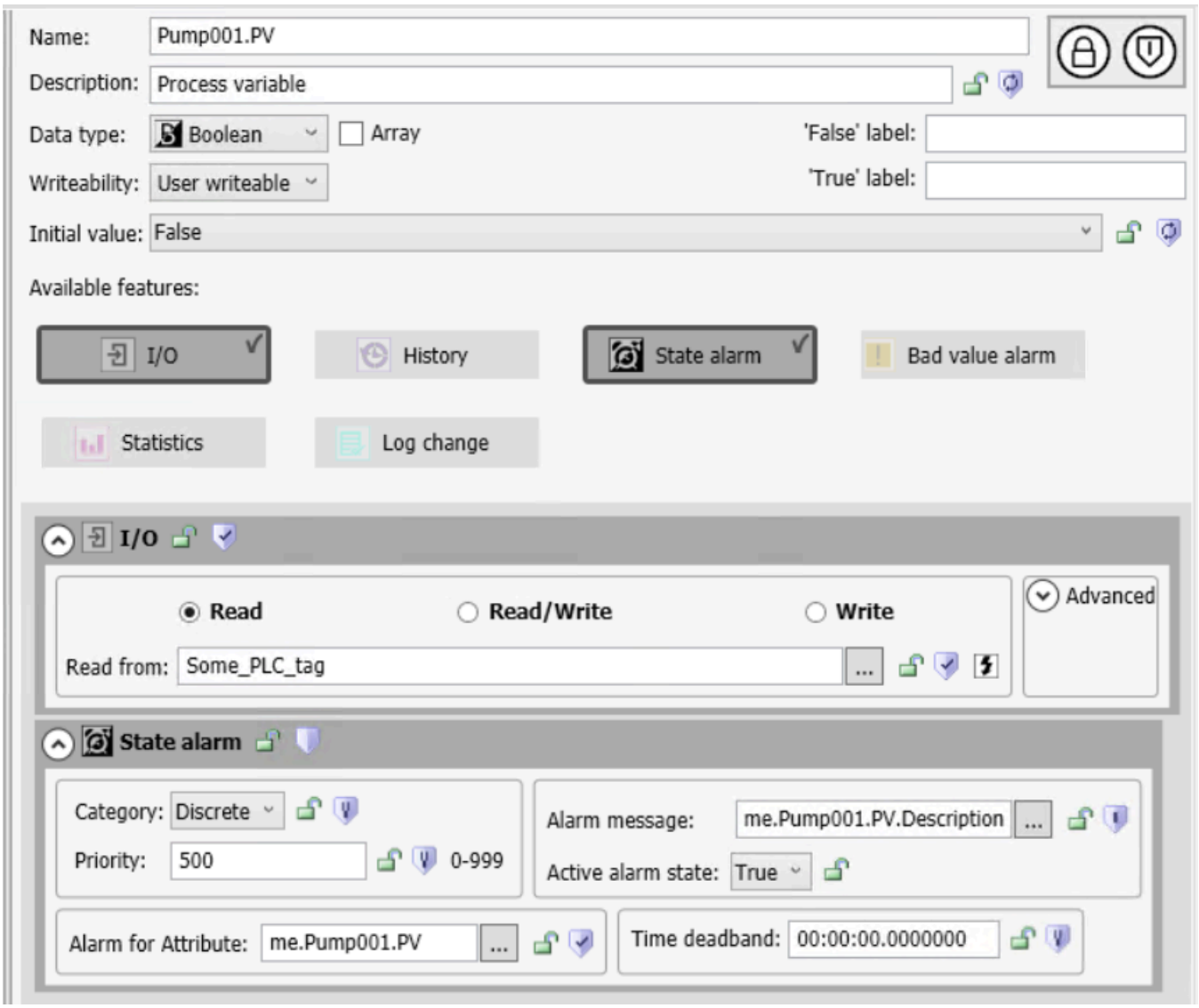
To get the alarm history records for an externally detected alarm (from within a PLC), create a reference to the source_processvariable property, as follows:

1. Create a boolean user-defined attribute.
2. Enable I/O and State Alarm features for the boolean.
3. Bind the IO feature for the boolean to a tag in the PLC.
4. Set the I/O to read the PLC alarm bit where the alarm is being detected in the PLC. Note that there is nothing in the boolean attribute definition to indicate it represents an alarm associated with another attribute (like a ".PV" attribute).
5. To establish the alarm association, associate the boolean (created in step 1) with another attribute. At runtime, this association will set the "source_processvariable" in the alarm history event. This allows clients of History Event and Alarms records to correlate externally detected alarms with the attributes against which they are being reported.

See the sample configuration, below, for additional details.

Sample external alarm source configuration

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.



Attribute Name: Pump001.PV

Type: Boolean

IO: Enabled

Points to a tag within the PLC.

State Alarm: Enabled

Alarm For Attribute: me.Pump001.PV

Attribute Name: Pump.PV.Hi

Type: Boolean:

IO: Enabled:

Points to an alarm tag in the PLC.

This tag will be true if there is a high alarm generated by Pump.PV.

State Alarm: Enabled

Alarm For Attribute: me.Pump001.PV

Attribute Name: Pump.PV.HiHi

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Type: Boolean:

IO: Enabled:

Points to an alarm tag in the PLC.

This tag will be true if a high-high alarm is generated by Pump.PV.

State Alarm: Enabled

Alarm For Attribute: me.Pump001.PV

Enable a bad value alarm for an attribute

Select the **Bad value alarms** Feature to add and configure a Bad Value alarm on an attribute of boolean, integer, float, or double data type. If enabled, alarms will be triggered when the attribute has bad data value or data quality.

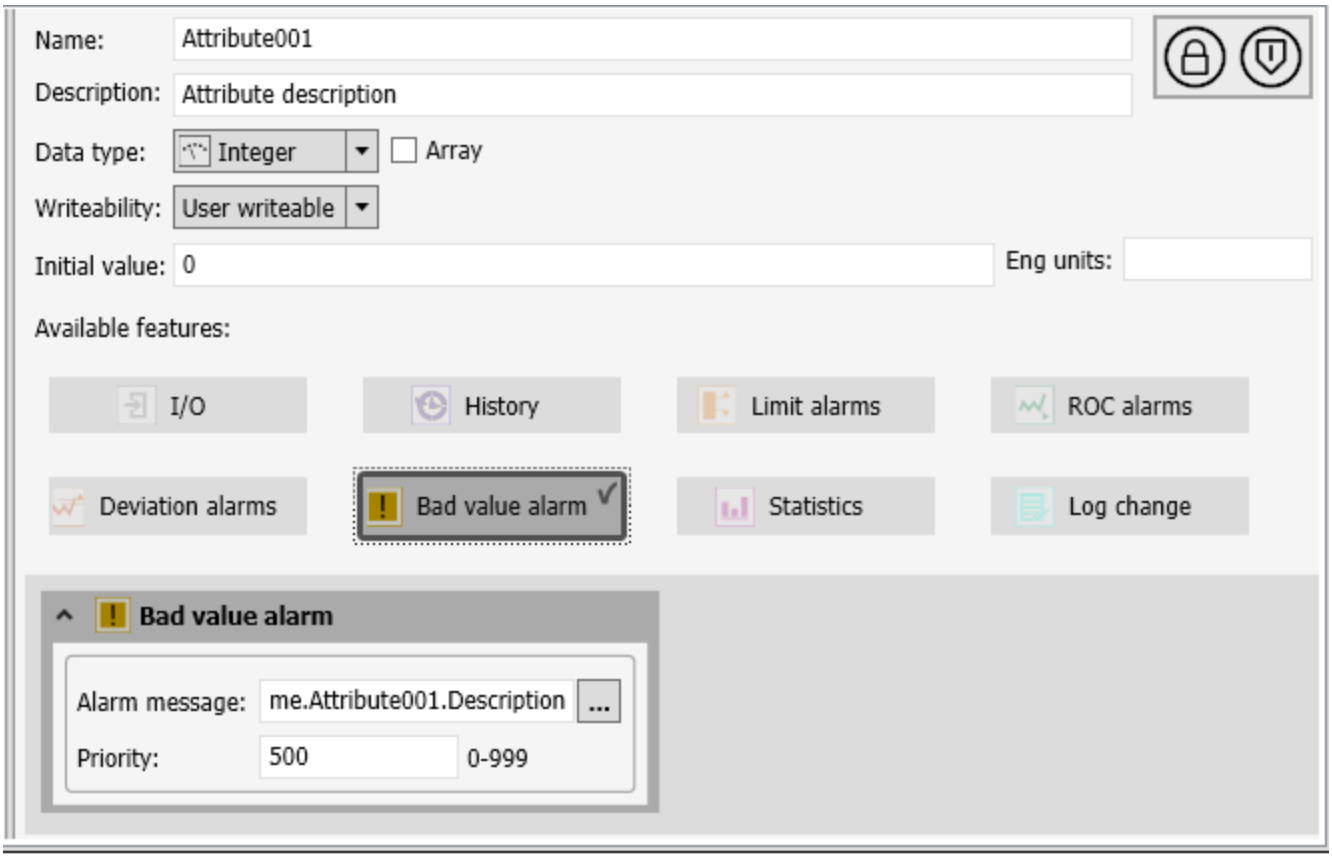
A bad value results from bad data quality. Bad quality can occur when:

- The PLC indicates that quality is bad (for example, if communications with a physical device is lost).
- There is a communications error (for example, an OPC or I/O server failure).
- There is a configuration error (for example, a broken I/O reference).
- An out of range value is recorded.
- A script sets data quality to bad.

You can configure graphic elements to highlight when a bad data quality state occurs, and its underlying cause. See Use quality and status indicators to provide runtime feedback for more information.

You can add a Bad Value alarm Feature to a template or instance. If added to a template attribute, the Bad Value alarm Feature is automatically locked in derived objects. Bad Value alarm Features cannot be added to attribute arrays.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.



In the **Alarm message** box, you can browse and select an existing attribute or you can type a text string as an alarm message. This text string appears in the InTouch alarm view.

Specify a **Priority** for this alarm, a numeric value for the urgency of the alarm. Valid values are 1 through 999, with 1 being the most urgent.

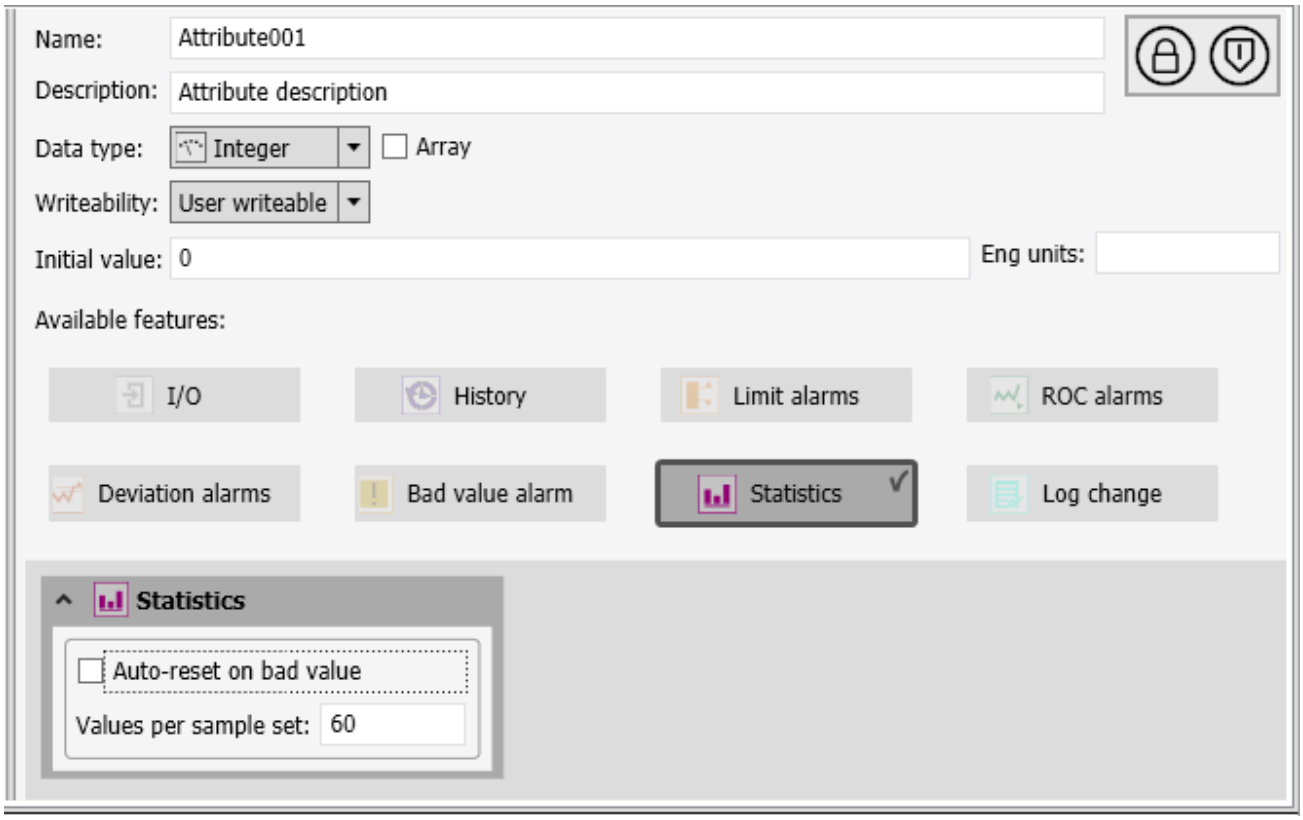
For more information about pre-configuring alarm priorities, see Configure priority ranges for alarm historization, mapping, and shelving .

Enable statistics for a numeric attribute

Select the **Statistics** Feature to add and configure calculation of statistics on an attribute of Boolean, Integer, Float, or Double data type. If enabled, the following statistics are calculated:

- Time since last transition: If no change in the input value, the elapsed time value increments to show the total elapsed time since the last transition. A change in input value resets the elapsed time since last transition.
- Average: Defined as the sum of the sample size divided by the number of sample size attributes in the data set.
- Standard deviation: Defined as the amount of variation or "scatter" from the average.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.



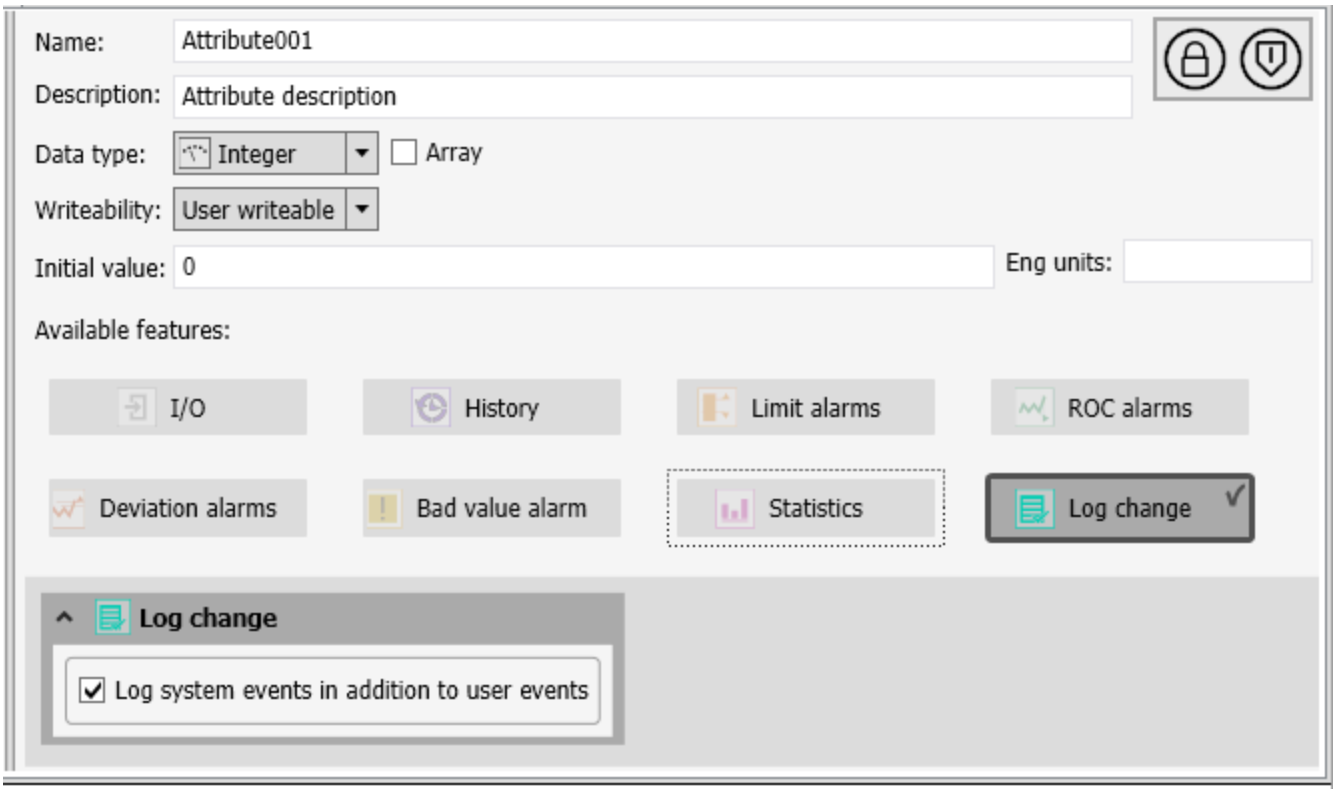
You can configure the following parameters for the Statistics Feature:

Statistics Feature Parameters	**Description**
Auto reset on bad value	If enabled, the input automatically triggers a reset command when the quality of an input goes from BAD or UNCERTAIN to GOOD.
Values per sample set	The maximum number of samples to collect in calculating the mean and standard deviation. The minimum number of samples is 2. A large sample set will impact runtime performance. This attribute is unavailable for a Boolean data type.

Enable log change for an attribute

Select the **Log Change** Feature to log system events for an attribute of Boolean, Integer, Float, or Double data type.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.



When enabled, the Log Change feature logs data change events as well as user events.

I/O auto assignment

Instead of configuring I/O references manually, or writing scripts to set them at runtime, you can use I/O auto assignment. Manual configuration of I/O references can be time consuming. Scripting these references eliminates the issues of manual configuration, but can significantly increase the time needed for deploying objects. With I/O auto assignment, you do not need to check out individual objects to configure I/O references, and you do not experience the runtime penalties associated with scripting.

Note: I/O auto assignment is the default setting for application and other system objects, such as area objects. Device Integration objects, such as the OPCClient object, must be manually configured.

When you add input or output attributes to an area or application object in the **Attributes** page of the Object Editor, the default setting prepares these attributes for I/O automatic assignment. The auto assignment reference appears in the I/O section of the **Attributes** page if you have enabled the I/O attribute feature. The default string for an input reference is:

```
<IODevice>.[ObjectName].[AttributeName].InputSource
```

where *[ObjectName]* is the hierarchical name of the application or system object, and [*AttributeName]* is the name of the attribute.

The default string for an output reference is:

```
<IODevice>.[ObjectName].[AttributeName].OutputDest
```

The string *<IODevice>* is a placeholder that indicates the I/O reference will be built automatically in a process called auto binding. The reference resolves automatically as an auto-bound reference when you link the object to a scan group and DI object, without having to manually enter or script the reference.

The following is an example of an I/O reference string before the object has been assigned to a scan group and DI

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

object:

```
<IODevice>.Mixer.Tank.Inlet.InputSource
```

Once you assign the object to a scan group, the reference resolves to include the assigned Device Integration object and scan group. For example, assigning the object to the scan group "Fast" under DI object "OPC001" will change the reference to:

```
OPC001.Fast.Mixer.Tank.Inlet.InputSource
```

Important: Do not lock InputSource or OutputDest attributes when using I/O auto assignment. If either InputSource or OutputDest attributes, or both, are locked in the parent template, the attributes cannot be updated with the resolved auto-bound reference when the object is deployed, and the runtime value will be "---Auto---".

I/O auto assignment is configured in the **IO Devices view**. Use this view to associate application and system objects with DI objects and scan groups.

Scan groups are contained by DI objects and help categorize devices that are associated with them on the basis of how often their I/O points are scanned. For basic information about DI objects, see Device integration templates. For basic information about scan groups, see the OPCClient Object help file.

Note: A DI object will not appear in the **IO Devices view** unless at least one custom scan group has been defined for it.

Assign objects that need to be closely monitored or that have a faster rate of change to a scan group with a faster scan rate. Objects with slower rates of change can be assigned to a scan group with a slower scan rate. The scan rate is the rate at which the AVEVA runtime will perform scans. This may be different than the scan rate a PLC uses to monitor the hardware attached to it.

Note: Fast scan groups consume more network and computing resources than slow scan groups. In a large Galaxy, load balancing may require you to use multiple scan groups with similar scan rates.

The I/O references for the objects' attributes set to use I/O automatic assignment are dynamically generated when you assign an object to a scan group. The auto-bound reference replaces the placeholder string and is displayed in the IDE. It is generally in the following format:

```
<DIObject>.<ScanGroup>.<Object>.<Attribute>
```

`<DIObject>` and `<ScanGroup>` correspond to the name of the DI object and scan group to which you assign the object. For example, if the DI object name is `DI01` and the scan group name is `FastSG`, then the auto-bound reference will start with `DI01.FastSG`.

`<Object>` may consist of multiple elements if there are contained objects. The following is an example of an object string:

```
Tank3.Valve1.Attr2
```

Here, `Tank3.Valve1` corresponds to <object>. The complete auto-bound I/O reference would be:

```
DI01.FastSG.Tank3.Valve1.Attr2
```

Although you assign objects, references are generated for each I/O attribute associated with the object, and not for the object itself. Therefore, you may have to change the I/O references for a subset of an object's attributes. Once objects are associated with DI objects and scan groups, you can validate the auto-bound assignment for each of the object's attributes in the **Mapping view**, and if necessary, you can enter override values in this view.

Assign areas and objects to scan groups

The **IO Devices view** displays all application and system objects, as well as DI objects that have custom scan groups associated with them. Templates are not shown in this view, nor are DI objects that do not have custom

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

scan groups defined for them (DI objects with only the default scan group are not displayed in the IO Devices view).

Note: You cannot assign application and system objects to a default scan group; these assignments can only be made for custom scan groups.

In the **IO Devices view**, objects and areas are initially unassigned and placed in a flat view under the "Unassigned IO Devices" folder. You then select one or more of these unassigned objects or areas and drag and drop them onto a scan group. Deployed application and system objects cannot be assigned to a scan group. Undeploy them first.

Note: DI objects can remain deployed during the assignment process.

Once an object is attached to a scan group, the I/O references for the objects' attributes set to use I/O automatic assignment are automatically generated for you by concatenating the DI object, scan group, object (including contained objects), and attribute names.

You can assign multiple objects for auto-binding at once. Hold down the **shift** or **ctrl** key to select multiple objects and drag the selected items to a scan group. Object assignment is based on hierarchy and area inheritance, as follows:

- If all the objects you select are at the same hierarchical level, all contained (subordinate) objects that have not been assigned, except sub-areas, are also selected and will move to the same scan group.
  - To assign a sub-area, it must be specifically selected.
  - If some contained objects were previously assigned to a different scan group, these will retain their original assignments. To change an existing assignment, you must specifically select the assigned object and move it to a different scan group.
- If you select objects at different hierarchical levels for assignment to a scan group – for example, if you select an area, an object within the same area, and an object from another area – then only the selected items move to the scan group.
- If you select an area that contains another area, only the area selected will move. You must specifically select an area to assign it, regardless of its hierarchical level.

When an area or containing object is assigned to a scan group, the objects it contains are assigned as follows:

- Objects within the area that share the area's device linkage will be reassigned to the new device along with the area. This remains true even in the case where both the area and the contained objects are not currently linked to a device. Unassigned objects within the area will use the area's assignment.
- Objects within the area that are already assigned to a scan group that is different from that of the hosting area will retain their original assignments.
- If necessary, you can reassign contained objects to a different scan group after the initial assignment, or override assignments in the **Mapping view**.

  Note: You cannot change scan group or DI object assignments for deployed objects in the **IO Devices view**. You must undeploy the object first.

Deleting a scan group will delete the I/O assignments of any objects that were previously assigned to it, including override values.

Caution: Creating or deleting scan groups within a derived DI object template may overwrite I/O assignments when these changes are propagated to instances of the DI object template. If this happens, all I/O auto assignment information for objects linked to the updated DI objects, including override values, will be lost. The objects will move to the "Unassigned I/O Devices" folder.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

To assign an object or area to a scan group

1. In the **IO Devices view** or **Model view**, select application objects and areas for assignment.
2. Drag and drop the objects and areas to a scan group in the **IO Devices view**.
3. The object or area and all contained objects will be moved to the selected scan group.

   Note: If the area or object has contained objects that were previously assigned to a different scan group, these will retain their original assignments. Contained areas will not move and must be overtly selected.
4. The I/O references for each attribute set for I/O auto assignment will be dynamically configured and displayed as auto-bound in the **Mapping view**.

Rename application, system, and device integration objects

You can rename application, system and DI objects in the **IO Devices**, **Model** and **Deployment** views. If an area or object has already been assigned to a scan group, its I/O references are updated automatically.

Rename scan groups

Changes to scan groups (renaming, adding, or deleting them) can only be done by editing the DI object that contains the scan group in the Object Editor. Use the DI object's **Scan Group** page to make any necessary changes.

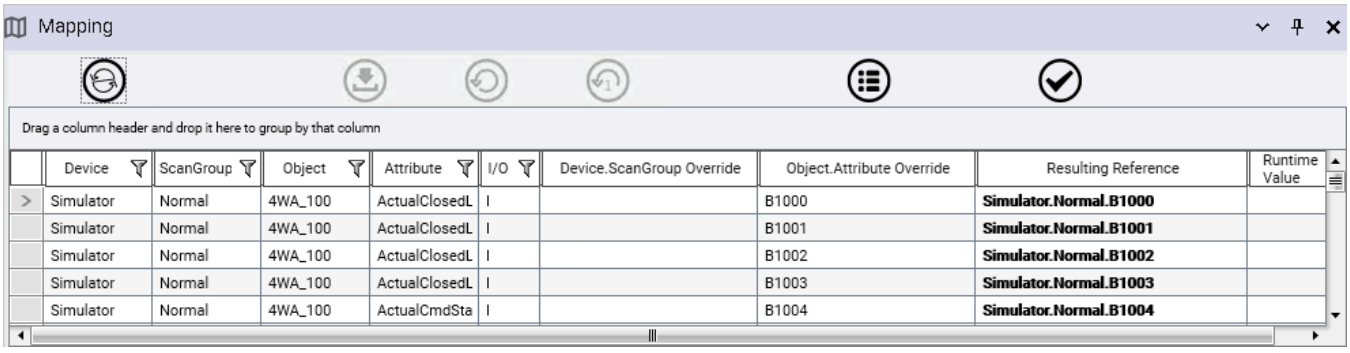
If scan groups are renamed at the template level, the changes will propagate to the DI object instances and I/O assignments will be preserved.

Note: You cannot rename or delete the default scan group.

Edit and validate I/O assignments

The **Mapping** view is a table that displays I/O auto assignment references for application and system objects that are linked to DI object scan groups. Only auto-bound references are displayed in the **Mapping** view. Application and system objects not yet assigned to a scan group, and manually configured references and are not shown. To be visible in the **Mapping** view, the object must be selected in the **IO Devices** view.

When you initially open the **Mapping** view after starting the IDE, the table will scroll so the far right column is in view.



Device	ScanGroup	Object	Attribute	I/O	Device.ScanGroup Override	Object.Attribute Override	Resulting Reference	Runtime Value
Simulator	Normal	4WA_100	ActualClosedL	I		B1000	**Simulator.Normal.B1000**	
Simulator	Normal	4WA_100	ActualClosedL	I		B1001	**Simulator.Normal.B1001**	
Simulator	Normal	4WA_100	ActualClosedL	I		B1002	**Simulator.Normal.B1002**	
Simulator	Normal	4WA_100	ActualClosedL	I		B1003	**Simulator.Normal.B1003**	
Simulator	Normal	4WA_100	ActualCmdSta	I		B1004	**Simulator.Normal.B1004**	

- Selecting a DI object in the **IO Devices** view lists I/O auto assignment attributes for all auto-bound objects linked to all scan groups under it.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

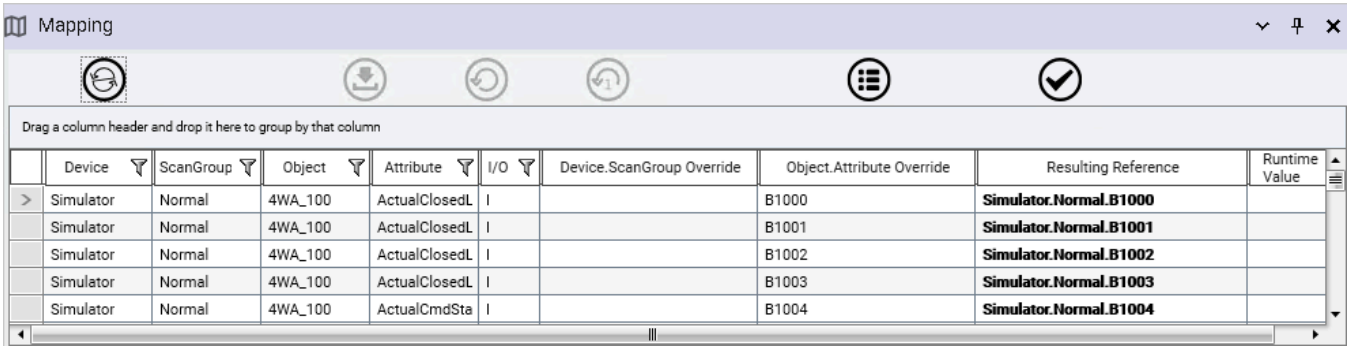
- Selecting individual scan groups restricts the scope of the information displayed in the **Mapping** view to the auto-bound objects that have been linked to the selected scan groups.
- Selecting individual application or system object further restricts the scope of attributes displayed to only those associated with the selected object.

Note: You can select multiple items in the **IO Devices** view. Selected items can be at different hierarchical levels. Selecting a subordinate object will exclude non-selected objects within the device hierarchy, even though the parent object is selected.

In the **Mapping** view, you can view and validate I/O references for each automatically generated attribute, and you can override the automatically generated I/O references. As is the case in the **IO Devices** view, you do not have to check out objects to change their I/O assignments.

How to use the mapping view for I/O auto assignment

The **Mapping** view is divided into columns, each of which displays information for an I/O attribute that has been auto assigned.



Column Heading	Definition
Device	Device Integration Object (DIO) name
ScanGroup	Scan group name
Object	Application object or area object name
Attribute	Attribute name
I/O	Input or Output
Device.ScanGroup Override	Override setting: enter an override value to replace the DIO and scan group that was automatically assigned
Object.Attribute Override	Override setting: enter an override value to replace the object and attribute names
Resulting Reference	Concatenated I/O reference string (for example, "DIO1.SG2.Mixer3.Attr5" or "DIO1.SG2.Conveyor.40001")

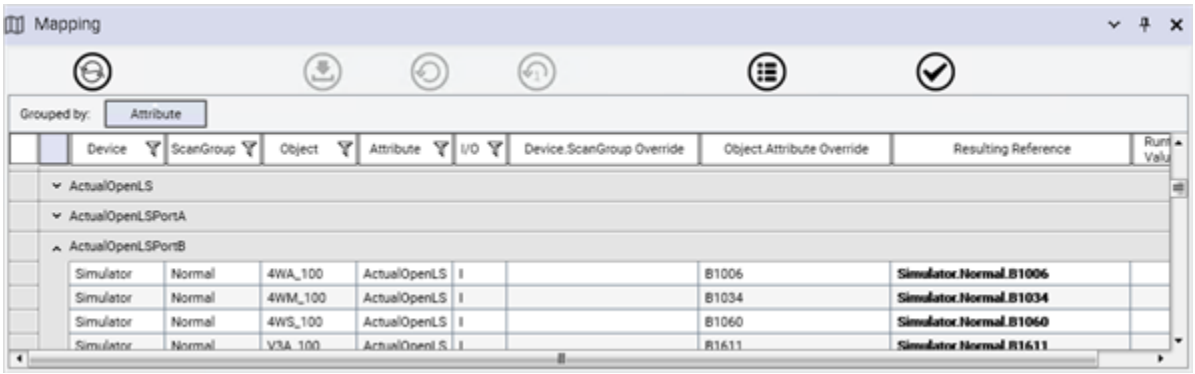
© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Column Heading	Definition
Runtime Value	Data returned from runtime when the **Validate References** button has been clicked

You can change how I/O attributes are displayed by sorting, grouping, or filtering the attributes.

Attributes are initially sorted by Device name (DI object). To sort attributes using a different column, click on the column heading once to sort in ascending order. Click a second time on the same column heading to sort in descending order. A third click on the column heading turns off sorting for that column.

To group objects by I/O assignment criteria, drag a column heading to the top of the table in the **Mapping** view. For example, if you drag the heading "Attribute" to the top of the table, I/O assignments will be grouped by attribute name. In the figure below, I/O assignments are grouped by attribute name. Note each attribute grouping has been expanded.



Note: Grouping is not available for the **Device.ScanGroup Override**, **Object.Attribute Override**, or **Resulting Reference** columns.

You can use more than one column heading to group I/O assignments. For example, you could group IO assignments first by attribute name and then by I/O type.

Columns can also be filtered to display only certain entries by clicking the filter symbol and then selecting the filtering criteria. When a filter is applied, the color of the filter symbol changes to orange.

Note: Filtering is not available for the **Device.ScanGroup Override**, **Object.Attribute Override**, or **Resulting Reference** columns.

The table contains additional information that you can reveal by clicking the **Show Details** button.

Details Column Heading	Definition
Attribute Data Type	Attribute type as defined by the attribute definition set in the Object Editor
Target Data Type	Attribute type of the target, as determined by the runtime
Runtime Quality	Data quality from runtime

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Validate I/O references

Use the **IO Devices** pane and the **Mapping** pane to populate the **Resulting References** column in the **Mapping** grid. You can then validate auto-bound I/O attributes that have been assigned in **Mapping**. To begin I/O reference validation, press the **Validate References** button, available in the **Mapping** pane.

Validating I/O references is separate and distinct from the **Validate** function available in the **Template folder** and in the application views. That **Validate** function validates the object's references configuration against the the Galaxy Repository. It is a configuration validation.

In contrast, validating I/O references is a runtime validation, and performs a one-time "get" from the runtime subsystem to retrieve the value and quality of the resulting references. A reference string in the **Resulting References** column is validated if a valid value and good quality are returned for that string. Objects should be deployed for an I/O reference validation to be meaningful.

Note: As a best practice, it is more efficient to validate each reference while performing I/O auto-assignment than it is to perform a single validation after configuring and deploying the entire application.

An estimate of the time needed for validation is displayed. The time is updated as validation progresses. When validation completes, the total number of unresolved I/O references is displayed along with the total number of auto-bound references checked, for example, "2 out of 5000 reference(s) failed to validate." To view references that did not validate during the validation period, click the "Show Advanced Columns" button. This will reveal the **Runtime Quality** column.

You can sort the **Runtime Quality** column to aggregate unresolved I/O references at the top of the list of references. See How to use the mapping view for I/O auto assignment for additional information on sorting. You can then move the associated objects to a different scan group or apply overrides. Recheck the references after completing any required reassignments and overrides.

Override I/O auto assignments

The **Mapping** view contains two override columns. To change an I/O assignment, enter the new parameter in one or both of the two override columns.

The information in other columns cannot be edited directly. You can also reassign an object to a different scan group in the **IO Devices** view, in which case the new assignment will be reflected in the **Mapping** view.

Note: If an object is already deployed, changing any of its I/O assignments in the **Mapping** view will change the object’s status from "Deployed" to "Deployed with pending configuration changes."

To override the DI object and scan group assignment that was made in the **IO Devices** view, enter the value you want in the column **Device.ScanGroup Override.** Naming restrictions for the Device.ScanGroup Override value are as follows:

- The value must contain one DI object name and one scan group name contained by the DI object, separated by a period (.).
- The value cannot start or end with a period.
- The value cannot have any leading spaces.
- The I/O reference cannot be longer than 329 characters.

To override the object and attribute name, enter the value you want in the column **Object.Attribute Override**. Naming restrictions for the Object.Attribute Override value are as follows:

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

- The value cannot have any leading spaces.
- The I/O reference cannot be longer than 329 characters.

You can change attribute information using the Object Editor (after checking out the object), instead of overriding values in the **Mapping** view. However, if you enter the specific I/O reference for an attribute in the Object Editor (manual configuration), it will no longer appear in the **Mapping** view. Only attributes configured for I/O automatic assignment appear in this view.

Batch overrides of I/O assignments

If you have a large number of overrides to enter, information can be copied and pasted between the **Mapping** view and Microsoft Excel. However, you must be careful when moving data to and from the **Mapping** view. Observe the following conditions:

- Only data in the two override columns is editable. All other columns are read only.
- Even though you can only change the override columns, always copy all columns. This will help you verify that data is being copied correctly when you paste back into the **Mapping** view.
- When you copy data from the **Mapping** view, column headers are automatically included to help you while editing in Excel. When you copy your edited data back to the **Mapping** view, do not copy this header row, only copy the data.
- If you need to sort attributes, sort them in the **Mapping** view before you copy them into Excel.
- Do not change the sort order of the attributes in the **Mapping** view until you have pasted the edits made in Excel and applied them.
- Do not sort the attributes in Excel. Attributes can only be sorted in the **Mapping** view and the sort order cannot change until your edits have been applied.

  WARNING! The Mapping view does not verify data that is copied to it. Therefore, if the order of attributes is different in Excel than it is in the Mapping view, the I/O references will be broken. Once changes are applied, there is no way to undo them.

To validate and edit I/O assignments

1. Click the **Validate References** icon to check that the I/O assignments in the **Mapping** table are valid. This action retrieves data and quality information from the runtime data acquisition subsystem.

   Important: I/O assignments can only be validated if the Platform, AppEngine, and associated DI object have been successfully configured and deployed.
2. Check the data returned in the **Runtime Value** column. If the assignment is not valid or if the quality is bad, no data will be returned (the **Value** column will be empty), and "Not Connected" will be displayed in the **Runtime Quality** column (click the **Show Details** button to reveal this column).
3. To isolate invalid assignments, select the filter icon at the top of the **Runtime Quality** column filter for "Not Connected" messages.
4. Edit the I/O assignment of objects as needed.
   - To override a scan group and DI object assignment, enter the new value in the **Device.ScanGroup Override** column. The override value for this column cannot contain more than one period (.) to separate the DI object name from the scan group name.
   - To override an object and attribute name, enter the new value in the **Object.Attribute Override** column.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

5. Click **Apply Pending Overrides** to save your changes.
   - To discard the last unsaved override setting, click **Undo Last Pending Override**.
   - To discard the all unsaved override settings, click **Undo All Pending Overrides**.

Upload runtime configuration changes for auto-assigned objects

If changes are made during runtime to an auto-assigned I/O reference, you can upload these changes as overrides to the original assignments. Invalid changes are ignored, and an error message is sent to the OCMC Logger.

- Runtime changes to scan groups or DI objects are saved as **Device.ScanGroup** overrides.
- Runtime changes to objects or attributes are saved as **Object.Attribute** overrides.

When you look at the object in the **IO Devices view**, you will see that the original I/O auto assignment setting is retained and the object remains auto assigned to its original DI object and scan group.

The changes uploaded from runtime will appear in the **Resulting References** column of the **Mapping view**. The same information from runtime will appear when you examine the object's I/O attributes in the **Object Editor**. These changes are preserved across subsequent redeployments.

To upload runtime configuration changes

1. From the **Model** or **Derivation** view, select the object(s) with runtime changes you want to preserve.
2. Click **Upload Runtime Changes** from the **Object** menu. The **Upload Runtime Changes** dialog box will appear and provide status of the upload process.

I/O auto-assignment workflow example

This section describes the basic steps for implementing I/O auto-assignment, from start to finish. The basic steps are:

1. Use the **Attributes** page of the **Object Editor** to prepare system objects and application objects for I/O auto assignment.
2. Use the **Object Editor** to configure DI objects and scan groups.
3. Use the **IO Devices** view to assign system and application objects to scan groups (objects must be undeployed).
4. Use the **Mapping** view to validate I/O references. You can enter overrides in this view, if necessary.

To prepare objects for I/O auto assignment

1. Create a derived template from the $UserDefined base template and open it in the Object Editor.
2. Add attributes as needed. For each attribute that requires input or output, click the I/O button.

   The default I/O mode is **Read/Write**, and the default string indicating that the attribute is primed for I/O auto assignment appears in the **Read from / Write to** field.
3. Change the I/O mode as needed.

   Important: Do not change the default string if you want to use auto assignment for the attribute. The default

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

string indicates that auto assignment is not yet complete and the object is not linked to a PLC.

4. Save and check in the template, then close the Object Editor.
5. In the **Template folder**, create instances from the UDO derived template. The instances appear in the **Model view**, under Unassigned Area.

To configure DI objects and scan groups

1. In the **Template folder**, create a Device Integration (DI) object from the $OPCClient, $DDESuiteLinkClient, or $Redundant DIObject template. The DI object appears under "Unassigned Area" in the **Model view** and under "Unassigned IO Devices" in the **IO Devices** view.
2. Add one more scan groups to the DI object you created in the previous step. The naming convention for the scan groups must conform to the naming convention of the PLC with which you are connecting.

   The scan groups will show as belonging to the DI object in the **IO Devices view**.

To assign system and application (user created) objects to scan groups

1. Create a new area and set it as the default. Do not deploy the area yet.
2. In the **IO Devices view**, drag and drop the area to a scan group.
3. Create new application objects. As you create new objects, they will appear in the **IO Devices view** under the default area.

   The IDE will complete the references for each I/O attribute for you.
4. If you need to change the scan group assignment for an object, drag and drop it to the new scan group.

To assign system and application (user created) objects to scan groups - alternate method

1. Create a new area and assign it to a scan group.
2. Create new application objects. These will appear under the Unassigned folder in the **Model view**.
3. Assign the application objects to the area assigned to the scan group in step 1. All application objects assigned to the area will inherit the area's linkage to the DI object and scan group.
4. If you need to change the scan group assignment for an object in the area, drag and drop it to the new scan group.

To validate I/O references (and override auto assigned values, if necessary)

1. Open the **Mapping** view. This view opens automatically if you select an object, scan group, or DI object in the **IO Devices** view.

   Each row in the **Mapping** table contains the I/O reference for one attribute.
2. Enter an override value for the attribute in one of the override columns.

   Note: The **DI Object.Scangroup Override** column can only contain the DI object name and scan group name, separated by a period (.).
3. Deploy the platform and associated DI objects.
4. Click the **Validate References** button to check that communication is established with the PLC.
5. Deploy the objects.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

I/O auto-assignment with OPC clients

Using I/O auto-assignment with the OPC communication protocol may require that you add overrides to allow communication between the IDE and certain PLCs, such as Allen-Bradley Logix and Siemens S7 controllers.

If you are using auto assignment to communicate with a device connected to an OPC Server on a different node (for example, a Communication Driver on a remote Galaxy), you will use overrides to build valid I/O references with all required parameters to fulfill the OPC hierarchical path. If bulk edits are required, you can copy and paste between the **Mapping** view and Microsoft Excel.

I/O auto assignment uses the syntax: *<DI Object.ScanGroup>. <Object.Attribute>*. It is the first part of the syntax that limits how the OPCClient builds the I/O auto assignment reference, since additional parameters in the device hierarchy, such as Ethernet ports and backplanes are not included. While these parameters may be logically viewed as part of the *DI Object.ScanGroup* reference, you must add the parameters to the **Object.Attribute override** column (not the **DI Object.ScanGroup override** column) to create the fully qualified I/O reference.

Using DDESuiteLink instead of OPC will accommodate the automatically generated I/O reference to these PLCs, and in most cases, overrides will not be needed.

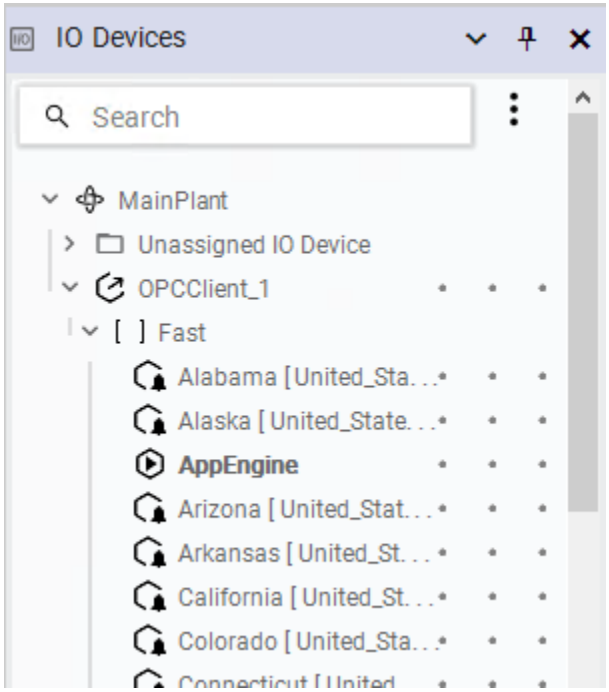
Caution: Do not change the order of references, either in Excel or in the **Mapping** view by sorting, grouping, or filtering the reference list while editing references.

Override scenario 1 — Siemens S7 controllers with an OPCClient object

The following scenario illustrates using I/O auto assignment with a Siemens S7 PLC. The Siemens S7 PLC in this example scenario includes a TCP/IP port that is in addition to the OPCClient object and scan group names captured by I/O auto assignment in the **IO Devices view**.

To configure the Siemens S7 PLC with the OPCClient object

1. Select the OPCClient object or scan group that contains the PLC in the **IO Devices view**.



© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

I/O attributes for each object associated with the OPCClient or scan group (depending on which is selected in the **IO Devices** view) are shown in the **Mapping** view.

2. In the **Mapping** view, the resulting reference will be in the form:
   `OPCClientName.ScanGroupName.ObjectName.AttributeName`

   Example (**Resulting Reference** column):

   `OPCClient_01.SG1.UDO_01.Attribute_001`

3. Do not edit the **Device.ScanGroup Override** column (leave it blank). This leaves the OPCClient Object and scan group assignment as defined in the **IO Devices** view.
4. Enter the fully qualified field reference in the **Object.Attribute Override** column. For example:

   `S7C_1.S7C_200_1.UDO_01.Attribute001`



5. Apply the override. The resulting reference will be displayed. For example:

   `OPCClient_01.SG1.S7C_1.S7C_200_1.UDO_01.Attribute001`

Override scenario 2 — Allen-Bradley CIP controllers with an OPCClient object

The following scenario illustrates using I/O auto-assignment with a Allen Bradley Logix PLC. The Allen Bradley Logix PLC in this example scenario includes a device hierarchy with a port object, ethernet port, backplane, and controller. These are in addition to the OPCClient object and scan group names captured by I/O auto assignment in the **IO Devices view**.

To configure the Allen-Bradley Logix PLC with the OPCClient object

1. Select the OPCClient object or scan group that contains the PLC in the **IO Devices** view.

   I/O attributes for each object associated with the OPCClient or scan group (depending on which is selected in the **IO Devices** view) are shown in the **Mapping** view.

2. In the **Mapping** view, the resulting reference will be in the form:
   `OPCClientName.ScanGroupName.ObjectName.AttributeName`

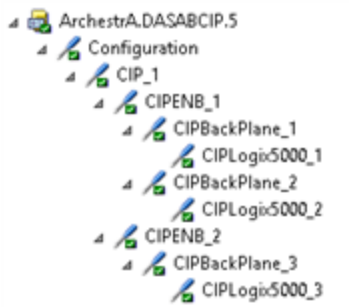
   Example (**Resulting Reference** column):

   `OPCClient_01.SG1.UDO_01.Attribute_001`

3. Do not edit the **Device.ScanGroup Override** column (leave it blank). This leaves the DI Object and scan group assignment as defined in the **IO Devices** view.
4. Enter the fully qualified field reference in the **Object.Attribute Override** column. For example:

   `CIP_1.CIPENB_1.CIPBackPlane_1.CIPLogix5000_1.UDO_01. Attribute001`

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.



5. Apply the override. The resulting reference will be displayed. For example:

   `OPCClient_01.SG1.CIP_1.CIPENB_1.CIPBackPlane_1. CIPLogix5000_1.UDO_01.Attribute001`

Write and edit scripts

You can write scripts that run commands and logical operations based on specified criteria being met to extend and customize your objects. For example, a script can be configured to start running when a key is pressed, a window is opened, or the value of an attribute changes.

With scripts, you can create a variety of customized and automated system functions. A script adds behavior that runs when the object that contains the script is deployed and the object is either:

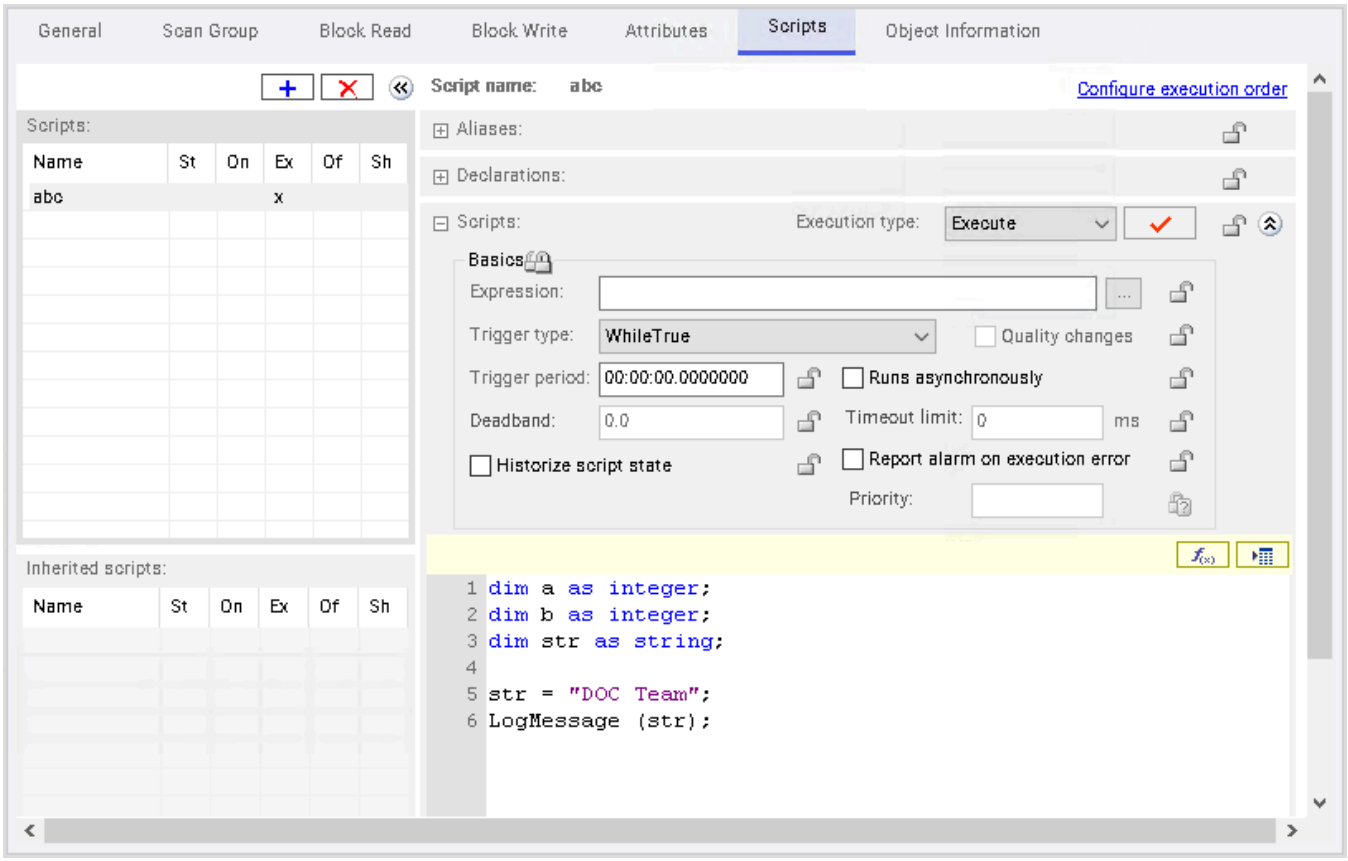
- On scan in the runtime environment or
- Changes scan or start/shutdown state

A script typically runs based on attributes of the object that contains it, but can be started by another script based on changing values of attributes of more than one object.

When a script condition is true, the script runs at least once immediately. The maximum length of a script trigger period is 49-days. A script never runs if the trigger period exceeds 49-days.

For specific information about writing scripts, including the scripting language, syntax, commands, and using .NET, see the *Application Server Scripting Guide*. For more information about the scripts page, see Scripts page.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.



Important: You cannot pass attributes as parameters for system objects. Instead, use a local variable as an intermediary or explicitly convert the attribute to a string using an appropriate function call when calling the system object.

About object scripts

The following characteristics apply to the scripting environment:

- Script text has no length limitations.
- Selecting a script function from the **Script Function Library** dialog box adds it and its syntax to the script text where you can edit it.
- You can save a script with syntax errors, but the object cannot be deployed until you correct the script syntax errors.
- You can validate your scripts before using them. This helps you avoid syntactically correct but semantically incorrect combinations such as two statements declaring the same variable. Variables can be declared only one time in a single block.
- You can change the name of a script at any time by renaming it in the Object Editor.
- In the runtime environment, a script execution error stops the script's current execution. Script execution is retried on the next AppEngine scan.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Script execution

The existence and execution order of scripts associated with an object are inherently locked at each stage of development in the template, derived template, and instance. For example, a set of scripts associated with a template are treated as an ordered block in the **Configure Execution Order** dialog box when configuring execution order in a next-generation derived template.

New scripts in the derived template can be run in any order before and after the template's block of scripts. The derived template's execution order is treated as a block in any downstream derived templates or instances. Scripts cannot trigger any faster than the scan period of the AppEngine the script is associated with or faster than the scan period of the AppEngine that hosts the object that the script is associated with.

Scripts run in one of two modes:

- Synchronous scripting mode is the default for running scripts in the runtime environment. This mode runs scripts in order while an object is running on scan.
- Asynchronous scripting mode is a group of scripts running on the same, lower priority execution thread. These scripts only support Execute triggering and run independently from each other. Set the maximum number of independent threads in the AppEngine configuration editor. To use either scripting mode, you must select **Execute** as the **Execution Type** in the **Scripts** area on the **Scripts** page.

To create a script and associate it with an object

1. Add a script. On the **Scripts** page of the Object Editor, click the **Add** (**+**) button. A script is added to the **Scripts Name** list.
2. Type a name for the script and press Enter. Script names can be up to 32 alphanumeric characters, including periods. At least one character must be a letter.

   Note: For detailed information about each item on the **Scripts** page, see Scripts page.
3. Select a trigger that starts the script in the runtime environment.

   Execution Type triggers include: Startup, On Scan, Execute, Off Scan and Shutdown.

   - If you select **Startup**, **On Scan**, **Off Scan**, or **Shutdown**, the **Basics** group is unavailable. The script is triggered when the object starts, goes on scan, goes off scan, or shuts down.

     If you select **Execute**, the **Basics** group is available.

     If you selected **Execute** as the script trigger, select a **Trigger Type**. Depending on the type selected, you are required to enter an **Expression** and/or **Trigger Period** and **Deadband** values. When the combination of **Expression**, **Trigger Type**, **Trigger Period** and/or **Deadband** is satisfied in the runtime environment, the script starts running. See the following table for more information.

     The **Trigger Period** format is as follows:
     Hours:Minutes:Seconds:Milliseconds

     For example, 3 hours, 5 minutes, and 10.5 seconds is:
     03:05:10.5000000

     Expressions are limited to one language statement in length and calling only synchronous mode script functions. Avoid using script functions with side effects in expressions because subtle behaviors can occur.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Trigger Type	Description
Periodic	When the object containing the script is going On Scan, a Periodic script evaluates its expression at the next scheduled scan period of the AppEngine. The script then runs periodically at the trigger interval specified in the **Trigger Period** box. A time interval of zero (0) starts the script during every scan. This trigger does not require an expression.
While True	When the object containing the script is going On Scan, a While True script evaluates its expression at the next scheduled scan period of the AppEngine. The script runs if true and then periodically thereafter at the trigger interval. The script continues running as long as the **Expression** value evaluates to true. A Trigger Period is required. Zero (0) evaluates the expression at the AppEngine scan period and non-zero means the expression is evaluated at the specified time interval.
On True	When the object containing the script is going On Scan, an On True script evaluates its expression at the next scheduled scan period. The script starts at the transition between the expression going from false to true.
On False	When the object containing the script is going On Scan, an On False script evaluates its expression at the next scheduled scan period. The script starts at the transition between the expression going from true to false.
Data Change	Scripts run when the value or quality of the expression changes. The expression must evaluate to a single, non-arrayed value of the following types: integer, real, time, elapsedtime, string, double, Boolean, custom enumeration and quality. To allow execution based on quality, select the **Quality changes** check box. A deadband can be specified for all data types. Deadband units for time and elapsedtime types are milliseconds. Deadband is always ignored for strings because any change (even from "ABC" to "abc") is considered a change. Only major changes in quality (from Good/Uncertain to Bad/Initializing or vice versa) are considered changes.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Trigger Type	Description
	After the object is put on scan, Data Change-triggered scripts start running at the AppEngine's next scan period and then at each subsequent scan period in which the value or quality changes.
While False	When the object containing the script is going On Scan, a While False script evaluates its expression at the next scheduled scan period, runs if false, and then periodically thereafter at the trigger interval. The script runs as long as the **Expression** value evaluates to be false. A Trigger Period is required. Zero (0) evaluates the expression at the AppEngine scan period and non-zero means the expression is evaluated at the specified time interval.

4. Select one or more of the following:
   - Set the **Runs Asynchronously** and associated **Timeout Limit** parameters, as needed. The Timeout Limit is set in milliseconds. Note that if a script exceeds the Timeout Limit, it continues to execute all sequential statements, but generates a warning in the logger when it completes. If the script contains a loop, it will exit the loop and complete execution of the script, when the timeout limit is reached.
   - Select **Report Alarm on Execution Error** and set a **Priority** for the alarm if you want the alarming function to alert you if a script execution failure occurs.
   - Select **Historize Script State** to store the state of the script in your application's Historian.
5. In the **Declarations** area, type variable declarations about the script you are writing.
6. Set aliases for the reference strings in the **Aliases** area. This can simplify the script code and allows script code to be created and locked at a template level using alias names. When an individual instance of that template is created, you can link external attributes to the alias names.

   In the **Aliases** area, click the **Add** button to add a new alias. An alias is added to the list. The name is shown in edit mode. Double-click the **Reference** entry, and enter a reference string for the alias. You can also click the **Browse** button at the end of the **Reference** block to open the Attribute Browser for easy selection of an object's attributes.
7. Write the script in the **Script Creation** box. Use the **Display Script Function Browser** and **Display Attribute Browser** buttons to help you insert script functions and object attribute references in your script. For help

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

with the specific commands and syntax, see "QuickScript .NET Functions" in the *Application Server Scripting Guide*.

Click the **Validate Script** button to validate if your script contains any syntax errors.

8. Order the scripts. If you have more than one script associated with a single object, click **Configure Execution Order**. Ordering does not apply to asynchronous scripts. If a script is added to an instance derived from a template that contains scripts, the new script automatically defaults to running after the derived scripts.
9. When you are done creating your script and setting its execution triggering parameters, save and close the Object Editor.

Script locks and change propagation

When you lock a script in a template, the following rules apply:

- The name of a script and its existence is implicitly always locked. This means:
  - You cannot delete the script in derived objects.
  - You cannot change the name of the script in derived objects.
  - If you rename the script in the template, the name changes in all derived objects.
  - If you delete a script from the template, the script is deleted from all derived objects.
  - If you add a script to the template, the script is added to all derived objects.
  - You can add scripts to derived objects. Adding scripts to derived objects does not affect parent object scripts.
- You can lock or unlock the script text in a template. There is script text for **Declarations, Execute, Startup, Shutdown, On Scan** and **Off Scan**.
- You cannot separately lock a script in the script editor. A single group lock is used to lock or unlock all scripts at once. Once a script is locked, derived templates and instances cannot modify any of the script text.
- You can lock alias names and alias references separately.
  - When you create aliases for a script, an AliasReference attribute is created for each script with aliases. To lock alias references, you must lock the AliasReference attribute.
  - Changes to alias names only propagate to derived objects when aliases are locked in the **Scripts** page.
  - Changes to alias references only propagate to derived objects when the AliasReference attribute is locked in the **Attributes** page.
  - If the AliasReference attribute is not locked, the alias references will be editable in derived objects, but changes made in the parent template will not propagate to derived objects.
- The script description, runs asynchronous flag, expression, trigger type, trigger period, deadband and execution error alarm are individually lockable and can be locked separately from the script text. A group lock is provided for this group of attributes.
- When you add a script to a template, all properties of the script are editable.
- When you add a script to an instance, all properties of the script are editable, except the lock properties (locks do not serve any purpose in instances since you cannot derive child objects from them).

Important: An expression typically uses attribute references. If you lock the expression and the associated script in a template, use aliases in both the expression and the script. This allows you to specify the attributes that the aliases point to on a per-instance basis while the script code is locked.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

The following rules apply to the derivation behavior of locked script attributes:

- If an attribute is locked in a template, then all templates and instances derived from the template share the copy of the value of the locked attribute. A change to the value is only allowed in the template that locked it. The change propagates to all derived templates and instances.

  For scripts, locking an attribute of the script, such as its script text or execution type, in a template means all derived templates and instances point to that locked attribute. Future changes to that locked attribute's value, such as modifying the script text, propagate and appear in all derived templates and instances.

  If instances are deployed, they are marked pending update status. After they are redeployed, the change to the locked attribute in the template exists in the deployed instance.
- If an attribute is not locked in a template, then all templates and instances derived from that template receive their own copy of the value of that unlocked attribute. A change to that unlocked value is allowed in derived templates and instances because they own their own copy. Any change to the unlocked attribute value in the template does not propagate to any derived template or instance.

  An unlocked attribute in a script (such as expression or script order) in a template means that all derived templates and instances have their own copy and the value of that unlocked attribute can change. Future changes to that locked attribute's value (for example, modified expression) in the template does not propagate to any derived template or instance. If instances are deployed, their status does not change to pending update. Redeploying them does not cause the value to change in the deployed instance.

Create and work with content

The **Content** pane of the **Attributes** page lets you:

- Add, modify, rename or delete object-owned graphics. Object-owned means that the graphic exists only for that particular object. Note that object-owned graphics are propagated to child objects.
- Link the object to content (graphics, layouts, External Content) in the Visualization folder. This is shared content, that can be linked to multiple objects. Links to content are propagated to child objects.
- View content inherited from parent templates. However, you cannot add, modify, or delete inherited content. To edit an inherited content item, you must check out and edit the derived template that owns the content.

You must have the derived template or object instance checked out in order to add, modify, or delete content associated with the object; otherwise, the content is read-only and you can only view it.

You can only add content to derived templates or instances, not to a base template.

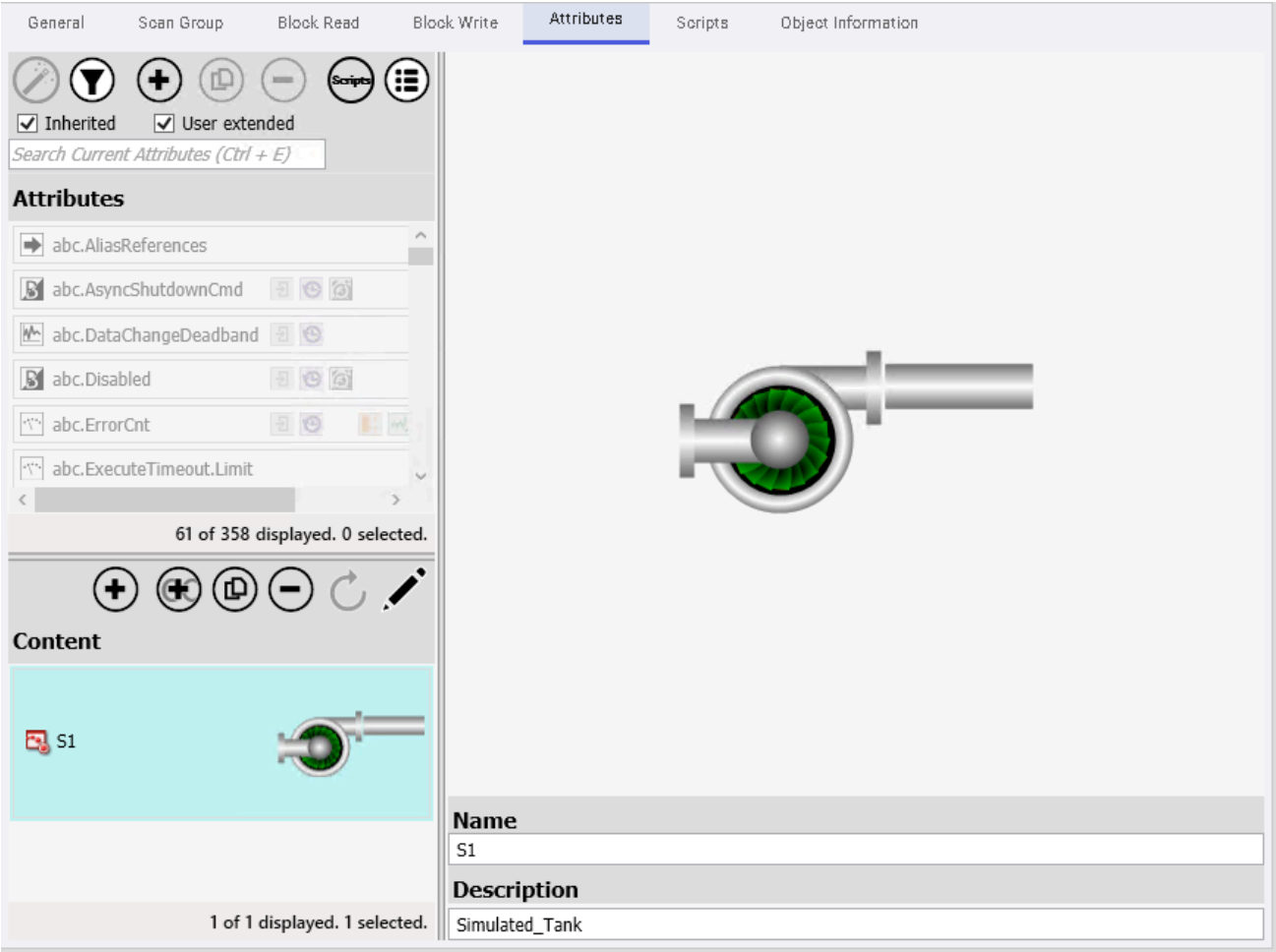
To add or link to content:

Linked content can be a graphic, an external content item, or a layout.

- To link to a content item in the Visualization folder to the object, select the **Link Content** button. This opens the **Galaxy Browser**. Then, select the graphic, layout, or External Content item to be linked to the object. Linked content can be shared by multiple objects. Relative references within the linked content will resolve correctly.
- To create an **object-owned graphic**, select the **Add Content**  button.
- To edit an owned-graphic, select the graphic and click the **Edit**  button.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Note: If you edit a linked graphic, layout or other shared content, the changes will appear in all objects that link to that content.



Add a graphic to an object

This procedure adds an owned graphic to the object. To link a graphic or external content item, see Link to shared content in the visualization folder.

1. From the **Content** pane in the **Attributes** page of the Object Editor, click the **Add** (+) button. A new graphic name and graphic icon are added to the pane.
2. Select the new graphic in the **Content** pane.
3. In the **Name** field, edit the new local graphic name.
4. In the **Description** box, type a description for the graphic being added.
5. Click the **Edit** icon. The Graphic Editor opens. For instructions on using the Graphic Editor, see "About Creating and Managing Industrial Graphics" in the *Creating and Managing Industrial Graphics User Guide*.

Errors and warnings in an owned graphic

When you save an owned graphic that you have edited in the **Graphics Editor**, the owning object is also saved. If

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

there are any errors and warnings, they will be shown in the **Save Confirmation** dialog. However, the errors and warnings include not just the graphic you are saving, but because the owning object is also being saved, the dialog lists all errors and warning related to the owning object and other graphics owned by the object.

To determine if the warning or error is related to the graphic you are saving, or if it is related to the owning object (or other graphic owned by the object), check the prefix in the warning/error message. The first (left-most) entry in each error/warning message lists the graphic or object name.

Edit a graphic

All modifications made to graphics referenced by other objects are visible in the referenced objects.

To edit a graphic

1. On the **Attributes** page of the Object Editor, in the **Content** area, select the graphic to modify.
2. Click the **Edit Content** icon. The Graphic Editor opens, showing the selected graphic.
3. Make changes. See "About Creating and Managing Industrial Graphics" in the *Creating and Managing Industrial Graphics User Guide*.

Set graphics properties with a galaxy style library

To view and manage the settings for the Galaxy Style Library, select **Configure** on the the **Galaxy** menu. A Galaxy Style Library lets you set the appearance and behaviors of different graphics properties within a ViewApp at runtime.

- Use element styles to modify graphics — define graphics properties such as font size, font color, fill color, line weight, and line color.
- Use quality and status indicators to provide runtime feedback — provide visual cues about different quality and status states.
- Use UI themes to set the appearance of WPF controls — define graphics properties for WPF and navigation controls used in ViewApps.

Use element styles to modify graphics

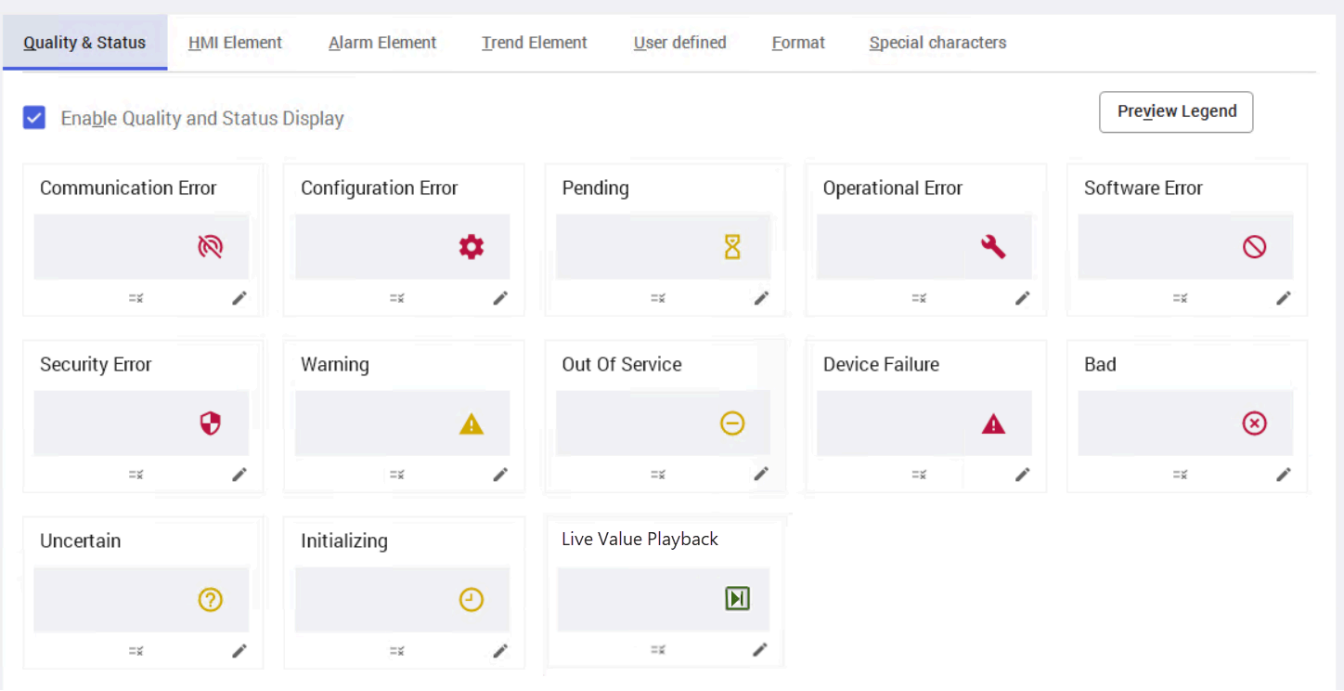
Element styles define one or more visual properties such as fill, line, text, blink, outline, and status properties of graphic elements. You can apply an element style to a graphic element to set preconfigured visual properties defined in that element style. Element styles establish consistent visual standards for graphics.

For more information about element styles, see Apply element styles to elements.

Use quality and status indicators to provide runtime feedback

In the same way that element styles can be used to preconfigure consistent visual properties of graphic elements, Quality and Status (QS) indicators can be preconfigured to provide visual elements specific to quality and status states at runtime. Only the highest priority QS indicator is shown at runtime. Therefore, if two or more indicators are active, only the highest-priority indicator is shown. For example, if there are both operational and software errors, only the operational error is shown.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.



The indicators are listed in order of priority, with the highest priority indicator listed by row from left to right, and then top to bottom. You can configure text, fill, line, outline, blink, and associated display icons for the following QS indicators:

- Communication Error
- Configuration Error
- Pending
- Operational Error
- Software Error
- Security Error
- Warning
- Out of Service
- Device Failure
- Bad
- Uncertain
- Initiating
- Live Value Playback (used for the AVEVA OMI Historical Playback feature)

Use UI themes to set the appearance of WPF controls

UI Themes define the color settings that are applied to WPF controls used in ViewApps. In order to set the colors for a WPF control, the control must comply with the Material Design Spec. See the MSDN WPF Community Projects website for additional information.

The UI Theme includes the following configurable parameters:

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

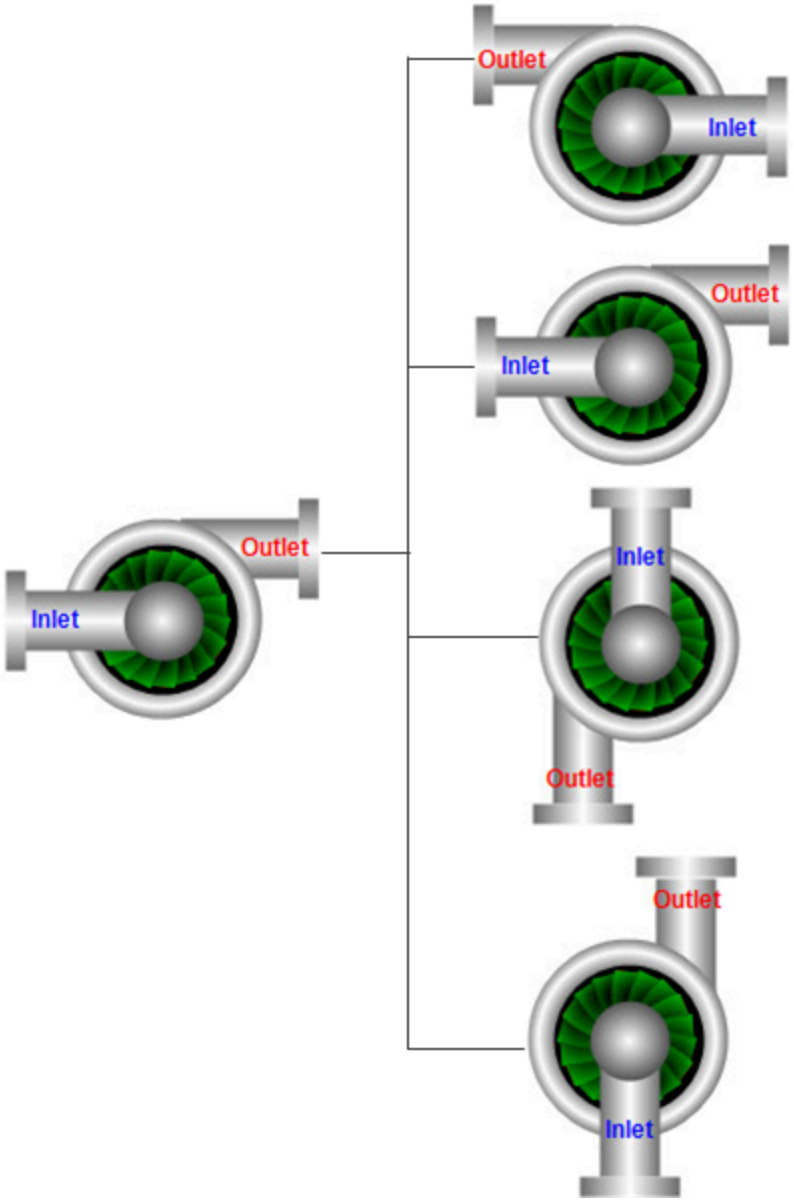
- **Theme** (light or dark): default – **light**
- **Background** color
  - Default **light** background color: grey, lightest setting
  - Default **dark** background color: blue-grey, second darkest setting
- **Primary** color – default **teal**
  - **Low.** This setting is only used by WPF controls that explicitly include the Primary Low property. The **low** setting is always darker than the **mid** setting. Default is the second darkest setting (second from left).
  - **Mid**. This setting is used by all WPF controls that use the Material Design Spec. Default is the middle setting.
  - **High**. This setting is only used by WPF controls that explicitly include the Primary High property. The **high** setting is always lighter than the **mid** setting. Default is the second lightest setting (second from right)
- **Accent** color – default **purple** (darkest setting)

Symbol Wizards

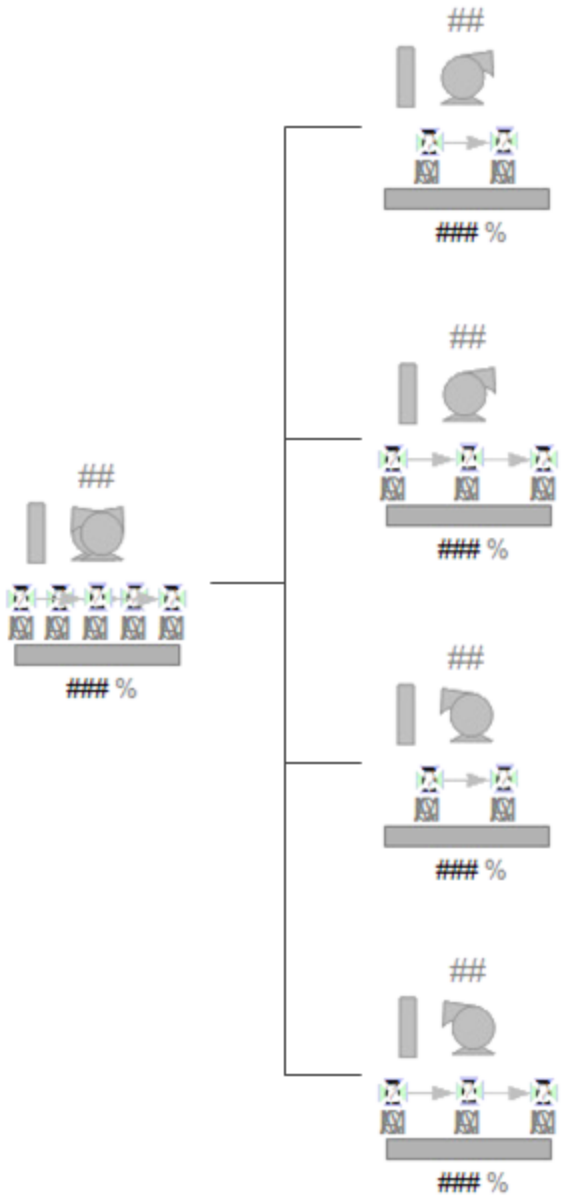
The Industrial Graphic Editor includes the Symbol Wizard Editor. Use this editor to create symbol wizards. A Symbol wizard contains multiple visual or functional configurations. A symbol wizard can be embedded into InTouch or OMI applications in the same way that you embed any Industrial Graphic.

Users select options from the symbol wizard to meet the requirements of an application. Incorporating multiple configurations in a single symbol wizard reduces the number of graphics needed to develop an AVEVA application.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.



© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.



The Symbol Wizard Editor can create symbol wizards from traditional Industrial Graphics and Situational Awareness Library graphics. Both types of graphics are located in the Visualization folder in separate folders. Symbol wizards are saved in the Visualization folder of the IDE and are not associated with any specific AVEVA object template or object instance. Except for the ability to select a specific graphic configuration, symbol wizards behave like standard Industrial Graphics.

Situational Awareness Library graphics provide an additional benefit of including defined properties and their associated attributes to more easily create configurations. Some Situational Awareness Library graphics include a Type property to assign a specific function for a configuration. For example, a meter graphic can be configured to represent a thermometer, a pressure meter, or flow meter by simply changing the attribute assigned to the Type property.

Typically, the process of creating and embedding a symbol wizard in an application requires the involvement of a Designer and a Consumer. A Designer creates symbol wizards using the Symbol Wizard Editor. A Consumer selects a configuration of a symbol wizard and embeds the instance of the graphic into an InTouch or OMI application.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Create Symbol Wizards

A Designer uses the Symbol Wizard Editor to define the various required graphic configurations based on a set of rules and graphic layers. A Designer defines a set of layers, which are used to group a set of graphic elements, custom properties, and named scripts. Graphic elements and other properties can be assigned to no layers or multiple layers. Graphic elements that are not assigned to any layer always appear in all configurations.

A Designer can create a rule for each layer that defines the conditions when the layer is included in a graphic configuration. Rules are assigned with Choice Groups, Choices, and Options. A Designer selects a configuration to be the graphic default that appears when the graphic is embedded in a managed InTouch application.

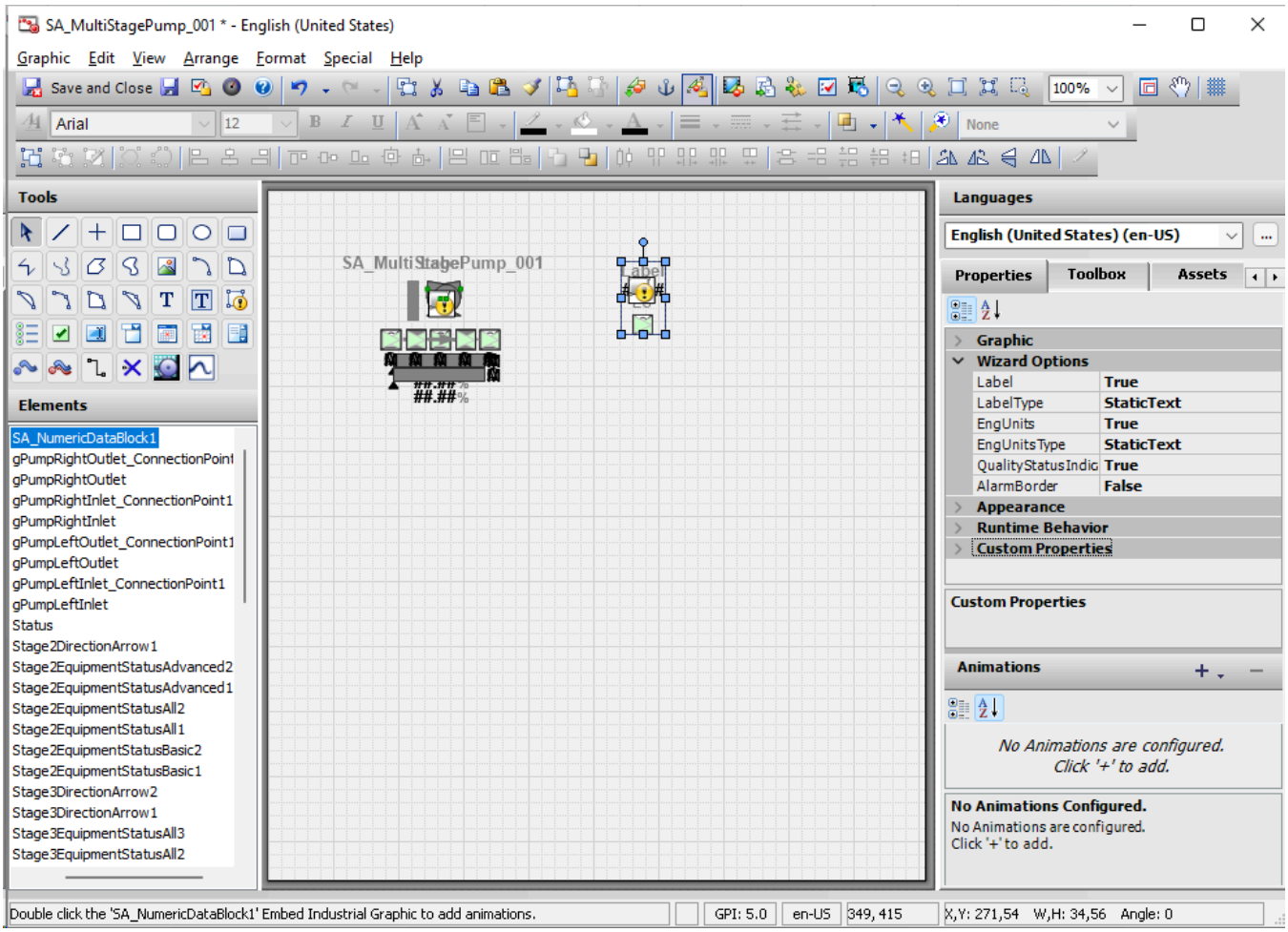
After creating all configurations, the Designer verifies each configuration of a graphic using the Symbol Wizard Preview. Designers set values in the **Wizard Options** view to verify that each configuration appears as designed based on the layer rules set for the graphic.

When a Symbol Wizard is finished, the Designer saves it to the Galaxy library so that it is available for use in InTouch and OMI applications.

Embed a Symbol Wizard into an application

Symbol wizards are stored in a Galaxy library just like standard Industrial Graphics. When a Consumer selects a symbol wizard and embeds it into a InTouch or OMI application, the default configuration of the Symbol Wizard is selected. The Consumer can change the graphic's configuration by changing the options from the **Wizard Options** section of the **Properties** view. Depending on the selected configuration, there can be additional configuration-related properties that can be selected by the consumer.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.



After selecting a graphic configuration and changing any properties, the Consumer saves the Symbol Wizard so that it can be embedded into a window from WindowMaker or into an OMI pane.

While the InTouch or OMI application is running, the Symbol Wizard appears as the configuration selected by the Consumer. A Symbol Wizard configuration cannot be changed during runtime.

For more information about how to embed Symbol Wizards into InTouch applications, see Using Symbol Wizards in an Application in the Industrial Graphic Editor help.

Rename a graphic

To rename a graphic:

1. On the **Attributes** page of the Object Editor, in the **Content** area, select the graphic to rename.
2. In the **Name** field, type the new name and press **Enter**. The new name is saved.

Delete a graphic

Caution: If you delete a graphic with embedded references, the links to the related objects will be broken. A "Symbol Not Found" message appears when you open objects whose embedded graphics have been deleted.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

To delete a graphic

1. On the **Attributes** page of the Object Editor, in the **Content** area, select the graphic to delete.
2. Click the **Delete** (-) icon. The **Confirm Delete** dialog box appears. The left pane lists the graphic selected for deletion. The right pane shows all embedded references to the selected graphic.
3. Click **Delete**.

Rearrange Objects to Improve AVEVA OMI ViewApp Navigation

By default, a running ViewApp shows objects as navigation items that appear in alphabetic order by their names. Alphabetic order may not be the most useful organization for operators. You can rearrange objects to more closely resemble their functional organization.

You can rearrange objects by dragging and dropping them in the **Model** view of the System Platform IDE. After objects are moved, the **Assets** listings of objects in the ViewApp and Layout editors update to show the new arrangement.

For example, you could move the Tank_Filters area from its alphabetic location to a new location immediately beneath the Inlet_Outlet_Pumps area. As you move the object, the indentation of a horizontal line indicates the level within the hierarchy where the object will be placed.

Rearrange objects: drag and drop

You drag and drop an object with your mouse to move an object to a different location within the ViewApp navigation model. You can rearrange objects from the **Model** view of the System Platform IDE.

To rearrange objects from the System Platform IDE

1. Open the System Platform IDE and show the **Model** view.
2. Make sure that all objects shown in the **Model** view are undeployed.
3. Select the object to be moved within the **Model** view.
4. Drag the object to the desired location within the hierarchy of objects in the **Model** view.

   A horizontal line shows the hierarchical level within the navigation model where the object will be placed.

   The horizontal line does not appear when you attempt to make an invalid move.

About rearranged objects

Rearranged objects maintain their location within a list of objects for the following Galaxy operations:

- Export and import of a Galaxy
- Back up and restore a Galaxy
- GRLoad and GRDump operations

Also, objects can be programmatically moved to another location within a list of objects using the GRAccess ToolKit.

Rearranged Objects in the Export and Import of a Galaxy

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

When rearranged objects are exported to an aaPKG file, rearranged objects are retained when the aaPKG file is imported into another Galaxy. If the imported Galaxy includes rearranged objects, but the existing Galaxy does not, the object order of the imported Galaxy takes precedence.

Rearranged Objects in Backup and Restore of a Galaxy

When rearranged objects are backed up to a CAB file, rearranged objects are retained when the CAB file is restored to another Galaxy.

Rearranged Objects in a GRLoad and GRDump

When one or more objects are selected and exported through GRDump, then GRLoad can be used to import the contents of GRDump into another Galaxy. Objects retain their rearranged locations in the GRLoad import operation in the new Galaxy.

Rearrange an Object Programmatically using GRAccess Toolkit

You can use the GRAccess object model to write programs that automate the configuration of local and remote Galaxies. You can move an object in the Model view hierarchy using the SortOrder property that programmatically moves an object after a specified predecessor object in the list of objects shown in the **Model** view.

Class

IgObject

Syntax

[C#]

```
string SortOrder { set; get; };
```

[Visual C++]

```
[propget]
HRESULT SortOrder(
[out, retval] BSTR* thePredecessorName
);
[propput]
HRESULT SortOrder(
[in] BSTR newPredecessorName
);
```

Parameters

thePredecessorName

The current predecessor of this object.

newPredecessorName

The new predecessor of this object. The object does not need to be checked out.

Remarks

SortOrder sets or returns the preceding object name in the navigation hierarchy. When the sort order is set, the tagname of the asset is displayed in the ViewApp navigation hierarchy just below the predecessor.

Assign an alias name to an object

A running ViewApp can show objects as navigation items that appear by their assigned Galaxy names in the

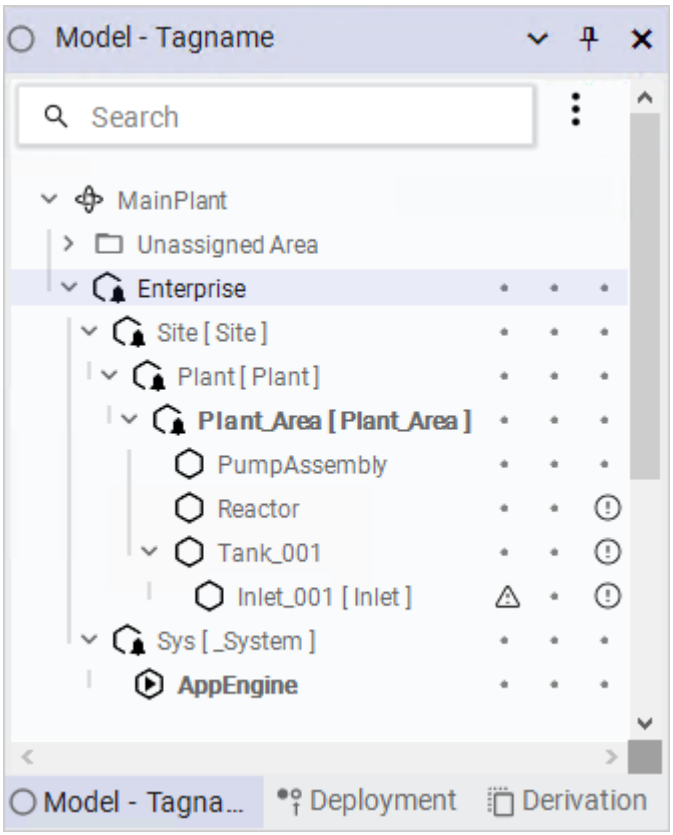
© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

navigation model. Galaxy names may not be intuitive and easily recognizable to operators. Objects can be assigned more understandable alias names.

An alias name is just a string and not associated with any object attribute. An alias does not provide another way to address items in the navigation model of a ViewApp. Scripts and animations must still reference an object based on the assigned Galaxy tagnames.

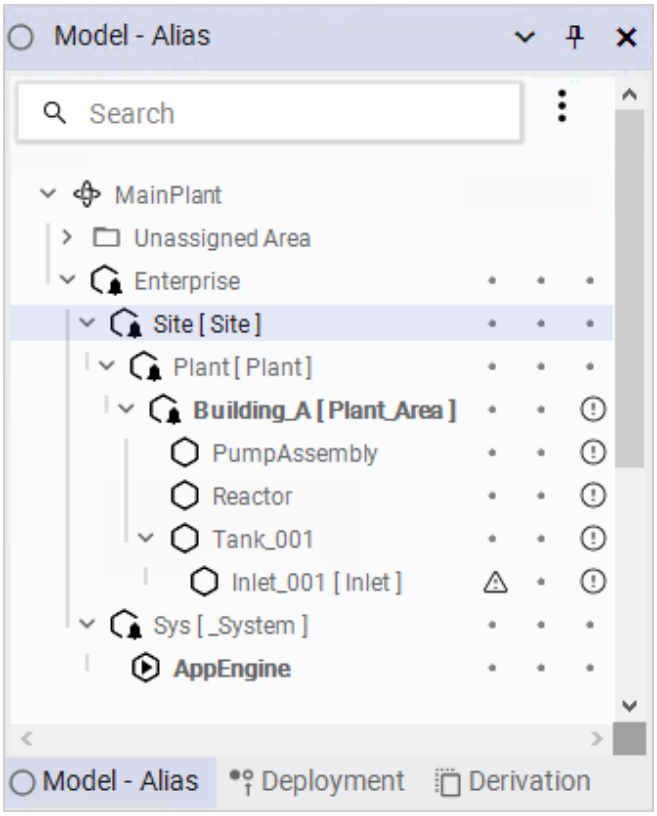
Alias names are assigned from the **Model** view of the System Platform IDE. You can toggle the **Model** view to show the objects by their tagname or alias names. In the example below, the Site area object has been assigned a more descriptive Cambridge alias name and the Plant_Area object has the alias Building_A.

Objects by Tagname



Objects by Alias Names

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.



Alias names can be shown from the ViewApp and Layout editors but cannot be modified. Also, if an alias was assigned to an object and the ViewApp or Layout editors were open, the editors must be closed and reopened to show the new alias name.

An alias name appears in the navigation model of the ViewApp during run time or in preview mode.

Alias names are associated with navigation attributes that have the word Title in their names. For example, the CurrentTitle attribute returns the alias name of the current navigation item in a running ViewApp. You can write scripts using alias names to change the current focus of a ViewApp. For more information about navigation attributes that reference alias names, see More Information About Navigation Attributes in the AVEVA OMI help.

Assign alias name to an object in Model view

You can assign an alias name to a derived instance of an object that appears in the **Model** view of the System Platform IDE. The System Platform IDE includes a **Show Alias** option that enables you to toggle the **Model** view to show the original tagname and alias names of objects.

An alias name must adhere to the following rules:

- An alias name can be up to 128 characters in length.
- An alias name can consist of alphanumeric characters and special keyboard characters (!,@,#, $,%,^,&,*,(,),-,_,=,+).
- An alias name can start with any alphanumeric or special character.
- An alias name can contain spaces.
- Multiple objects can be assigned the same alias name in a Galaxy.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Note: You cannot assign an alias name to an object if it is deployed, or if it is checked out for editing.

To assign an alias name to an object

1. Open the **Model** view of the System Platform IDE to show the list of instances.
2. Select the instance to assign an alias name to
3. On the **Home** ribbon, in the **Edit** area, select **Alias name**.

   The **Rename Alias** dialog box appears with a field to assign a new Alias name.
4. Enter an alias name that meets the format rules and select **Save**.

   The object should appear in the **Model** view with its assigned alias name.
5. If you want to verify that the alias name was assigned correctly, switch the **Model** view to show alias names: on the **View** ribbon, select **Alias**. This toggles the **Model** view between tag names and alias names.

About alias names during operations

Objects retain their aliases when objects are exported, backed up, and loaded as part of a Galaxy operation. Also, object aliases can be programmatically set and returned using the GRAccess ToolKit.

Alias retention in galaxy operations

Alias names assigned to Galaxy objects are retained for the following Galaxy operations:

- Export and import of a Galaxy
- Back up and restore a Galaxy
- Migrate a Galaxy
- Exporting instances to and importing them from a CSV file

  Alias names in the export and import of a galaxy

When objects assigned alias names are exported to an aaPKG file, object Alias names are retained when the aaPKG file is imported into another Galaxy with the following exception:

AliasName is a reserved keyword in System Platform 2020 R2 and newer versions. If an import package contains previous versions of objects with name “AliasName”, the import process skips the attribute and logs a warning message in the OCMC log. Attributes with name AliasName are not imported.

Alias names in backup and restore of a galaxy

When objects assigned alias names are backed up to a CAB file, object alias names are retained when the CAB file is restored to another Galaxy.

Migrate a galaxy

When objects assigned alias names are migrated, object Alias names are retained when migrated to another Galaxy with the following exception:

AliasName is a reserved keyword in System Platform 2020 R2 and newer versions. If a migration package contains previous versions of objects with name “AliasName”, the migration process skips the attribute and logs a warning message in the OCMC log. Attributes with name AliasName are not migrated.

Alias names when exporting to and importing from a csv file

When one or more objects are selected and exported to a CSV (Comma-Separated Value) file, you can import the contents of the CSV file into another Galaxy. Objects retain their alias names when imported to the new Galaxy.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Assign an alias name programmatically using the GRAccess Toolkit

You can use the GRAccess object model to write programs that automate the configuration of local and remote Galaxies. You can rename object aliases using the Alias property that programmatically gets or sets the alias name of an object.

Class

IgObject

Syntax

[C#]

```
string AliasName { set; get; };
```

[Visual C++]

```
[propget]
HRESULT Alias(
[out, retval] BSTR* theAlias
);
[propput]
HRESULT Alias(
[in] BSTR newAlias
);
```

Parameters

theAlias

The current alias of this object.

newAlias

The new alias of this object, or blank if the object does not have an alias. The object does not need to be checked out. The alias rename operation is immediate.

Configure historization

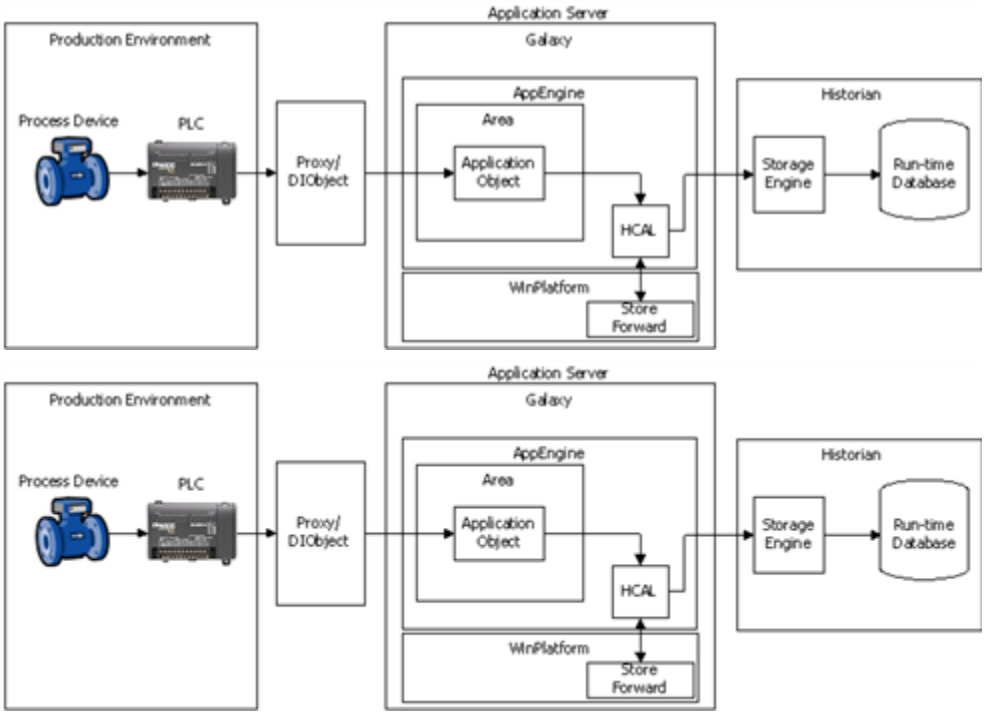
You can configure application objects to store process data in the AVEVA Historian. Historical data from Application Server can be retrieved and viewed using standard Historian database utilities. Also, you can use historical data to produce reports shown from your client applications. InTouch includes a set of ActiveX controls designed to show historical data from the Historian in trends or graphs that can be embedded in client application windows.

Application Server historization components

To save your process data to a historical database, you must install the Historian Server. An Historian Server database can be installed on any computer outside the Galaxy, but on the same network. In a production environment, the Historian should be installed on a dedicated computer without other System Platform products running on it.

The following figure shows the major System Platform components to save process data from a production device to the Historian.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.



A single Historian installation can receive historical data from a single Galaxy. A push model is used to send and save new historical updates to Historian. Each system object Engine (Platform, AppEngine, and ViewEngine) includes a historian feature that sends all history updates for all hosted objects to the historian. The historian feature receives the history updates from objects on the same engine only. All AppEngine and ViewEngine objects include an attribute to specify the node name of the computer hosting Historian.

Note: A WebViewEngine does not send any data to the Historian.

The figure shows a single Historian. This may be a common configuration, but other Application Server configurations support multiple Historian databases for a Galaxy. However, each Engine object only sends its historical data to one Historian.

There is a one-to-one relationship between a historical object attribute and a tag in Historian.

Send historical data between Application Server and the historian

Application Server communicates with the Historian through an interface called the Historian Client Access Layer (HCAL).

HCAL can establish and maintain a connection to one or more historians either synchronously or asynchronously. If a disconnection from the historian occurs, HCAL attempts to restore the connection.

If HCAL cannot communicate with the historian, all data currently being processed can be stored locally on the computer running HCAL. This hard drive location is called the store-and-forward path. Historical data is stored until the threshold capacity of the path is reached or communication to the historian is restored. In the event that all store forward disk capacity is used to store historical data, no more data is stored. An error message is logged. Remote store-and-forward paths are not supported.

HCAL can go into store-and-forward mode even if it has never been connected to the historian. After the tags are configured successfully on the historian, current data will start to be sent, along with the locally-stored data.

You can specify how disconnects between Application Server and the Historian should be reflected in the data until the disconnect period can be backfilled with store-and-forward data.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

- If you enable the **Reconnect as soon as possible & do not mark disconnects** option for an AppEngine, client-side trends during the disconnect interval are filled in with the last-received value before the disconnect, and trends do not show a gap.
- If you disable the option, a gap appears in the client-side trends during the disconnect interval. NULL values are injected during the disconnect interval to create the gap.

For more information about HCAL, see the Historian documentation. For more information about the **Reconnect as soon as possible & do not mark disconnects** AppEngine option, see AppEngine reconnect configuration for objects linked to the telemetry server.

Save object attribute data to the historian

Some attribute values can be saved as historical data when values change. Data quality and time are also associated with attribute values. When available, Application Server uses the attribute's value, the timestamp when the value changed, and the calculated quality of the data to create a Value, Time, Quality (VTQ) packet sent to the Historian. If there is no timestamp associated with the attribute value, the current scan time from the AppEngine hosting the object is used instead.

Save process values (PV) as historical data

For attribute data to be stored in the historian, a host AppEngine must be configured to send history data to a Historian node. For each object, you can configure attributes of the following data types to be saved to the Historian.

- Float (numerical).
- Double (numerical) maps to a Historian float data type. If the value of the double exceeds the range of a float, its value is clamped to the maximum value for the float and the quality is set to Uncertain.
- Integer (numerical)
- Boolean (non-numerical)
- String – Unicode (non-numerical). Limited to 512 characters. Characters that exceed the 512 maximum length are truncated. If truncation occurs, the quality value of the packet is set to Uncertain.
- CustomEnum (non-numerical) maps to a Historian integer.
- ElapsedTime (numerical) maps to a Historian float and is converted to seconds.
- Arrays or parts of arrays are not supported.
- Enum type attributes are saved as Historian integer ordinal values.
- IEEE NaN values for float and double data types are converted to null values prior to being sent to the historian. NaN values are associated with a Bad OPC data quality.
- All numerical attributes sent to the Historian are in the engineering units specified for the attribute. The Historian does not scale numerical values.

Save data quality (VTQ) as historical data

The Historian supports the OPC Quality definition. A 16-bit value for OPC data quality is sent to the Historian in the VTQ packet. Within the 16-bits, the low-order byte is the standard OPC portion. The high order byte is

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

reserved and is currently unused. The Good, Bad, Initializing (which is a form of Bad) and Uncertain quality states are specified in the low-order byte.

Any change in the mapped Historian Quality Detail saves a new record to the Historian, regardless of whether the attribute value has changed. If an attribute value remains constant over a period of time with varying changes in quality, multiple records are stored in the Historian. The OPC quality stored in the Historian is the actual quality of the attribute in Application Server without modification.

The Historian can create and insert additional (non-System Platform generated) VTQ records that modify quality in the database. These additional VTQ records provide additional information about network outages or other events to client applications that use historical data from the Historian.

Additional quality data saved to the historian

The Historian also has two additional data quality fields (quality and quality detail) that are non-OPC compliant. Both fields are stored with each record in addition to the OPC quality. The Historian maps the value of these fields in a manner consistent with its quality definition for those fields. The values of those fields are driven by the OPC quality sent by Application Server. For example, an OPC Good quality results in the Historian quality detail of Good.

Historian Quality is a 1-byte flag that summarizes the quality of the associated data value in the packet sent by Application Server. Historian Quality is an enumerated type with the following values:

Hex	Decimal	Name	Description
0x00	0	Good	Good value.
0x01	1	Bad	Value was marked as invalid.
0x10	16	Doubtful	Value is uncertain. Only fabricated at retrieval time.
0x85	133	Initial Value	(Good) Initial value for a delta request.

OPC Qualities from Application Server result in storage of a Historian Quality as follows:

OPC quality (substatus form)	Historian quality decimal (1-byte)
Good (any)	0 - Good
Bad (any)	1 - Bad
Bad (Initializing form)	1 - Bad
Uncertain (any)	0 - Good

Historian QualityDetail contains the detail of data quality as a 32-bit value. QualityDetail is derived from OPC quality. Each source is allocated a byte position within the QualityDetail 32-bit word, which is formatted as:

0xXXRRDDDD

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Only the low-order two bytes (DDDD) are mapped to OPC quality for data sent by Application Server and saved to the historian:

OPC quality (any substatus form)	Results in any one of these Historian quality details
Good (any)	192 - Good value 150 - Initial value (good) 151 - Initial store/forward value (good) 252 - First value received (good) 44 - First good value received by storage after the connection to HCAL has been restored. 248 - First value in the second stream during a store-and-forward.
Bad (any)	24 (DAServer disconnect)
Bad (Initializing form)	24 (DAServer disconnect)
Uncertain (any)	192

Save the data timestamp as historical data

Application Server includes a timestamp in the VTQ packet sent to the historian for each attribute value/quality update that is saved as a historical record. Application Server propagates the timestamp associated with an attribute value. The timestamp is propagated throughout all Application Server components as UTC in FILETIME format.

If an object attribute has a Value DeadBand specified, a quality change or value change from previous scan cycle greater than Value DeadBand causes the attribute's Value, Time and Quality (VTQ) to be saved as historical data in the Historian. When a Value DeadBand is not specified, any change to the attribute’s value, timestamp, or quality cause the attribute to be saved as historical data.

When an attribute is configured to periodically save a historical record, the attribute’s current time is used as the timestamp. If the attribute does not support a timestamp, the hosting AppEngine's current scan time is used as timestamp included in the packet.

If slow updates for real-time data values are received from an I/O source and an intermittent network disconnect occurs, the timestamp of the first data value of a historized attribute uses the AppEngine’s timestamp instead of the source’s timestamp. In the following example, in row number 4, the value "30" is sent again to the historian, but the timestamp is the AppEngine timestamp and not the actual timestamp of "2008-11-24 18:50:41.061" when the value of 30 was generated from the I/O source.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

DateTime	TagName	Value	Quality	QualityDetail	OPCQuality
2008-11-24 18:48:30.081	Realtime_ Client.ival	20	133	44	192
2008-11-24 18:50:41.061	Realtime_ Client.ival	30	0	192	192
2008-11-24 18:51:38.666	Realtime_ Client.ival	NULL	1	24	0
2008-11-24 18:52:06.670	Realtime_ Client.ival	30	0	192	192 –> Row number 4
2008-11-24 18:53:22.988	Realtime_ Client.ival	40	0	192	192

If slow updates for late data values are received from an I/O source and an intermittent network disconnect occurs, the timestamp of the first data value of a historized attribute is modified to maintain the time sequence. In the following example, in row number 4, the value "30" is sent again to the historian, but the timestamp is modified by adding 5 milliseconds to the previous timestamp of the NULL data value. Note that a QualityDetail of 704 is stored for the data value in row number 4.

DateTime	TagName	Value	Quality	QualityDetail	OPCQuality
2008-11-24 18:48:30.081	Latedata_ Client.ival	20	133	44	192
2008-11-24 18:50:53.624	Latedata_ Client.ival	30	0	192	192
2008-11-24 18:50:53.625	Latedata_ Client.ival	NULL	1	24	0
2008-11-24 18:50:563.630	Latedata_ Client.ival	30	0	704	192 –> Row number 4
2008-11-24 18:53:26.863	Latedata_ Client.ival	40	0	192	192

Save alarms and events as historical data

Alarms and events detected by the Application Server at run time are saved as historical data to the Historian for the host engine. Alarms are a type of event that can be historized. In addition, alarm-related events such as Acknowledge are also saved as historical alarm data.

You can save alarm counts by severity level as historical data. For more information about mapping alarm severities to priority ranges and saving alarm counts aggregated by severity level to the Historian, see Configure priority ranges for alarm historization, mapping, and shelving .

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Deploy and undeploy attributes

When an object is deployed with historical attributes, the history feature causes dynamic reconfiguration of the Historian. Each history feature causes a new tag to be created and configured automatically in the Historian at deployment time. The Historian storage system that will be used is determined by the configuration of the host engine object.

This dynamic configuration causes appropriate row/column configuration in the Historian schema.

If the connection to the Historian is down at deploy time, the attempt to dynamically reconfigure Historian is achieved when the connection is restored. In other words, automatic retry is built in. However, any data generated by the object during the time between deployment and the recovery is lost and cannot be recovered by store forward.

When an object configured for history is redeployed, changes to an object's attributes makes the historian reconfigure storage. For example, if the engineering units string of an object changes from "Deg F" to "Deg C" when the object is redeployed, the Historian configuration database shows the change.

When an object configured for history is undeployed, all history remains in the Historian. The history data can be viewed in the future even if the object is no longer deployed.

Save historical data during run time

While an application is running, data is saved to the historian as follows:

- If no previous historical attribute value exists in the historian, the first value is always saved to the historian.
- Numerical attributes (double, float or integer): If the value for the attribute changes and that change is more than the value deadband, or the value's quality changes (for example, from Good to Bad), the data is saved to the historian.
- If the attribute type is qualitative (enumeration, string, or Boolean) and the attribute's value or quality changes.
- If the current time exceeds the last Forced Store period occurred for the attribute by the time period that was set as the Force Storage Period. If no last Forced Store occurred since starting the object, a new store occurs immediately.

  Note: If enabled, the Value Deadband mechanism resets itself based on this new stored value.
- All new attributes value, timestamp, and quality are sent to the Historian for storage.
- The Historian stores the historical records to the database.

Advanced communication management

To ensure that attribute data saved to the historian is always current and continuous, Application Server registers a reference to the attribute. By registering a reference to the attribute, it prevents Message Exchange from suspending historical updates during the period when the client application has minimized the window containing the object. Historical data continues to be saved to the Historian even during the period when the object is in an Advanced Communication Management Suspended state.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Store-and-forward mode

If the Historian shuts down or the network connection to it is lost while an application is running, historical data continues to be stored locally on the computer hosting the WinPlatform object. You define the folder to hold this "store-and-forward" data when you configure the historical data options for the WinPlatform.

When the Historian node recovers, data is forwarded from the local node to the Historian node at a low priority.

If an AppEngine loses connectivity to the Historian node, the Historian reports bad data quality to clients. When you undeploy an object with attributes configured for history, the Historian stores the final data points with Bad quality.

Buffered data behavior

The buffered data feature enables efficient accumulation and propagation of VTQ (Value, Time, and Quality) data updates, without folding and data loss, to data consumers such as objects, alarms, the Historian, and scripts from field devices that support buffering.

Buffered data is defined as data captured and stored locally on a remote device for later transfer to a supervisory system for processing, analysis, and long-term storage. The Buffer property is input-only.

When an object is associated with an attribute that supports buffered data (HasBuffer property is true), the object monitors and processes the Buffer property of the attribute. Each VTQ entry in the buffer that meets value deadband criteria is pushed to the Historian.

Attributes that are buffer-enabled are registered to Historian in the same manner as non-buffered attributes are registered. The Historian does not differentiate a buffered from a non-buffered registered attribute. An attribute is considered registered after a Historian key is returned from HCAL to the object.

All the buffered VTQs that are received by the parent attribute while Historian registration is pending are ignored and discarded.

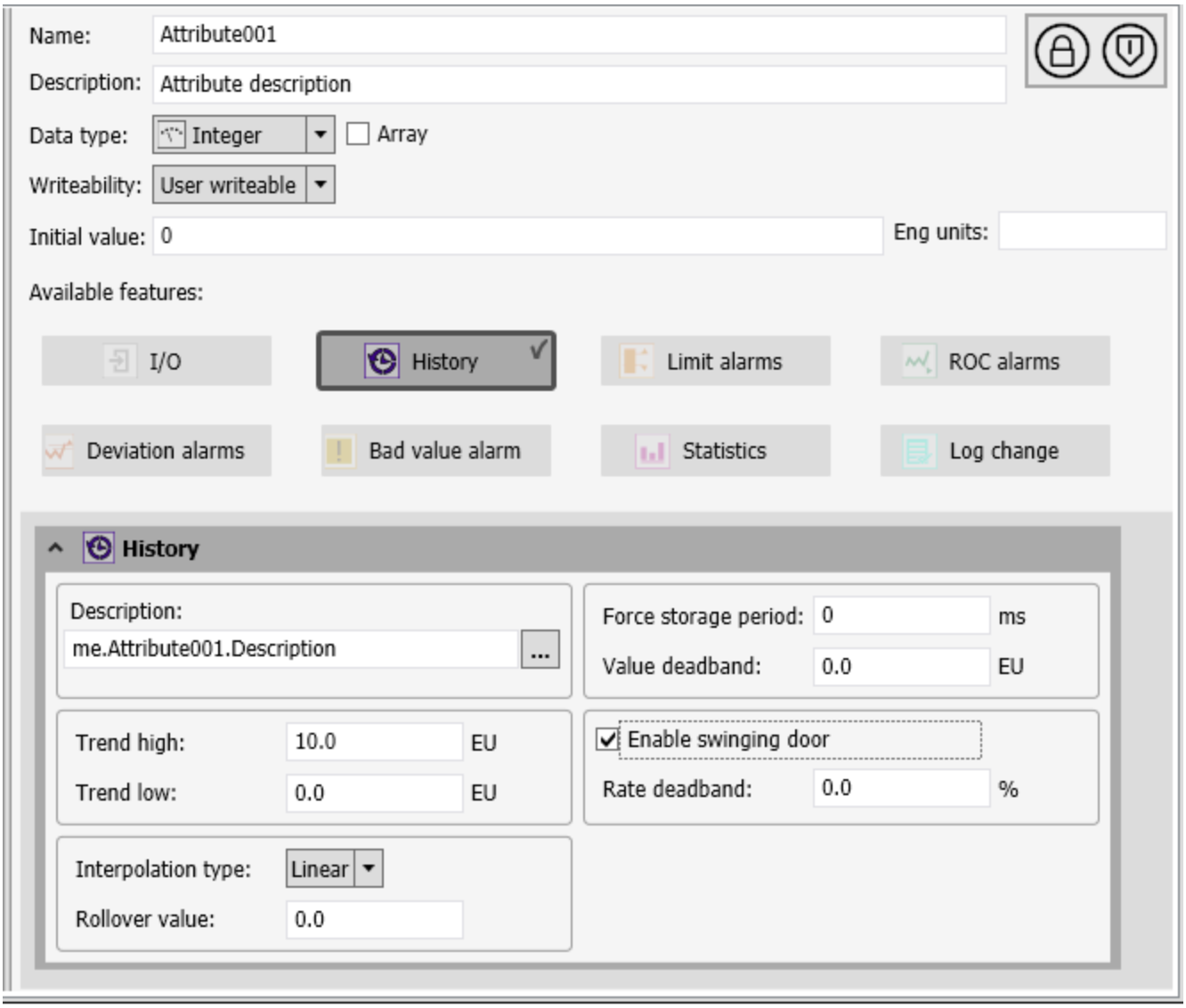
For more information about buffered data, see Enable buffered data. For further reading on historizing events and attributes that are buffering-enabled, see About buffered data and history.

Configure common historical attributes

Each application or system object that you select from the Template folder include a set of common attributes you configure to save data to the Historian. These are configured after activating the **History** feature in the **Attributes** page.

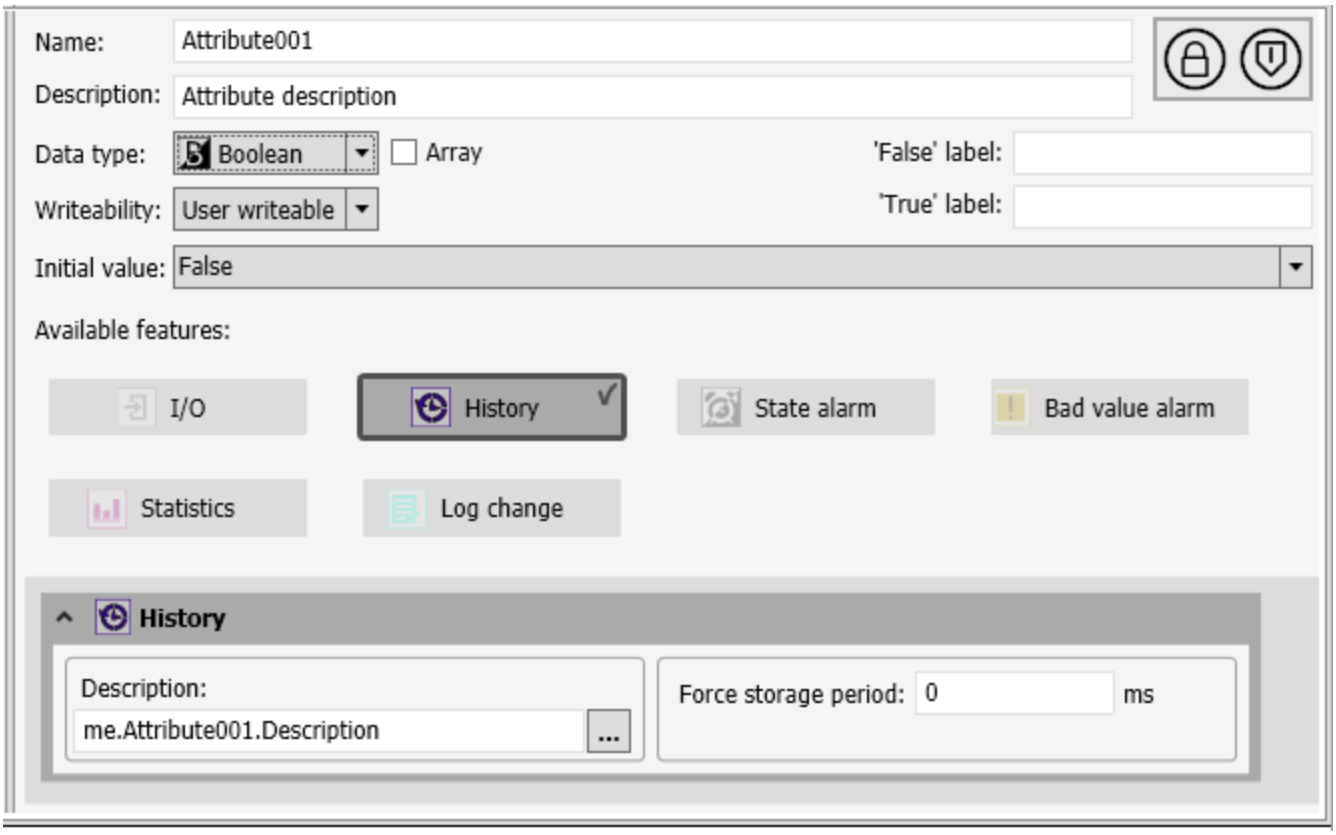
The historical attributes you configure are based on the data type of the object. For example, the following figure shows the common historical attributes for numerical attributes (integer, float, or double).

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.



If you are configuring a boolean or string attribute, you see a smaller set of historical attributes. Similarly, a discrete field attribute will also have this same, smaller set of historical attributes.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.



This section describes the common historical attributes that you can configure for your system and application objects. Refer to this section as you complete the procedures to configure your objects.

- **Force Storage Period**

  Interval in milliseconds in which an attribute value is saved to the Historian, regardless of whether the value exceeds its value deadband setting or not. An attribute value is saved to the Historian at every Force Storage interval. The default value of zero (0) disables the Force Storage period.

- **Value Deadband**

  Threshold value in engineering units, which is the absolute difference between the current and most recent saved historical values. The current value must exceed the absolute deadband to be saved as historical data.

  The Value Deadband filters out small, momentary value changes from being saved to the Historian. A new value that is within the range of the deadband is not saved to the Historian. The default value of 0.0 disables the Value Deadband and any change to a value is saved as historical data.

- **Trend Hi**

  Initial maximum trend value in engineering units for clients. The default is 100.

- **Trend Lo**

  Initial minimum trend value in engineering units for clients. The default is 0.0.

- **Enable Swinging Door**

  A flag that indicates whether the swinging door rate deadband is enabled or disabled. The default is disabled.

  Enable Swinging Door is disabled if the historical attribute data type is Boolean or a string.

- **Interpolation Type**

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

List of methods used by the Historian to interpolate analog historical data. The interpolation type determines which analog value is selected during a Historian data retrieval cycle. Interpolation types include System Default, Stairstep, or Linear. The default interpolation type is System Default.

- Stairstep

  No interpolation occurs. The last known value is returned with the given cycle time. If no valid value can be found, a NULL is returned to the Historian.

- Linear

  The Historian calculates a new value at the given cycle time by interpolating between the last known value prior to the cycle time and the first value after the cycle time.

- System Default

  The Historian system-wide interpolation setting. The system-wide setting must be either stairstep or linear interpolated.

- **Rollover Value**

  Positive integer value that represents a tag's reset limit when the Historian operates in counter retrieval mode. In counter retrieval mode the Historian uses a tag's rollover value to calculate and return the delta change between consecutive retrieval cycles. The default value is 0.0.

  The Rollover value applies only to numeric attributes. The Rollover value is disabled if the historical attribute data type is Boolean or a string.

- **Rate Deadband**

  Percentage rate of change deadband based on the change in the slope of incoming data values to the Historian. For example, specifying a swinging door rate deadband of 10 percent means that data is saved to the Historian if the percentage change in slope of consecutive data values exceeds 10 percent.

  The default is 0.0, which indicates a swinging door rate deadband is not applied. Any percentage greater than 0.0 can be assigned to the rate deadband.

Configure system objects to store historical data

The following table shows the historical attributes for Application Server system objects.

History attributes	**Application Server system objects**				
	WP	**AE**	**Area**	**VE**	**ITVA**
Description	**..+**		**..+**	**..+**	
Enable compression	**..+**	**..+**		**..+**	
Reconnect as soon as possible & do not mark disconnects		**..+**			
Enable Storage	**..+**	**..+**		**..+**	

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

History attributes	Application Server system objects				
to Historian					
Enable Swinging Door	..+	..+	..+	..+	
Enable Tag Hierarchy	..+	..+		..+	
Engineering units	..+	..+	..+	..+	
Force Storage Period	..+	..+	..+	..+	
Historian	..+	..+		..+	
History store forward directory	..+	..+		..+	
Interpolation Type	..+	..+	..+	..+	
Pre-processing buffer size	..+	..+		..+	
Rate Deadband	..+	..+	..+	..+	
Rollover Value	..+	..+	..+	..+	
Store forward threshold	..+	..+		..+	
Store forward minimum duration	..+	..+		..+	
TCP Port	..+	..+		..+	
Throttling network bandwidth	..+	..+		..+	
Trend High	..+	..+	..+	..+	
Trend Low	..+	..+	..+	..+	
Value Deadband	..+	..+	..+	..+	

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

History attributes	Application Server system objects				
Wait to send incomplete packets	..+	..+		..+	

WP: WinPlatform object

AE: AppEngine object

Area: Area object

VE: ViewEngine object

ITVA: InTouchViewApp object

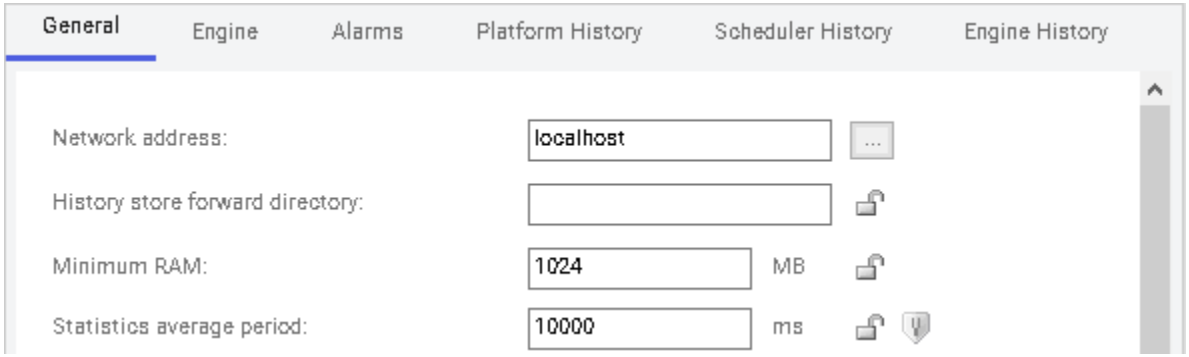
Note: WebViewEngine objects do not store historical data.

Configure the WinPlatform object to store historical data

A WinPlatform object contains attributes to configure how historical store-and-forward data is cached locally and then pushed to the Historian. Also, you can select the WinPlatform object's platform, scheduler, and engine data to be saved to the Historian. Finally, you can select the WinPlatform's attribute values to be saved to the Historian.

To configure a WinPlatform Object to store historical data

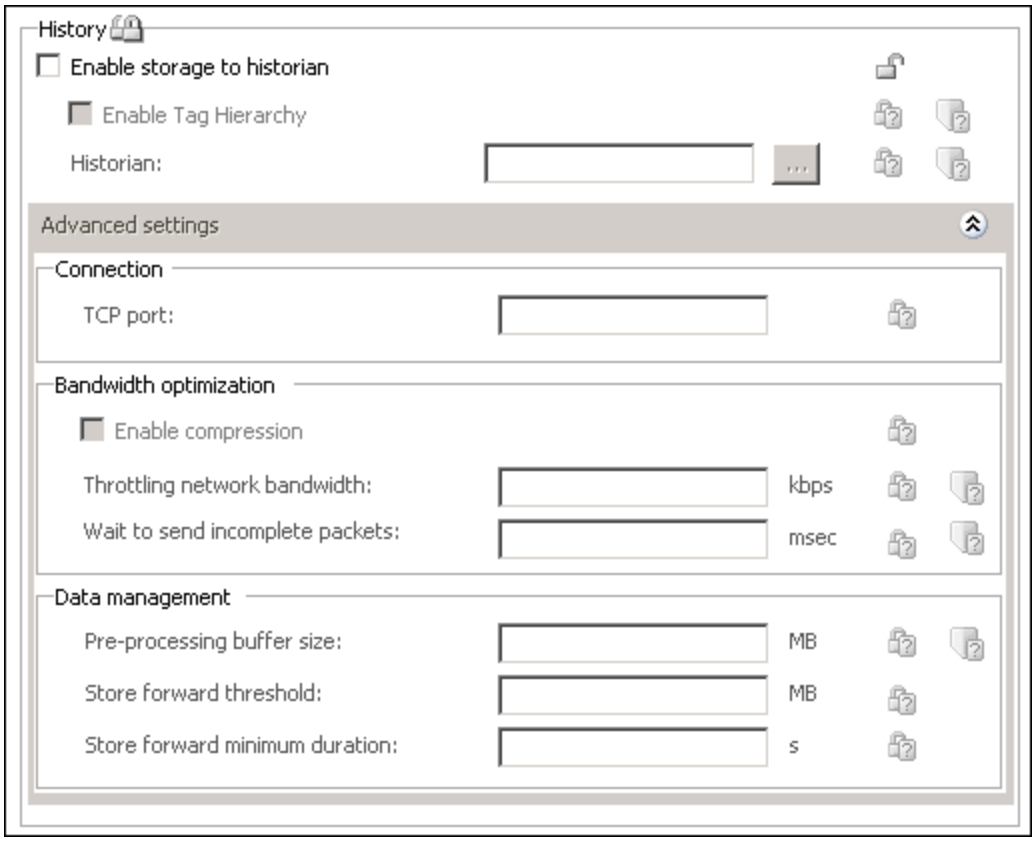
1. Double-click on a WinPlatform derived template or instance to open the Object Editor.
2. Click the **General** tab to show history attributes.



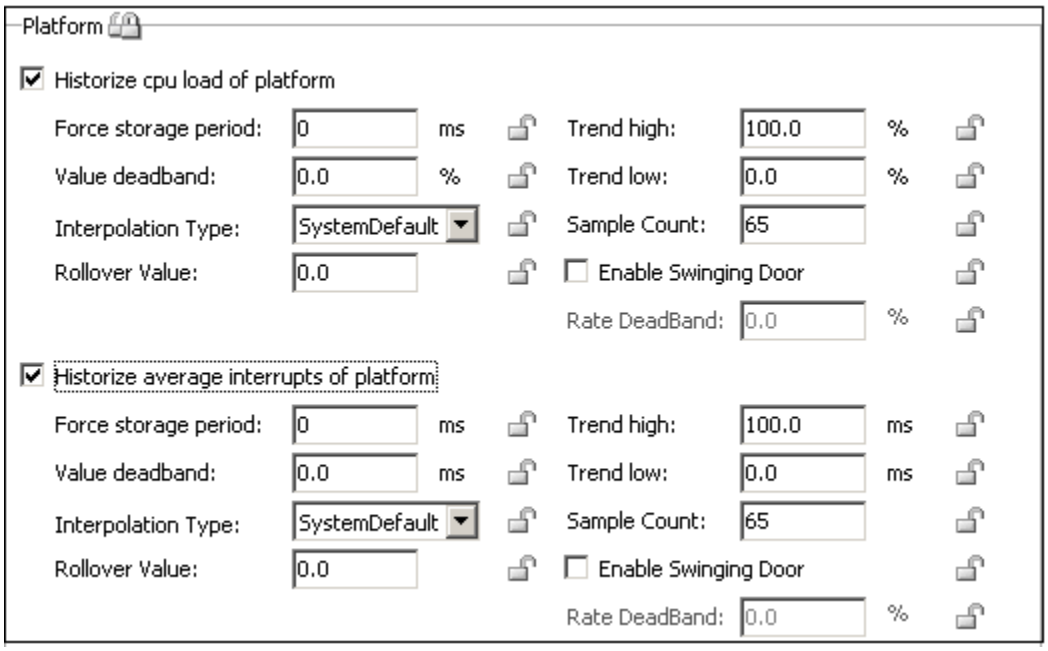
3. In the **Network address** box, type or select the node name of the computer assigned to the WinPlatform object.
4. In the **History store forward directory** box, enter the path to the folder to save store-and-forward historical data. The directory must exist on the computer specified in the **Network address** box. The Network User Account for the WinPlatform needs to have full file system permissions for this directory.

   Note: Once you have set the store forward directory and deployed the WinPlatform, you should not change the directory. Doing so might leave behind unprocessed data that has not yet been sent to the Historian.
5. Click the **Engine** tab to show history attributes.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.



6. In the **History** area of the **Engine** page, configure the attributes. For more information on each attribute, see the WinPlatform object Help.
7. Click the **Platform History** tab to show the attributes to save system data from the computer hosting the WinPlatform object to the historian.



8. Select the check box next to each attribute whose data you want saved in the historian. For more information about these historical attributes, see Configure common historical attributes.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

9. Click the **Scheduler History** tab to show the attributes to save data to the historian.
10. Select the check box next to each scheduler attribute whose data you want saved in the historian.
11. Click the **Engine History** tab to show the attributes to save data from AppEngines hosted by the WinPlatform object.
12. Select the check box next to each engine attribute whose data you want saved in the historian.
13. Save your changes and check in the WinPlatform object.
14. Deploy the object to its target computer in an on-scan state.

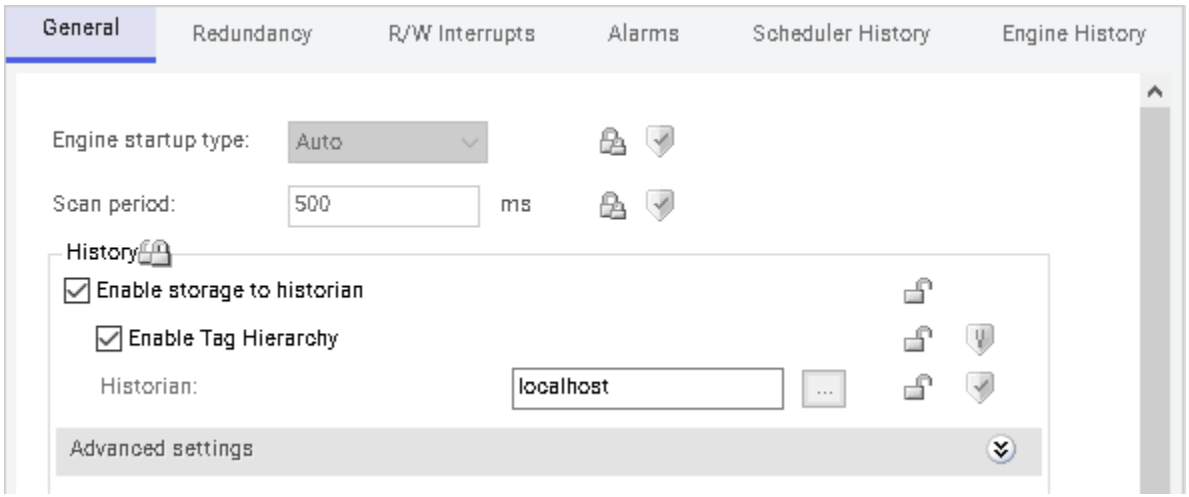
Configure an AppEngine object to store historical data

If an AppEngine is deployed before Historian is started, history data can be stored locally by HCAL until the objects successfully register with the Historian.

Note: Except for Late Data, a ViewEngine object contains the same historical attributes as an AppEngine object. A WebViewEngine object does not store any historical data.

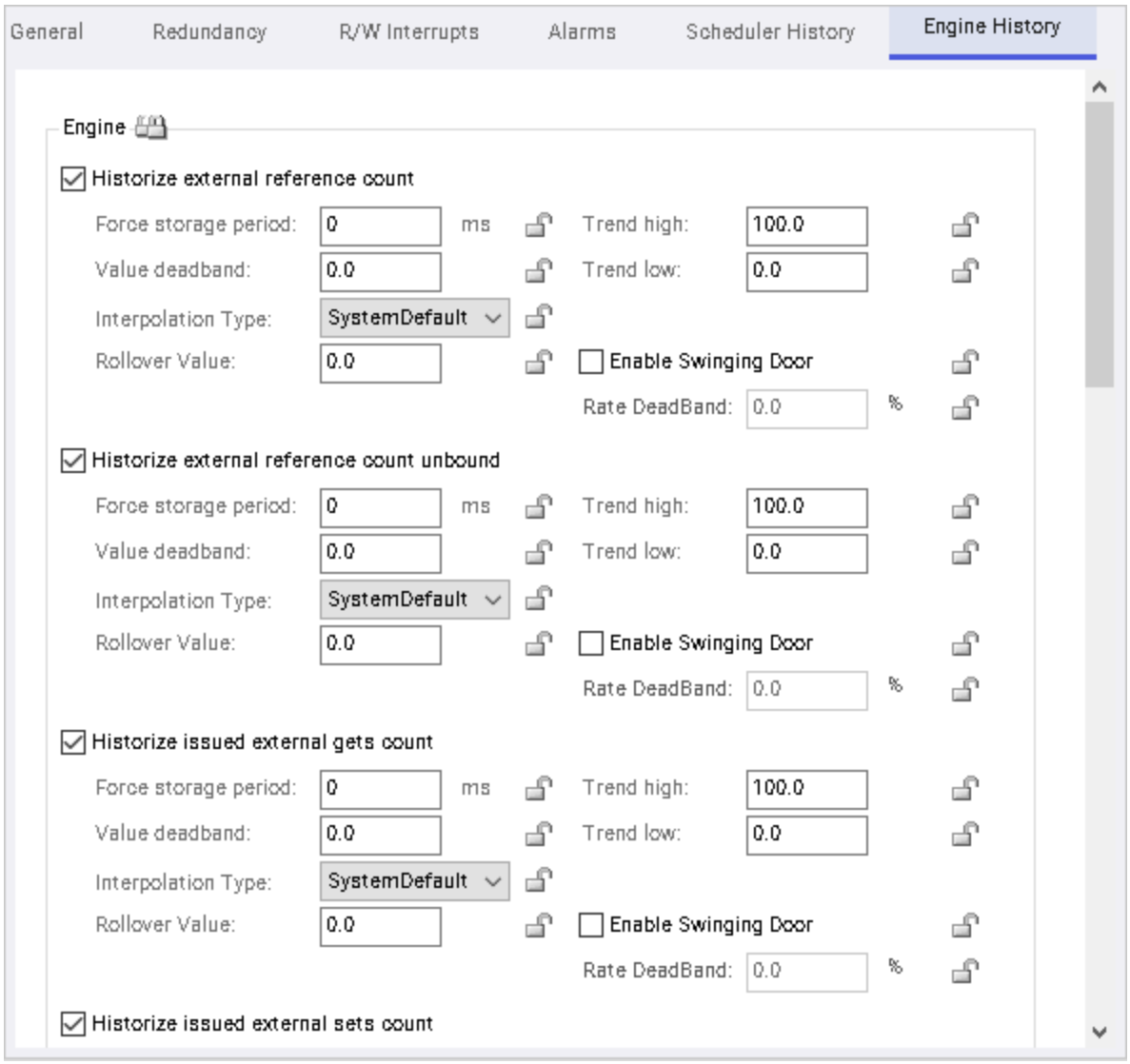
To configure an AppEngine object to store historical data

1. Double-click the AppEngine instance to open it in the Object Editor.
2. Click the **General** tab to show history attributes.



3. Configure the attributes. For more information on each attribute, see the AppEngine object Help.
4. Click the **Scheduler History** tab to show the attributes to save data to the historian.
5. Select the check box next to each scheduler attribute whose data you want saved in the historian.
6. Click the **Engine History** tab to show the attributes to save data from for this AppEngine.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.



7. Select the check box next to each engine attribute whose data you want saved in the historian.
8. Save your changes to the AppEngine object.

Configure an Area object to save alarm counts as historical data

An Area object represents a plant area to group objects for modelling and report alarms. You can configure a set of attributes to save the counts of an area's alarm states to the historian. An Area object contains a set of attributes to save the counts of the following alarm states to the historian:

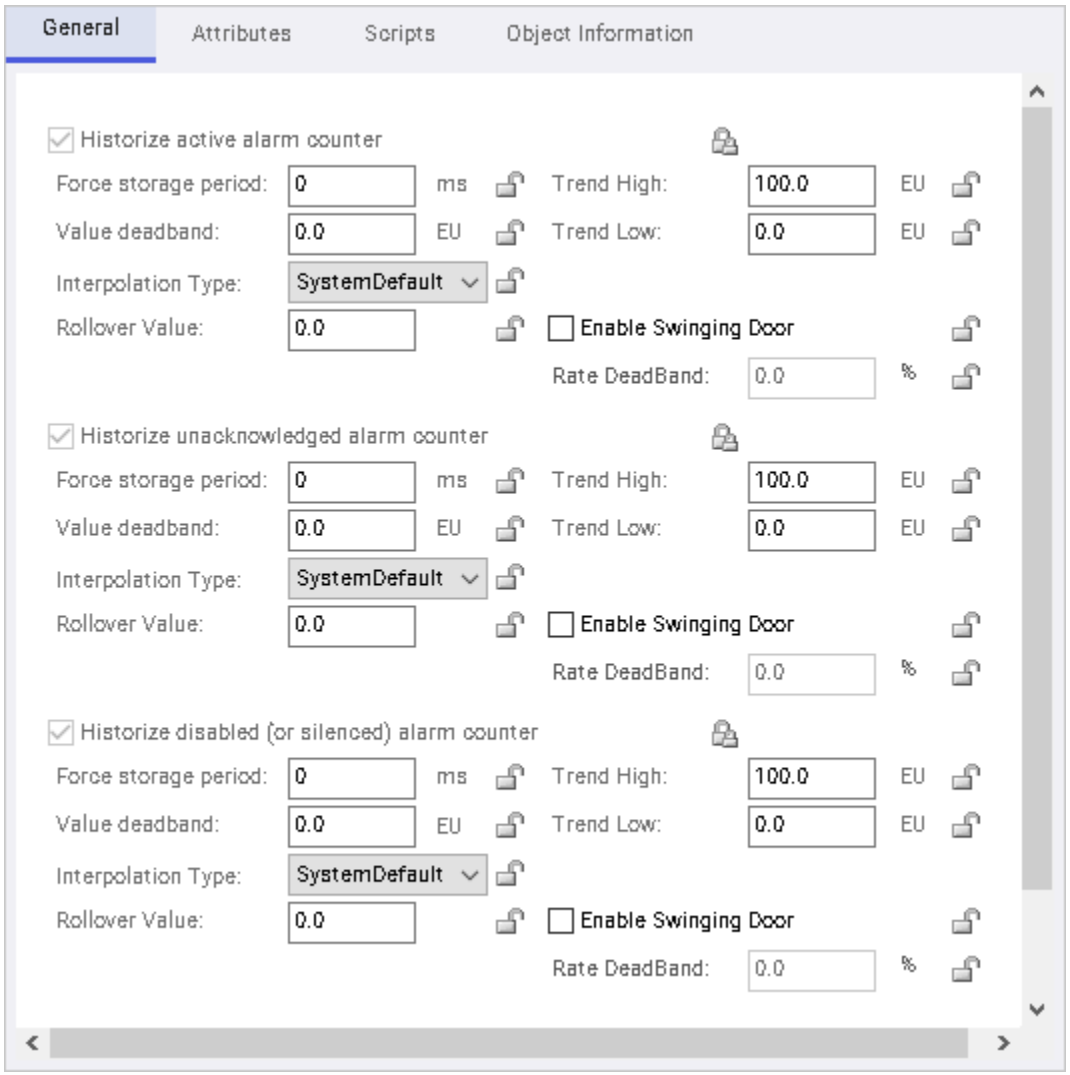
- Active alarms
- Unacknowledged alarms
- Disabled or silenced alarms

To configure an Area object to store historical data

1. Double-click the Area instance to open it within the Object Editor.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

2. Click the **General** tab to show attributes for area alarm counts that can be saved as historical data.



3. Select the check box next to each alarm state counter that you want to save to the historian.
4. Set the attributes for each alarm counter that you select. For more information about assigning values to historical attributes, see Configure common historical attributes.
5. Save your changes and close the Object Editor.

Configure an application object to save historical data

The following table shows the historical attributes for application objects. These history attributes are described in Configure common historical attributes.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

History attributes	Application objects										
	AD	**Bool**	**DD**	**Dbl**	**FR**	**Flt**	**Int**	**SQ**	**Str**	**SW**	**UDO**
Enable Swinging Door	..+	..+		..+	..+	..+	..+				
Force Storage Period	..+	..+	..+	..+	..+	..+	..+		..+	..+	
Interpolation Type	..+	..+		..+	..+	..+	..+				
Rate Deadband	..+	..+		..+	..+	..+	..+				
Rollover Value	..+	..+		..+	..+	..+	..+				
Trend High	..+	..+		..+	..+	..+	..+				
Trend Low	..+	..+		..+	..+	..+	..+				
Value Deadband	..+	..+		..+	..+	..+	..+				

AD: AnalogDevice

Bool: Boolean

DD: DiscreteDevice

Dbl: Double

FR: FieldReference

Flt: Float

Int: Integer

SQ: Sequencer

Str: String

SW: Switch

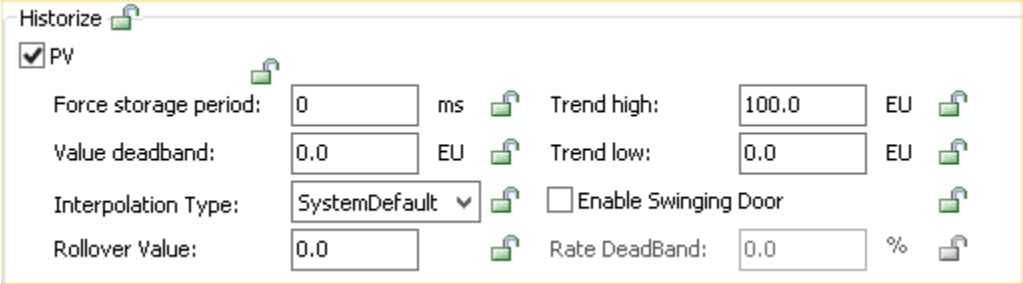
UDO: UserDefined

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Note: The historical attributes available from the UserDefined object are based on the data type associated with the selected attribute.

To configure an application object to store history

1. Open the application object in the Object Editor.
   - If you selected a UserDefined object, click the **Attributes** tab.
   - If you selected an AnalogDevice object, click the **History** tab. Then, click the General tab to show the history attributes for the object's PV value.



2. Enable historization.
   - For UserDefined objects, create an attribute and then activate the **History** feature. Assign values to the history parameters. For more information about assigning values to the history feature, see Configure common historical attributes.
   - For AnalogDevice objects, select the **Historize** feature to enable the common history attributes shown in the **History** area of the page.

     The History page includes check boxes to save **PV**, **SP**, **PV mode**, **Control mode**, and **Control Track Flag** data to the historian.

     Assign values to the PV history attributes that appear in the **History** area of the page. For more information about assigning values to the common history attributes, see Configure common historical attributes.
3. Save your changes and close the Object Editor.

Alarms and events

You can create AVEVA OMI and InTouch HMI applications that generate alarms and events to provide the status of a running application.

Application objects include built-in event and alarming reporting capabilities. You must configure alarms for each object in the IDE to use the event and alarm functions.

Event definition

Events represent normal system, application, or user occurrences that produce status messages. A typical event occurs when an operator logs on to an application at the beginning of a work shift. Application Server can detect events, store them as historical data, and report them to client applications.

An event indicates a significant occurrence that is detected, reported, and saved as historical data. Events can be detected by any Application Server component including automation objects and the OCMC.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Events provide a means for any software component to log information about a system, application, or operator action. Application Server components use an event API to send event messages to Alarm/Event distributors and the Historian. Also, events are sent to client applications like AVEVA OMI to be shown in real-time trends or reports.

Types of events

Application Server creates several types of runtime event messages, which are saved as historical data.

- System events

  Event messages are logged when a Platform, Engine, Area, DI Network or DI Device start or stop. Any system action that affects a large number of objects is logged as an event.

  A System event message contains the following information:

  - Event type, which is System.
  - Tagname, which is the name of the object generating the event.
  - Tag description, which is a short description of the object generating the event.
  - Area, which is the name of the area that contains the object generating the event.
  - Event description, which describes the system action and can be either Started or Stopped.
  - Timestamp, which is the current system time.
- Security-audit (operator change) events

  Event messages are logged to the Alarms and Events subsystem when an operator logs on to or logs off from an application, and when a logon attempt fails (for example, due to an incorrect password). Also, security-audit events are logged when an operator changes actions by means of User sets.

  A Security-audit event message contains the following information:

  - Event type, which is operator change.
  - Timestamp, which is the date and time when the operator change event occurred. The timestamp of the event is the current AppEngine scan time.
  - Tagname, which is the name of object generating the event.
  - Prim.attr, which is the reference string of the attribute being changed.
  - Tag description, which is a short description of the object. For Secured and Verified writes, this will contain the following:
    - Type of write (Secured or Verified)
    - Comment from the user, if any
    - Reason description, if any
  - FieldAttribute description, if the attribute is a FieldAttribute and has a description; otherwise, this is the Object description.
  - Area, which is the name of the area that contains the object generating the event.
  - UserEngine, which is the name of the view engine or other user engine requesting the operator change.
  - Operator1, which is the full name of the primary operator requesting a change. The full name is an attribute of the UserProfile.
  - Operator2, which is the full name of the secondary operator validating the change, if any.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

- Old value, which is the previous value of an attribute.
- New Value, which is the new value of an attribute.

- Application (or process) related events

  Application event messages are generated by application objects in response to process actions. For example, application event messages are created when a process pump starts or stops.

  An Application event contains the following information:

  - Event type, which is data change.
  - Timestamp, which is the date and time when the application event occurred. The timestamp assigned to the event is the timestamp of the attribute associated with the event, if available. Otherwise, the event timestamp is the current AppEngine scan time.
  - Tagname, which is the name of the object generating the event.
  - Prim.attr, which is the reference string of the attribute being changed.
  - Tag description, which is a short description of the object.
  - Area, which is the name of the area that contains the object generating the event.
  - Old value, which is the previous value of an attribute.
  - New Value, which is the new value of an attribute.

Alarm definition

Alarms warn about process conditions that can potentially cause problems. Typically, you set up an alarm to become active when a process value exceeds a defined limit. For example, you can set an alarm for a pump that warns when no fluid pressure is detected.

An alarm is an abnormal condition that requires immediate attention. An operator usually acknowledges an alarm. The System Platform framework handles the real-time reporting of alarms and provides special clients for viewing them.

Application and system objects can detect and generate an alarm. To detect an alarm, a system or application object sets a Boolean Attribute flag to indicate whether the alarm condition of the object is currently true or false.

To report an alarm, the object must contain an alarm feature. The alarm feature makes a reference to the object's Boolean flag to determine whether the alarm condition is true. It then combines this information with the current alarm mode to determine whether to report a this as an active or inactive alarm state. An alarm feature is dedicated to reporting a single alarm condition's state. The alarm feature sends alarm notification messages to System Platform alarm and event distributors.

Every alarm notification includes a set of fields containing data that describes the alarm. Some alarm notification data is saved as historical data. The following list describes all fields sent with an alarm notification.

- TagName: The name of the object generating the alarm. Saved as historical data.
- Name: The name of the alarm. Saved as historical data.
- InAlarm: A Boolean value that indicates whether the object's alarm state is currently active or inactive. Saved as historical data.
- Quality: The current quality of the data upon which the alarm is based. Saved as historical data.
- OnTimeStamp: The time that the attribute value transitioned to an alarm state. The value timestamp of the attribute is used, if available. If not, the timestamp is the AppEngine scan time. Saved as historical data.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

- OffTimeStamp: Time that the alarmed attribute value returns to normal. The value timestamp of the attribute is used, if available. If not, the OffTimeStamp is the AppEngine scan time. If an active alarm is disabled, it forces a return to normal and the timestamp is the current time. Saved as historical data.
- Category: An integer between 1 and 14 that identifies the type of alarm and source of the alarm. These values are associated with Internationalized category labels.

  Alarm category labels can be localized to other languages. Application Server uses the default Galaxy language to retrieve these strings and send them to InTouch. The alarm category labels appear in InTouch and InTouch history as the default Galaxy language strings. Saved as historical data.
- Priority: An integer value from 1 to 999 indicating the severity of the alarm. An alarm priority of 1 is most urgent and 999 least urgent.
- TargetValueReference: An optional field that makes a reference to the target value of the alarm. Not saved as historical data.
- ActualValueReference: An optional field that makes a reference to the actual attribute value for the alarm condition. Not saved as historical data.
- TargetValue Snapshot: An optional field containing the target value of the attribute at the time when the alarm became active. Saved as historical data.
- ActualValueSnapshot: An optional field containing the actual value of the attribute at the time when the alarm became active. Saved as historical data.
- EngUnitsReference: The reference to the engineering units string for the condition. Saved as historical data.
- AcknowledgedFlag: Indicates whether the alarm is acknowledged or not. If this flag is FALSE, the alarm is still unacknowledged. Saved as historical event data.
- AcknowledgeTime: Indicates the time when the alarm was acknowledged if the AcknowledgedFlag is TRUE. Saved as historical data at the time of acknowledgement.
- AcknowledgeUserId: The string containing the name of the user who acknowledged the alarm. Saved as historical data at the time of acknowledgement.
- AlarmMode: Indicates if the alarm mode is Enabled, Silenced or Disabled. Saved as historical data at the time when the alarm mode changes.
- Message text describing the alarm, which can be statically or dynamically constructed. The message typically contains the alarm description, the exceeded limit value, and possibly the variable value. For an alarm feature, this message is retrieved by means of a reference to a string/international string attribute in the object. The reference is setup at ObjectDesigner time for alarms. If none is specified, then the common feature short description attribute is utilized. This field is also provided in the alarm feature section and can be dynamically generated and scripted. Saved as historical data.
- Area: The name of the area that contains the object generating the alarm. Saved as historical data.

Alarm states

An alarms will have a particular status (state) that can be reported by an object for which alarming has been enabled, depending on both user actions and the reported attribute value. An attribute value that is not within normal preset parameters is reported as an alarm. Note that an alarm state (the status of the alarm) is not the same as a **state alarm**. See State alarms for additional information.

The Latched alarm state only exists if it has been enabled for the Galaxy. See Alarm latching for additional information.

The alarm states are:

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

- Normal (no alarm)
- Active (Unack/Alarm, usually displayed as UNACK)
- Acknowledged (Ack/Alarm, usually displayed as ACK)
- Unacknowledged/Return to Normal (Unack/RTN, usually displayed as UNACK_RTN)
- Acknowledged and returned to normal (Ack/RTN, usually displayed as ACK_RTN)
- LATCHED (optional)

  A latched alarm is one that has both returned to normal and has been acknowledged by an operator (LATCHED state). It will continue to be shown by the alarm client, for example, the InTouch Embedded Alarm Client or the OMI Alarm App, until an operator issues a Dismiss command.

Latched alarm attributes

Alarm latching uses three attributes on the alarm extension primitive, and one attribute on the engine primitive. These attributes are always present, and become active when latching is enabled.

The alarm extension attributes used for latching are:

- Latched: indicates whether the alarm is in the latched state
- DismissMsg: User-writeable attribute to dismiss a latched alarm
- TimeAlarmDismissed: the time the alarm is dismissed

The engine attribute used for latching is:

- AlarmLatchEnabled: indicates whether the galaxy-wide setting is turned on for alarm latching

For information about using latching at run time, see Dismiss a latched alarms .

Types of alarms

Application Server supports the following types of alarms:

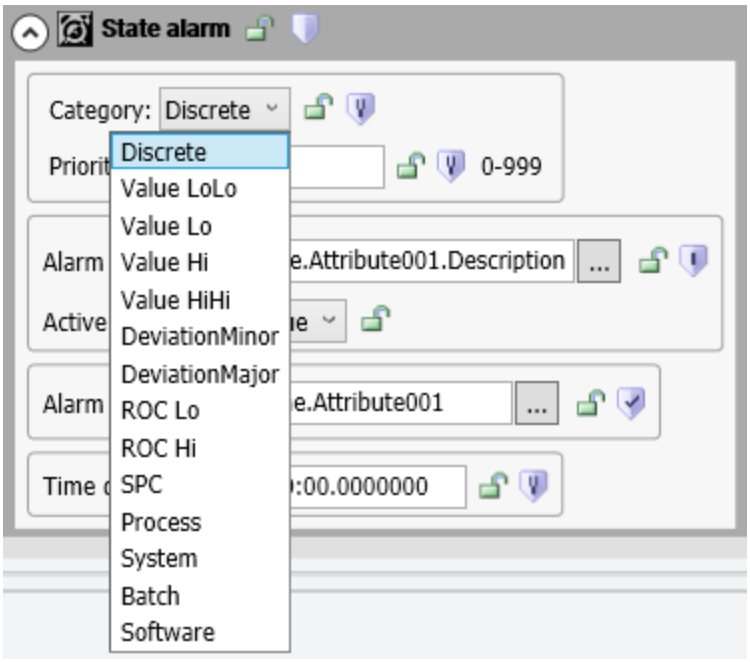
- State alarms, which are also known as Boolean alarms
- Limit alarms
- Target deviation alarms
- Rate of change alarms

The types of alarm that you can configure are based on the data type of the attribute's value.

State alarms

A state alarm set on an attribute of Boolean data type corresponds to a discrete tag with two possible states. When you create a state alarm, you configure whether the active alarm state corresponds to the TRUE or FALSE state of the attribute.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.



You can set an alarm category. Valid categories are:

Category	Description
Discrete	A discrete value event or alarm, such as a change of state.
Value LoLo	A continuous value is significantly below the acceptable range.
Value Lo	A continuous value is below or is approaching the low acceptable range.
Value Hi	A continuous value is above or approaching above the high acceptable range.
Value HiHi	A continuous value is significantly above the acceptable range.
DeviationMinor	A value has a minor deviation (plus or minus) from the target or setpoint.
DeviationMajor	A value has a major deviation (plus or minus) from the target or setpoint.
ROC Lo	The rate of change for a value is too slow.
ROC Hi	The rate of change for a value is too fast.
SPC	A value deviates from the SPC target/range.
Process	An alarm or event associated with the physical

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

	process/plant has occurred.
System	An alarm or event associated with the automation system has occurred.
Batch	An alarm or event associated with a batch process has occurred.
Software	An alarm or event associated with a software operation or logic (such as divide by zero in script) has occurred.

You can set an alarm message and Priority for a state alarm. The time deadband sets the length of time that an attribute value must continuously remain in an alarm or unalarmed state. The time deadband filters out rapid, transitory value spikes.

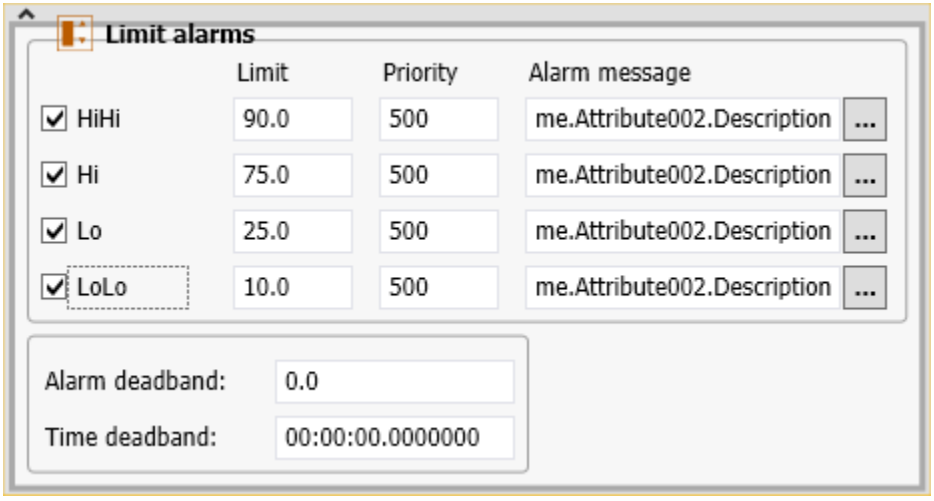
The timestamp when a state alarm becomes active or inactive is the most current timestamp of the corresponding input value. If there is no timestamp associated with the alarmed value, the AppEngine timestamp is used instead.

Limit alarms

A limit alarm compares the current value to one or more predetermined alarm limits within the attribute's full range of values. If the value exceeds a limit, an alarm occurs.

Run-time graphic showing limit alarm values for LoLo, Lo, Hi, and HiHi alarms

You can individually select and configure values and priorities for the LoLo, Lo, Hi, and HiHi alarm limits. You can set individual messages for each alarm limit.



You can also configure alarm and time deadbands for limit alarms. The alarm deadband is an absolute value, in engineering units, that the attribute value must exceed the configured HiHi, Hi, Lo, or LoLo limit before the alarm is triggered, or that the attribute value must revert to before the limit alarm resets to its inactive state.

For example:

- The HiHi alarm limit is set to 90.
- The alarm deadband is set to 5.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

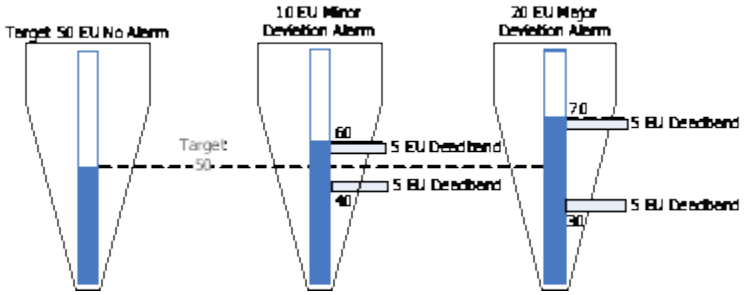
- The attribute value must exceed 95 (limit plus deadband value) to trigger the HiHi alarm.
- The attribute value must fall below 85 (limit minus deadband value) to reset the HiHi alarm to an inactive state.

The time deadband sets the length of time that an attribute value must continuously remain in an alarm or unalarmed condition. The process variable must remain above or below the indicated limit for at least the indicated deadband time before the application object updates the status of the alarm CONDITION Boolean. Then, standard Alarm feature logic determines whether to take that updated alarm condition and report changes to the alarm state or not.

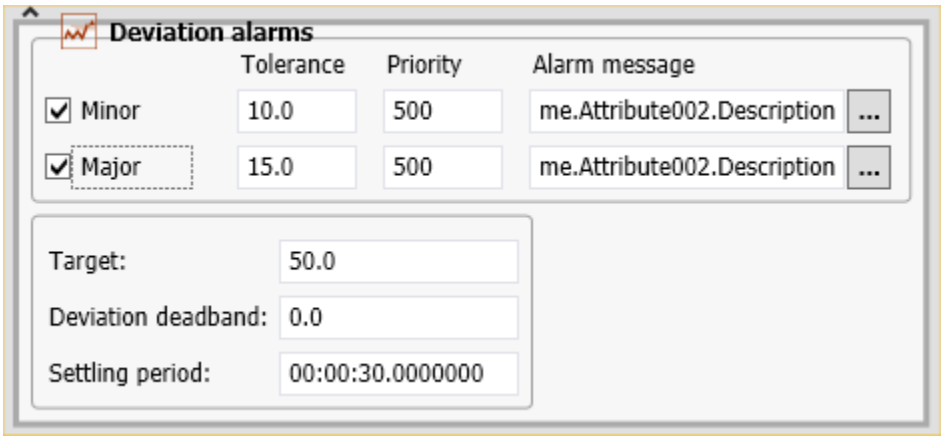
The timestamp when a limit alarm becomes active or inactive is the most current timestamp of the corresponding input value. If there is no timestamp associated with the alarmed value, the AppEngine timestamp is used instead.

Target deviation alarms

A target deviation alarm compares the current attribute value to a target Engineering Units value. Then, the absolute value of the difference is compared to one or more alarm deviation limits expressed in EngineeringUnits.



You can individually select and configure values, priorities, and alarm messages for **minor** and **major** deviation limits. A minor deviation can be thought of as analogous to a Hi limit alarm, and a major deviation would be analogous to a HiHi limit alarm.



Settling period is the time allowed for the attribute value to reach an expected target value after a device starts. No alarm can occur during the settling period.

Deadband range sets a threshold that an attribute value must change from a deviation limit to reset the alarm to the inactive state. Deviation deadband is expressed in Engineering Units.

The timestamp when a deviation alarm becomes active or inactive is the most current timestamp of the

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

corresponding input value. If there is no timestamp associated with the alarmed value, the AppEngine timestamp is used instead.

Rate of change alarms

A rate of change (ROC) alarm identifies when an attribute value is changing too quickly over time. For example, you can set a rate of change alarm for a tank level that indicates when the pump inlet pressure is too high.

Rate of change is the calculated slope. The slope is calculated by taking the absolute difference between the current and previous attribute values, divided by a specified interval. When the slope (positive or negative) exceeds a specified value, a rate of change alarm occurs. For example, if the volume of a tank increases from 17 to 45 liters over a 5 minute interval, the calculated slope is 5.6 liters per minute. If you set your rate of change alarm limit to 5.0 liters per minute, a rate of change alarm condition exists.

45 - 17 = 28

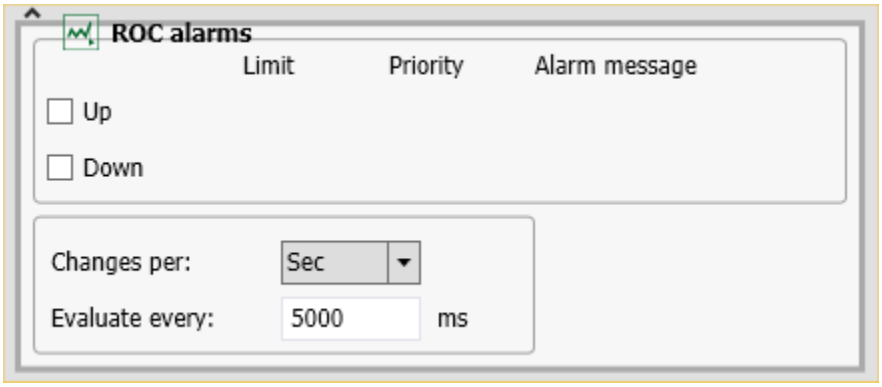
28 / 5 = 5.6

5.6 (calculated rate of change) > 5 (rate of change setting), thus ROC alarm condition exists

The following figure illustrates changes in level over a one minute interval.

Run time display showing changes in value over a one minute interval

Alarm limits are expressed in the Engineering Units of the attribute's value over an interval, which can be per second, minute, hour, or day.



You can select and configure the value and priority for the upward and downward ROC limits. You can set individual messages for ROC alarms that exceed the upward or downward limits.

The timestamp when a rate of change alarm becomes active or inactive is the most current timestamp of the corresponding input value. If there is no timestamp associated with the alarmed value, the AppEngine timestamp is used instead.

Statistical alarms

A statistical alarm is one in which a statistic is calculated, based upon an attribute. If the statistic exceeds some pre-set limit, the object flags the alarm condition as TRUE.

Set alarm state with object attributes

The Application Server alarm enable/disable mechanism includes four attributes to set an object alarm mode and report alarm status:

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

- AlarmModeCmd attribute
- AlarmInhibit attribute
- AlarmMode attribute
- AlarmModeEnum attribute

AlarmModeCmd attribute

AlarmModeCmd is a writable attribute that sets the current commanded alarm mode for the object. You set the AlarmModeCmd to enabled, silenced, or disabled with a script, user input, or from an attribute configured to read from an input source.

AlarmInhibit attribute

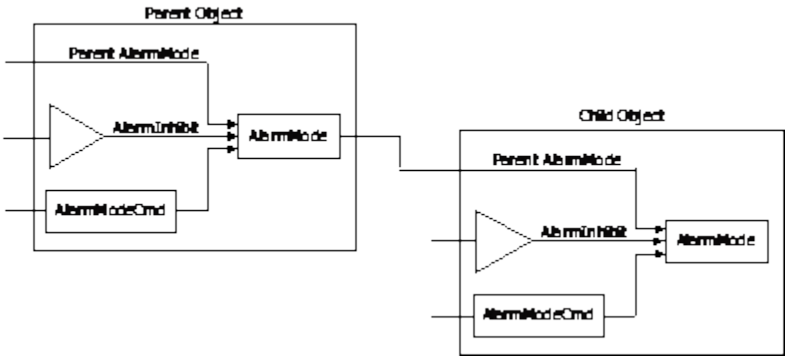
The AlarmInhibit attribute disables one or more alarms when set to TRUE. The value of the AlarmInhibit attribute is typically set by a script, manually by the user, or from an attribute configured with an input feature. If the AlarmInhibit attribute is set TRUE, all alarms of the object and of any contained objects are disabled.

When the AlarmInhibit attribute is set to FALSE, alarms are not inhibited and the object AlarmMode and parent object AlarmMode determine whether alarming is enabled, silenced, or disabled.

AlarmMode attribute

The AlarmMode is a calculated attribute that identifies the object alarm mode and is based upon the current values of an object's:

- AlarmModeCmd attribute
- AlarmInhibit attribute
- Parent object AlarmMode attribute



Application Server checks the AlarmModeCmd and AlarmInhibit attributes of an object and the AlarmMode status of the parent object. Application Server then updates the object's AlarmMode attribute to reflect the most restrictive setting.

All individual alarms use the object's AlarmMode status to determine whether they are enabled, silenced, or disabled.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

AlarmModeEnum attribute

The _AlarmModeEnum attribute can be used by any object to obtain an enumeration of permissible settings for the AlarmModeCmd attribute.

Alarms and buffered data

An object with alarms enabled detects the alarm condition and sets an associated alarm condition attribute to true. If the object detects the alarm condition based on an attribute with buffered data (HasBuffer property set to true), it enables buffer support for the condition attribute and sets the condition attribute's Buffer property to a VTQ buffer of true/false values representing alarm conditions.

Alarm detection

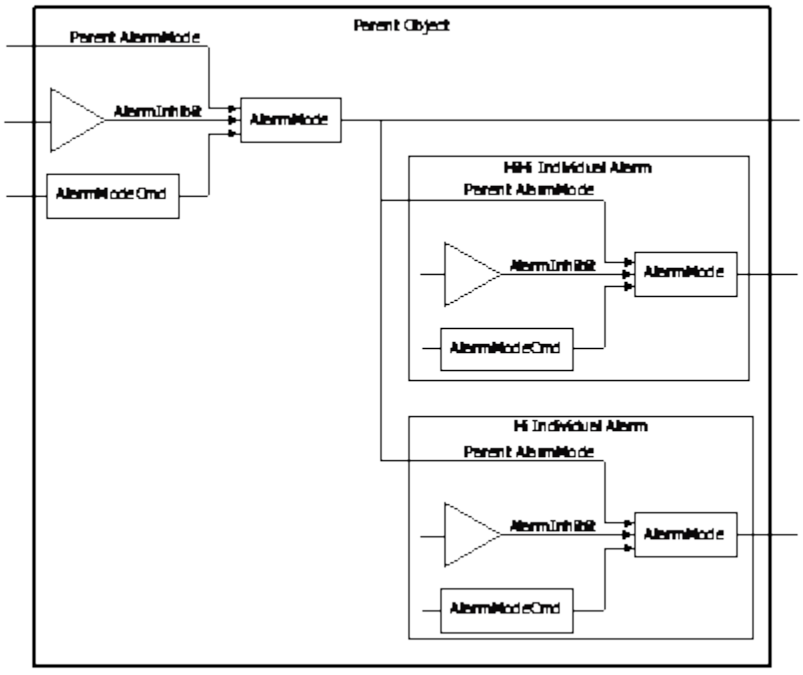
The Buffered Data feature will pass buffered alarm VTQs to the alarm subsystem in the same manner as with buffer data not enabled. When enabled, the Buffered Data feature can generate multiple alarm messages in the same scan.

For more information on alarm detection, see Alarms and buffered data.

Set the alarm state for individual alarms

You can set individual alarms within an object for each type of alarm. For example, you can set alarms for each of the limits of a level alarm.

The following figure shows an object's individual alarms for the HiHi and Hi alarm limits.



The calculated AlarmMode attribute value of an individual alarm uses the same inputs as an object alarm. The parent AlarmMode attribute is from the object itself. As with object alarms, the individual alarm mode is set to the most restrictive input state. For example, if the object's AlarmMode state is disabled and the individual

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

alarm's AlarmInhibit is FALSE, the individual alarm is disabled.

Each individual alarm is autonomous from other individual alarms in an object. The AlarmMode of an individual alarm is not propagated to other alarms. Unlike inhibit for the entire object, inhibit of an individual alarm does not affect the alarms of any contained objects. You can selectively enable, silence, or disable an individual alarm and not set other alarms to the same value within the object hierarchy.

Enable, silence, and disable alarms

Alarms can be enabled, disabled, or silenced while an application is running. An object's alarm state can be set at the Area level, at the container object level, or at the individual object. In addition, individual alarms within a single object can be enabled, silenced, or disabled.

- Enabled: All alarms for an object are reported to client applications and saved as historical data. The enabled state is less restrictive than the silenced or disabled alarm states.
- Silenced: All alarms for an object are detected and logged in the Historian alarm and event history database. Silenced alarms are not logged to the InTouch alarm database, and are not shown in alarm clients displaying current alarms or recent history alarms. The silenced alarm state is more restrictive than the enabled state, but less restrictive than the disabled state.
- Disabled: No alarms for the object are detected. The alarm is return-to-normal until the alarm is re-enabled. The disabled state is more restrictive than the silenced and enabled alarm states. A disabled alarm does not require acknowledgement.

When an alarm state changes from silenced to enabled, the following applies:

- Only the last occurrence of the alarm appears in alarm clients.
- Only the last occurrence of the alarm is logged into the alarm database.
- Only the last occurrence of the alarm is saved to history.

The following figure shows how the different alarm modes affect the different phases of an alarm. In the case of Alarm Suppressed, selected alarms can be filtered out of the InTouch AlarmViewer display by an operator, or programmatically by a script.

Alarm condtion, alarm state, and alarm notification settings affect the alarm display at runtime

To ensure that alarmed attributes always have current data, the alarm feature always registers a reference to the alarmed attribute. This guarantees that Message Exchange never suspends updates for this attribute. Even if alarms are disabled for a particular attribute, the alarmed attribute cannot enter an Advanced Communication Management Suspended state.

The object hierarchy and alarm states determine the final alarm condition of an object.

- An Area object's alarm state determines the alarm state for all alarms of objects that belong to the Area.

  Diagram showing that the alarm state of an area object determines the alarm state for all objects that belong to the area
- The most restrictive setting within an object hierarchy determines the object's alarm state.

  The most restrictive setting within an object hierarchy determines the object's alarm state
- When an individual object alarm is silenced or disabled, it applies only to that alarm and not to other alarms belonging to the object.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

A silenced or disabled alarm applies only to that alarm and not to other alarms belonging to the object

The alarms on any contained object are not affected. The disabled or silenced state of an individual alarm does not propagate downward through the object hierarchy to the alarms of any contained or assigned object.

Enable alarms

To enable an object's alarms, you must ensure that the AlarmModeCmd and AlarmInhibit attributes are enabled for the object, its container, and its area. An event, including the user's name, is generated indicating the object's alarms are enabled.

When object alarms are enabled, you can enable, silence, or disable an individual alarm.

Silence alarms

When object alarms are silenced, an individual alarm that is enabled or silenced is forced to be silenced. When object alarms are silenced, an individual alarm can be disabled.

Disable alarms

When object alarms are disabled, an individual alarm that is enabled or silenced is forced to be disabled.

When object alarms are enabled and an individual alarm is enabled or silenced, the individual alarm can be inhibited. This forces the individual alarm to be disabled.

When object alarms are silenced and an individual alarm is enabled or silenced, the individual alarm can be inhibited. This forces the individual alarm to be disabled.

When object alarms are inhibited, an individual alarm that is enabled or silenced is forced to be disabled.

Shelve alarms

You can shelve alarms to temporarily hide them from displays for a fixed period. Alarms continue to be historized, even when they are shelved.

Shelving typically is used to reduce alarm "noise", or to temporarily suppress alarms that might be triggered during system modifications or repairs, allowing you to focus on correcting other more urgent alarms.

Shelving is similar to silencing an alarm, but shelved alarms differ from silenced alarms in the following ways:

- Shelved alarms have a built-in associated time-out period. Shelved alarms are automatically unshelved when the configured shelving period expires. You can also manually unshelve alarms and return them to an active, displayed state.
- Alarm shelving must be enabled at an area level, but shelving applies only to individual alarms. You cannot shelve a hierarchy of alarms for an entire area or for an entire object. You cannot propagate alarm shelving through the Model View hierarchy.
- The system enforces role-based limitations on permission to shelve alarms, alarm severity levels that can be shelved, and the total number of alarms a user can shelve.

The system tracks who shelved the alarm, from what workstation, the reason for shelving the alarm, when shelving began, and when shelving will expire. Shelved alarms aggregate in similar fashion to silenced alarms. For

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

information about alarm aggregation, see Aggregated alarm information.

A set of seven attributes provide runtime alarm shelving information and control:

AlarmShelveCmd	User writeable. Use this attribute to shelve or unshelve an alarm. Default values: Duration = 0, Reason = ""
AlarmShelved	Read-only, Boolean value. Shows True if alarm is shelved, False if alarm is unshelved. Default value: False
AlarmShelveStartTime	A read-only date/time stamp indicating when alarm shelving began, based on the engine time when the shelving request was received, Default value: Blank
AlarmShelveStopTime	A read-only date/time stamp equal to the AlarmShelveStartTime plus the duration for which the alarm is to be shelved. Default value: Blank
AlarmShelveReason	A read-only string value providing the reason for which the alarm was shelved or un-shelved by the Alarm Shelve command. Default value: Blank (See AlarmShelveCmd attribute.)
AlarmShelveUser	Read-only, the name of the user who most recently shelved or un-shelved the alarm with the Alarm Shelve command. Default value: Blank
AlarmShelveNode	Read-only, the name of the computer node from which the user most recently shelved or un-shelved the alarm with the Alarm Shelve command. If the node is hosted in a Terminal Server client session, the node and the TSE ID are both identified. Default value: Blank

Enable alarm shelving

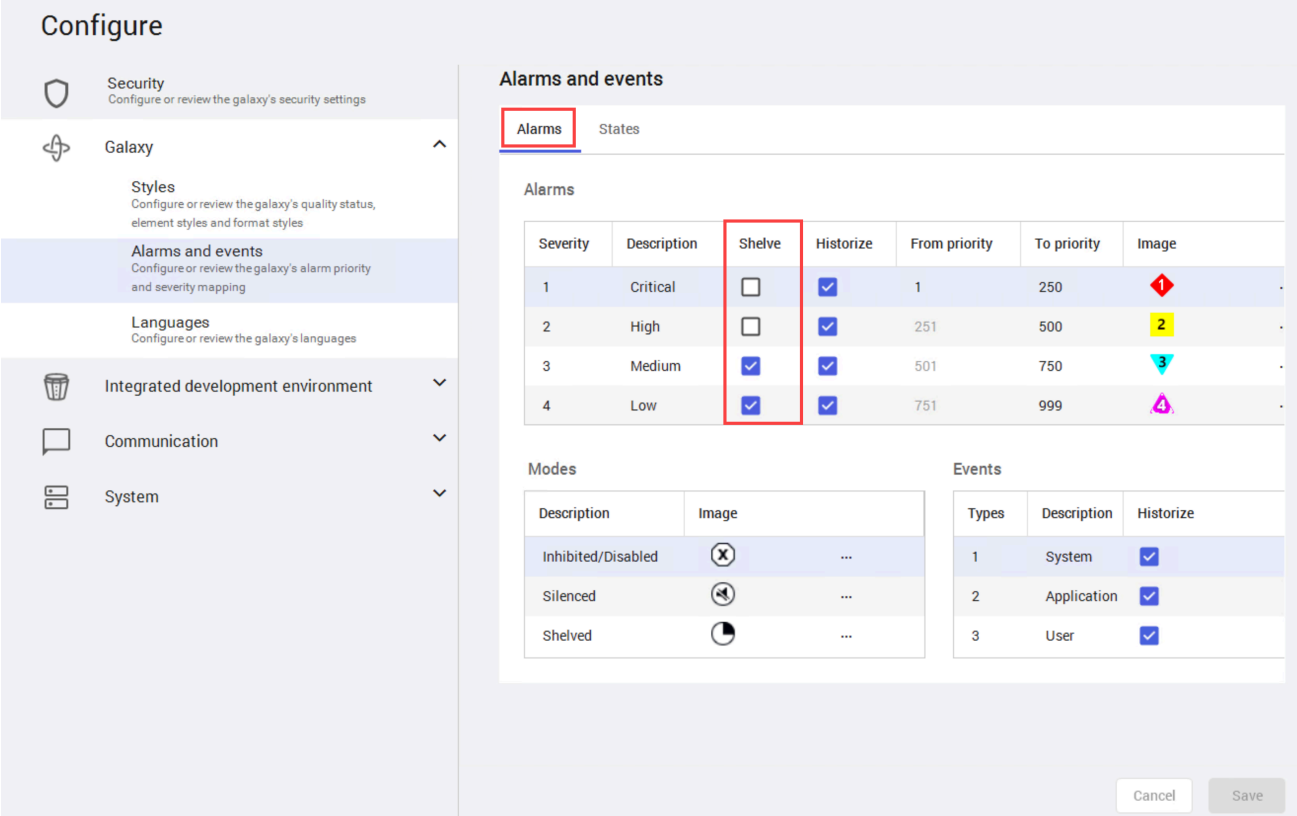
Alarm shelving is configured from the **Galaxy Configuration** menu, and is enabled on the Area object.

To enable alarm shelving

1. Enable at least one security role to configure alarm shelving. Associate the relevant user with that role, if not already associated, before proceeding to step 2.
2. Select the **Galaxy** menu item, then select **Configure**.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

3. Select **Galaxy**, then select **Alarms**. The **Alarms** page appears.



4. Select the severity level in the **Shelve** column. By default, severity levels 3 (medium) and 4 (low) are enabled.
5. In the IDE, open the relevant Area object editor. Click the **Shelve Alarms** checkbox to enable shelving for that area. Shelving is enabled by default.

Shelve alarms at runtime

Use a runtime client to shelve and unshelve alarms. From Application Server, you can use Object Viewer to monitor and control shelved alarm attributes. Shelved alarms can also be monitored and controlled from an embedded Alarm Control and at least one animation to write to the AlarmShelveCmd attribute. For information about using the Alarm Control to shelve alarms, see *Shelve and Unshelve Alarms at Run Time* in the AVEVA OMI help.

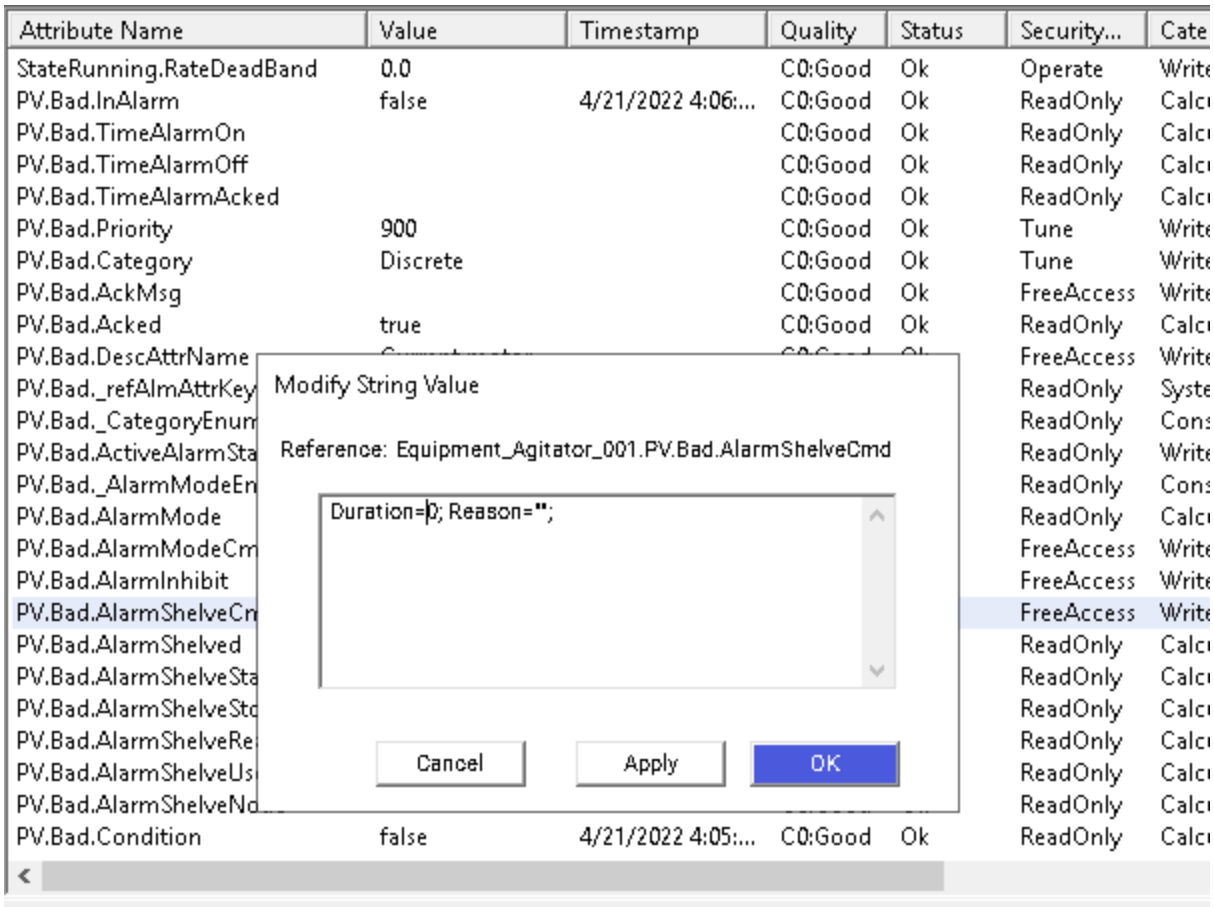
To shelve or unshelve an alarm in Object Viewer

1. In Object Viewer, select the object in the **Console Tree** (left pane) which contains the alarm you want to shelve. The object attributes display in the **Details Pane** (right pane).
2. In the Attributes list, find the attribute configured with the alarm you want to shelve.

   For example, "Inlet_001.Discrete_001".
3. In the Attributes list, below the attribute in step 2, find the **AlarmShelveCmd** attribute.

   For example, "Inlet_001.Discrete.001.AlarmShelveCmd".
4. Double-click the .AlarmShelveCmd attribute. The **Modify String Value** dialog opens.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.



a. To shelve an alarm, enter a duration in hours or in decimal fractions if the duration is to be less than one hour. The duration cannot be blank.

b. Enter a reason between the quotation marks for shelving or unshelving. The reason cannot be blank. Click **OK**.

c. To unshelve a shelved alarm, enter a value of "0".

The six read-only alarm shelving attributes will display the current shelving status information.

Throttle alarms

Alarm throttling prevents network and message queues from flooding during periods of high alarm activity. Throttling sets a maximum number of transitions into and out of an alarm state within a defined period. Alarm transitions that exceed the throttling limit are not reported.

The WinPlatform object includes tuning parameters that specify the maximum alarm rate per second on an engine. The Alarm throttle limit attribute must be used in conjunction with the scan period to determine the maximum number of alarms that can occur in a scan cycle. You can set the alarm throttle limit when you configure the WinPlatform object. For more information about setting the scan period and alarm throttle limit, see Configure WinPlatform object alarms.

Alarm messages can be throttled for alarms going into alarm and out of alarm state. Alarm acknowledgement messages are not throttled. Users can still acknowledge alarms even when alarm throttling is active. Users can still disable alarms for objects or areas when the alarm rate causes throttling to occur. Also, the alarm inhibit and the disable/enable/silence messages are not throttled.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

If an active alarm is disabled, the going out of alarm and disabled messages are sent to the notification distributor. If alarms are being throttled, the going out of alarm message can be throttled. The disabled message is accepted regardless of the throttling status. The going out of alarm message is not raised again. This can result in never logging a message for that alarm indicating it went out of alarm. The final alarm list in the Notification Distributor still shows the correct alarm state.

Propagate timestamps with alarms and events

An alarm feature always registers a reference to the alarmed attribute. This registered reference guarantees that Message Exchange never suspends updates for the alarmed attribute. Even if alarms are disabled for a particular attribute, the Advanced Communication Management feature cannot suspend the attribute.

Be aware that the time stamp propagates to all masked bits of an integer attribute, even if only one of the bits changes.

For example, you have an Integer address in a PLC that represents 16 different alarm states. You assign ObjA.Attr_Integer to point to the PLC address. You then split the bits to different alarm attributes, adding one attribute for each alarm and naming them ObjA.Attr_Alarm00 to ObjA.Attr_Alarm15. Each attribute has an input source that refers to a different bit of ObjA.Attr_Integer. For example, ObjA.Attr_00.InputSource -> me.Attr_Int.00, and so on. At runtime, when bit 00 is changing in the PLC, all of the attributes (ObjA.Attr_Alarm00 to ObjA.Attr_Alarm15) get a new time stamp, as all the bits changed. This can cause incorrect time stamps for alarms.

Alarms or events become active

For alarms or events, if an attribute has no timestamp, the current AppEngine scan time is used instead. If an attribute has a timestamp, the timestamp of the value that caused the alarm to occur is used as the value for the object's AlarmOnTime attribute.

When an alarm is silenced or disabled, and an alarm condition becomes TRUE, when the alarm is later enabled, the timestamp for AlarmOnTime is the timestamp of the most recent attribute change, not the time at which the alarm was enabled.

For an event, if an attribute has a timestamp, the timestamp of the value that caused the event to be reported is used for the EventTime. A System Event timestamp is the current system time.

Alarms become disabled

If an active alarm becomes disabled, the alarm is forced to return to normal. The timestamp corresponds to the current time when the alarm became disabled, not the timestamp of the attribute nor of the alarm condition. Otherwise, the assigned timestamp is the AppEngine scan time.

Alarms revert to normal

If an attribute value has a timestamp, the timestamp of the value that caused the alarm to revert to normal is used for the AlarmOffTime.

If alarm is based upon several attributes or upon several values of a single attribute, the most recent timestamp is used when assigning values to the AlarmOnTime or AlarmOffTime.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Alarm acknowledgement

Alarms are acknowledged by users who view unacknowledged alarms from their client applications like InTouch HMI or AVEVA OMI. Only alarms of certain priority levels need to be acknowledged.

The basic workflow to acknowledge an alarm consists of the following general steps:

- The alarm is detected and reported to any subscribed alarm clients. The alarm is unacknowledged unless it is of a priority level that does not require acknowledgement.
- An authorized user attempts to acknowledge the alarm from the client. The user can type an optional comment when acknowledging the alarm.
- The user's acknowledge request is sent to the detecting object's alarm feature. The acknowledgment must pass through standard security checks first. The acknowledge request contains the user's name and any alarm comment.

For the alarm utility, the alarm is acknowledged within the alarm feature immediately. The user name, comment, and acknowledged time are also saved. Alarm comments can be localized into any supported language.

The acknowledge is considered an alarm state change, which is sent to all subscribed clients. When an alarm is acknowledged, the current AppEngine timestamp is used as the acknowledgement time.

After an alarm is acknowledged for the first time, any additional (extra) acknowledgement attempts for the same alarm are rejected and an error is returned.

Acknowledge alarms when signature is required

You can use the SignedAlarmAck( ) script function in Industrial Graphics to require a signature to acknowledge alarms. See "Signature Security for Acknowledging Alarms" in *Creating and Managing Industrial Graphics User Guide* for information about adding the script function to a graphic. Then, embed the graphic in the WindowMaker window or an AVEVA OMI layout.

When you acknowledge the alarms at runtime, the script function checks for a required signature. If a signature is required, it checks Galaxy security settings and alarm priority. If any alarm in the list falls within the configured conditions, you must provide your signature to acknowledge the alarms. You must sign in to acknowledge the alarms if no user is logged on to the InTouch or OMI application.

If any alarm in the list is requires a signature, then a signature is required for the entire list of alarms.

Note: The SignedAlarmAck( ) script function supports only System Platform alarms.

By inspecting the Alarm Viewer or the Alarm Control client, you can determine if the alarm acknowledgements were successful.

Dismiss a latched alarms

When the alarm latched state is enabled, alarms will continue to be shown, even after the alarm has been acknowledged and the out-of-range value that triggered the alarm has returned to normal. The user must issue a dismiss command to remove the latched alarm from the display. See Alarm latching for additional information.

Alarms are dismissed by users who view the latched alarms from their client applications such as InTouch HMI or AVEVA OMI. Latched alarms continue to show in the run-time application, until an authorized user issues a dismiss command.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Key points about latched alarms

Only users with Ack permission can issue a dismiss command to clear a latched alarm.

The basic workflow to dismiss an alarm consists of the following general steps:

- The alarm is detected and reported to any subscribed alarm clients. The alarm is unacknowledged unless it is of a priority level that does not require acknowledgement.
- An authorized user attempts to acknowledge the alarm from the client. The user can type an optional comment when acknowledging the alarm.
- The user sends an acknowledge request to the detecting object's alarm feature. The acknowledgment must pass through standard security checks first. The acknowledge request contains the user's name and any alarm comment.
- The alarm condition returns to normal but continues to be displayed in the subscribed alarm clients as a latched alarm.
- The user sends a dismiss command. This is subject to same safeguards as an acknowledgement request.
- The latched alarm is removed from display of the subscribed alarm clients, but the user who issued the dismiss command and the time the command was issued is historized.

For the alarm utility, the alarm is acknowledged within the alarm feature immediately. The user name, comment, and acknowledged time are also saved. Alarm comments can be localized into any supported language.

Acknowledge and dismiss are considered a change to the alarm state, which is sent to all subscribed clients. When an alarm is acknowledged or dismissed, the current AppEngine timestamp is used as the acknowledgement or dismissed time.

After an alarm is acknowledged for the first time, any additional (extra) acknowledgement attempts for the same alarm are rejected and an error is returned.

Configure alarms

Alarming capabilities are a part of object templates, but they are not implemented until you configure the object in the IDE. After alarms are configured, you can view Application Server alarms in a client visualization application like InTouch HMI or the AVEVA OMI Alarm App.

You can add latched alarms as an additional alarm state for the Galaxy to provide an extra level of visibility and oversight to alarm conditions. See Alarm latching for more information.

Configuring an object to be an alarm provider includes the following general sequence of steps:

1. Decide whether alarm notification is needed for each possible alarm condition of an object. For example, a command time-out alarm for a valve if the output command fails to move the valve.
2. Edit the object and set an attribute that specifies alarming.
3. Edit the object from the IDE and assign values to alarm attributes.
4. Configure the alarm properties. Typically, the fields that require configuration are Category, Priority and Description.
5. Configuring any limit fields to set an alarm. For example, the feedback time-out time limit.

You can add alarm detection and reporting capabilities to objects that were not originally developed to detect

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

alarms. You do this by configuring alarm features for the object's attributes.

Configure alarms for system objects

To enable alarming for your application, you need to configure your Galaxy's WinPlatform and AppEngine objects as alarm providers. Both system objects report their own alarms.

Client applications subscribe to application object alarms by the area containing the objects. Client applications can also subscribe to WinPlatforms and AppEngines directly. These are called "pseudo-areas." They do not need to be assigned to an area for a client to see the alarms, although the user may want to assign them to an area, such as for simplifying the alarm query in the InTouch Alarm Viewer or Alarm DB Logger.

The following list shows the alarms for each system object.

- WinPlatform
  - Excess CPU load alarm
  - Low disk space alarm
  - Excessive page faults alarm
  - Low memory alarm
  - Engine failure alarm
  - Engine checkpoint failure alarm
  - Object quarantined condition
  - Subscription folding condition
  - Scheduler scan overrun condition

    A WinPlatform object includes a general communication alarm when it loses contact with the areas to which it is subscribed.
- AppEngine
  - Checkpoint failure alarm
  - Object quarantined condition
  - Subscription folding condition
  - Scheduler scan overrun condition
  - Redundancy failover alarm
  - Redundancy Standby unavailable
  - Redundancy Standby not ready
- ViewEngine
  - Checkpoint failure alarm
  - Object quarantined condition
  - Subscription folding condition
  - Scheduler scan overrun condition

WebViewEngine, Area and InTouchViewApp system objects do not include any alarms.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Configure WinPlatform object alarms

You select the areas of your Galaxy to monitor for alarms from the WinPlatform object. Also, you can select specific alarms for the status of the WinPlatform itself.

You must specify that the WinPlatform object is an InTouch alarm provider to subscribe to alarms from the various areas (and pseudo-areas) of the Galaxy and report them to the InTouch Alarm Manager.

Configure alarm priority for communication failure

InTouch Alarm Provider reports any communication outage to the InTouch Alarm Manager. Communication alarms are displayed and must be acknowledged. By default, the priority of a communication failure is "1", the highest priority.

There can be instances when a communication failure is not a high-priority alarm, and should not require the attention of a high-priority alarm. Shutting down or undeploying a platform as a normal operation can result in communication failure alarms, for example.

To indicate whether the alarm is of high or low importance, the communication failure alarm priority is configurable at design time as an integer of range 1–999. InTouch Alarm Provider must be enabled in the WinPlatform object editor, **General** tab.

Register using "Galaxy_<GalaxyName>" instead of "Galaxy" option

When you select the **Register using "Galaxy_<Galaxy name>" instead of "Galaxy" option**, you enable ITAlarmProvider contained in WinPlatform to register the ITAlarmSubsystem using a provider name of Galaxy_GalaxyName.

For example, if your galaxy is named Andromeda, the ITAlarmSubsystem registers as Galaxy_Andromeda.

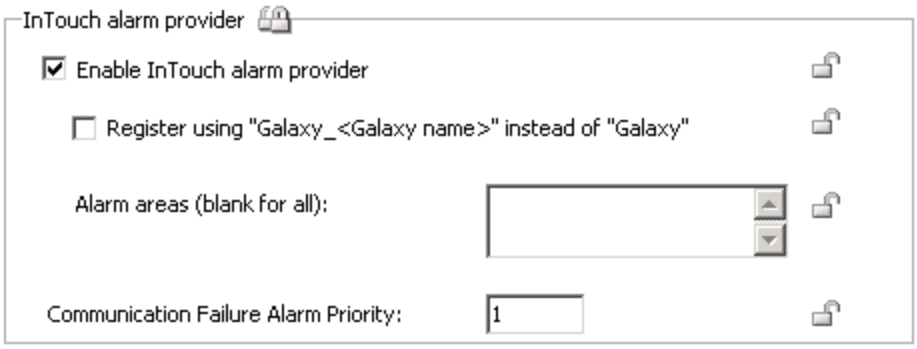
This configuration can be useful if you are using an Alarm Client to display alarms from multiple galaxies. For more information about multiple galaxies, see The multi-galaxy environment .

- This means that all InTouch applications which query alarms from this platform should be modified to use Galaxy_GalaxyName!AreaName.
- Select this option when the Alarm Control, the embedded alarm client (EAC), or the AlarmViewControl in InTouch queries alarms from different galaxies.

To configure a WinPlatform object to be an alarm provider

1. Open the WinPlatform object in the Object Editor.
2. Click the **General** tab.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.



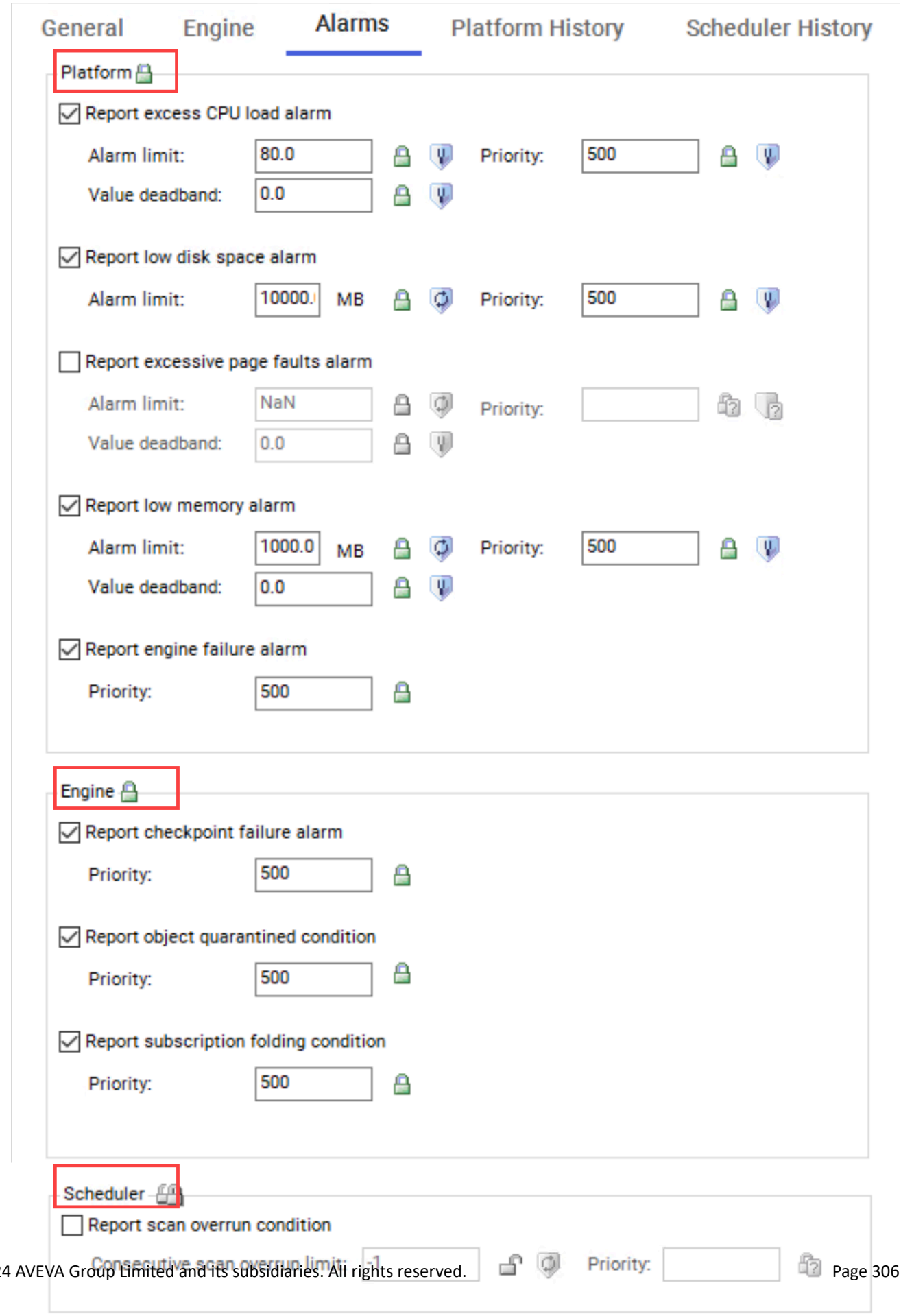
3. Select the **InTouch alarm provider** check box.
4. Select the **Register using "Galaxy_<GalaxyName>" instead of "Galaxy"** check box to enable Galaxy_<GalaxyName> registration for alarm comment language switching. An information box appears. Click **OK** on the information box to continue.
5. In the **Alarm Areas** box, type the names of areas to subscribe to for alarms.

   If you leave the **Alarm Areas** box blank, the WinPlatform subscribes to all areas in the Galaxy.

   If you want to subscribe to only selected areas within the Galaxy, insert a space between each area name. For example:

   Area1 Area2 Area3
6. Enter a number from 1 to 999 to configure the **Communication Failure Alarm Priority**.
7. Click the **Engine** tab to show the **Alarm throttle limit** box. Either accept the default throttle limit of 2000 alarms per second, or enter another value. A value of 0 disables alarm throttling. For more information about alarm throttling, see Throttle alarms.
8. Click the **Alarms** tab to show platform and engine, and scheduler alarms that can be set for the WinPlatform object.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.



© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

9. Select the check box next to each alarm that you want to enable for the WinPlatform object.
10. Set the limit, value deadband, and priority for each alarm you selected.
11. Save and close the Object Editor.
12. Check the object in to the Galaxy.
13. Deploy the object in an on scan state.

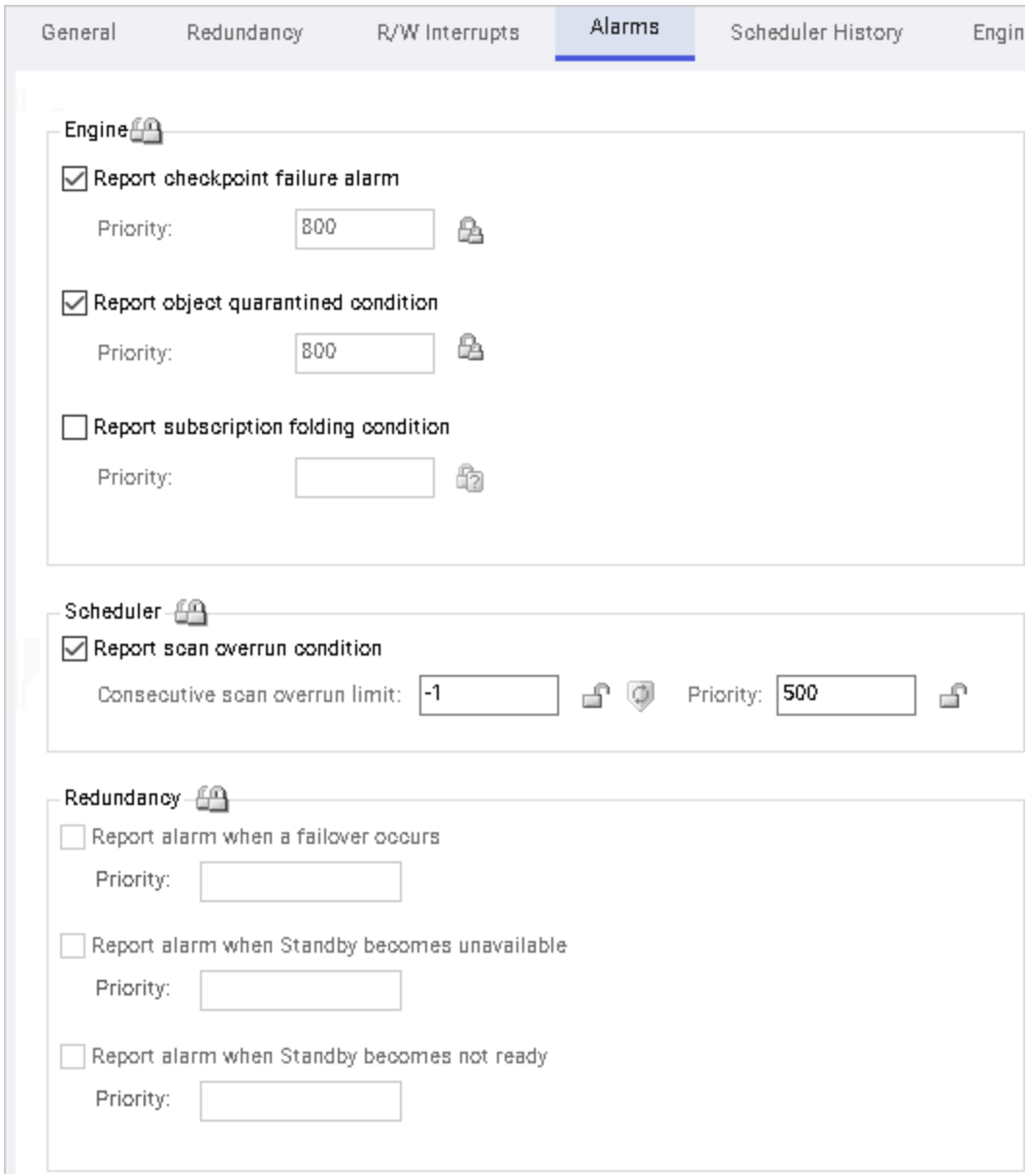
Configure alarms for an AppEngine object

You can set AppEngine attributes that determine whether alarms are enabled for an engine, scheduler, and redundant failover engine.

To configure AppEngine object alarms

1. Open the AppEngine object in the Object Editor.
2. Click the **Alarms** tab to show engine, scheduler, and redundancy alarms that can be set for the AppEngine object.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.



3. Select the check box next to each alarm that you want to enable for the AppEngine object.
4. Set the priority for each alarm you selected.
5. Save and close the Object Editor.
6. Check the object in to the Galaxy.
7. Deploy the object in an on scan state.

Configure alarms and events for application objects

The following table shows the different types of alarms that can be specified for application objects. The table shows the application objects containing native alarm attributes.

You can also set alarms for an object's attributes. The types of alarms available depend on the attribute type. For more information about setting alarms for attributes, see Set alarms on the attributes page.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Application Object	Alarm Types				
State	**Limit**	**Target Deviation**	**Rate of Change**	**Statistics**	
AnalogDevice	X	X	X	X	
DiscreteDevice	X				X
FieldReference					
Sequencer	X				
SQLData					
UserDefined	X	X	X	X	

The following list shows the types of alarms for each application object in more detail.

- AnalogDevice
  - Level alarms (HiHi, Hi, Lo, LoLo) [limit alarms]
  - Rate of Change alarms (Up, Down)
  - Target Deviation alarms (Minor, Major)
  - PV Bad Value alarm [state alarm]
- DiscreteDevice
  - Uncommanded change alarm [state alarm]
  - Command time-out alarm [state alarm]
  - Active1 state alarm [state alarm]
  - Active2 state alarm [state alarm]
  - Fault state alarm [state alarm]
  - Active1 state duration alarm [statistical alarm]
  - Active2 state duration alarm [statistical alarm]
- Sequencer
  - Execution halted [state alarm]
  - Condition trigger failure [state alarm]
  - OnEntry output failure [state alarm]
  - OnExit output failure [state alarm]
- UserDefined (Attributes can be alarmed)
  - PV State alarm [state alarm]
  - PV Bad Value alarm (that is, bad quality) [state alarm]
  - Attribute alarm features

    State alarms

    Limit alarms (HiHi, Hi, Lo, LoLo)

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Rate of Change alarms (Up, Down)

Deviation alarms (Minor, Major)

Bad Value alarm (that is, bad quality)

There are no built-in alarms for these application objects:

- FieldReference
- SQLData

You can also configure your application objects to generate an event each time the object's PV value changes.As mentioned above, you can also configure an alarm on any object for any attribute. See Set alarms on the attributes page for more information.

To configure alarming and events for application objects

1. Open the application object with the Object Editor.
2. For a DiscreteDevice, click the **General** tab, enable inputs, and then enable **Generate event upon PV**.

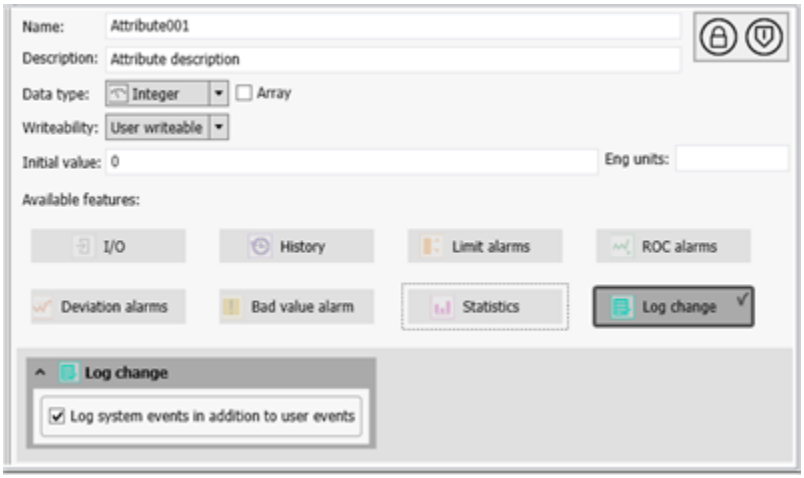


For objects that do not include this check box, use the **Attributes** page.

Note: If you are using field attributes instead of Attributes, use the **Field Attributes** page.

a. Enable the **Log change** feature.

b. Enable **Generate event upon change**.



3. Select or clear the check box based on whether you want to generate an event each time the object's PV value changes.
4. Click the tab that lists alarm attributes.
   - For the AnalogDevice object, click **Alarms**.
   - For the DiscreteDevice object, click **Alarms**.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

- For the Sequencer object, click **Settings**.
- For the UserDefined object, click **Attributes** (or **Field Attributes**, if you are using field attributes). See Set alarms on the attributes page for further directions.

5. Select the check box that enables alarming for the object.
6. Assign values to the attributes for the type of alarm you selected by completing the following steps:
   a. Assign values to the alarm limits based on the type of alarm.
   b. Assign an alarm priority (1-999) for each limit you set.
   c. Accept the default alarm message or include another message for each alarm limit.
   d. Assign values to the remaining attributes based on the type of alarm you selected. For more information about other alarm attributes, see Types of alarms.
7. Save your object changes and close the Object Editor.

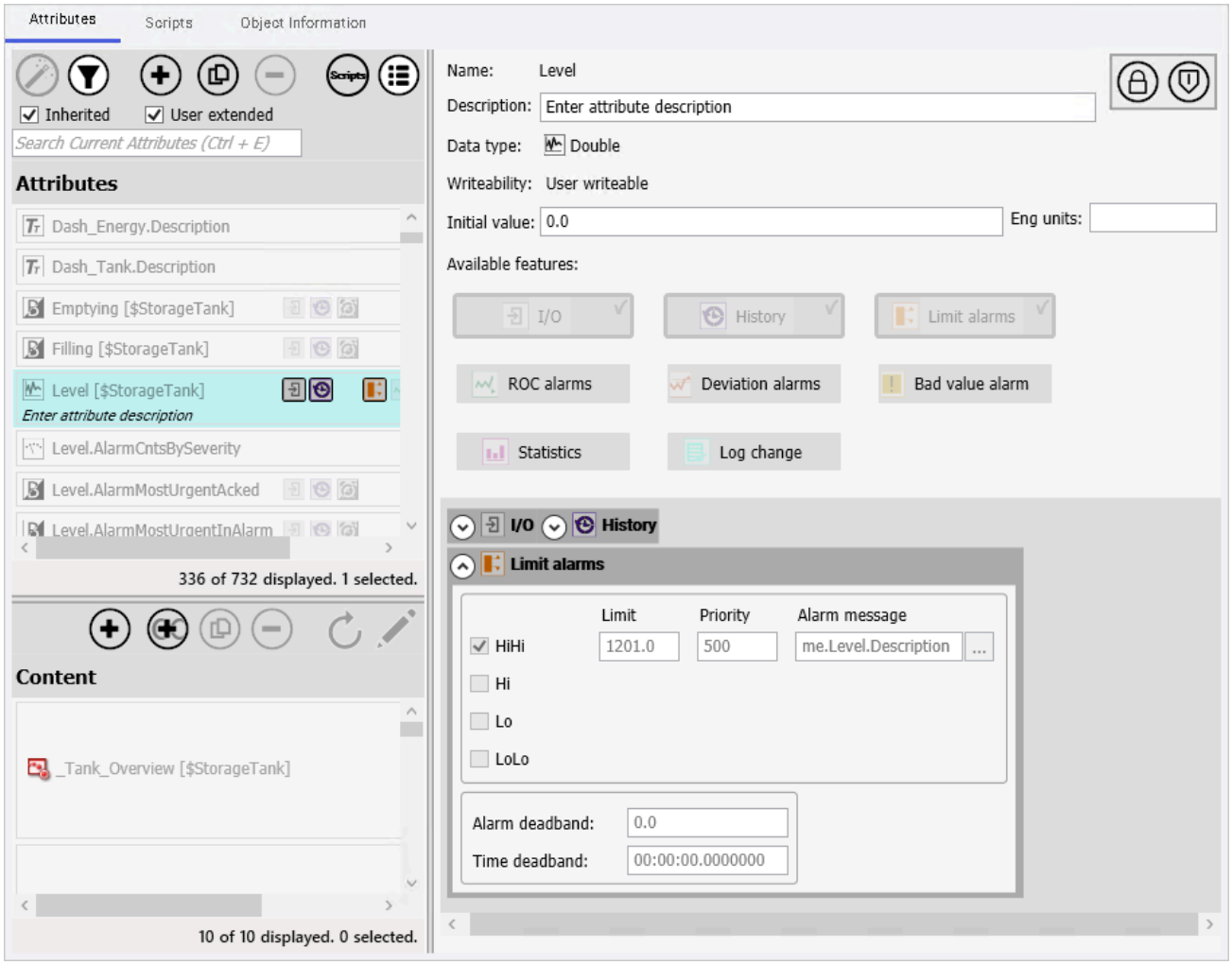
Set alarms on the attributes page

You set alarms for all Application Server objects in the Attributes page in the Object Editor.

To specify alarms for an object's attributes

1. On the **Attributes** page of the Object Editor, select an attribute from the **Attributes List**.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.



2. Select the alarm features you want to configure for the attribute. For Boolean data types, you can select:
   - **I/O Alarm**
   - **State alarm**
   - **Bad value alarm**

   For Integer, Float and Double data types, you can select:
   - **Limit alarms**
   - **ROC alarms**
   - **Deviation alarms**
   - **Bad value alarms**

   Note: Depending on the object type, some of these alarm types may not be available.
3. For each alarm feature that you activate, enter the settings for each alarm.
   a. Assign values to the alarm limits based on the type of alarm.
   b. Assign an alarm priority (1-999) for each limit you set.
   c. Accept the default alarm message or include another message for each alarm limit.
   d. Assign values to the remaining attributes based on the type of alarm you selected. For more information about other alarm attributes, see Types of alarms.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

4. To configure event logging for an object, select the **Log change** feature and click the checkbox **Log system events in addition to user events**.
5. Save your object changes and close the Object Editor.

Distribute alarms and events

After you configure object instances for alarm detection, deploy the instances and put them On scan. The instances begin checking for alarm conditions.

When an alarm is detected, or an event occurs, a notification is reported to its alarm and event distributor, which is running on the same AppEngine.

These alarm and event distributors include:

- **Area objects**: Area objects report detected alarms through the Area, which distributes them to alarm and event clients.
- **WinPlatform objects**: Report their own alarms and events.
- **AppEngine objects**: Report their own alarms and events.
- **Device Integration objects**: Report their own alarms and events

The Area object plays a key role in alarm and event distribution. All objects belong to an Area. Areas can contain sub-Areas. Alarm and event clients are configured to subscribe to a set of Areas.

Areas provide a key organizational role in grouping alarm information and assigning it to users who use alarm and event clients to monitor their Areas of responsibility.

WinPlatforms, AppEngines and Device Integration objects do not report their alarms and events to Area objects even though they belong to Areas. This allows alarm clients to receive alarm notifications without any dependencies on Area objects. For example, a deployed and running WinPlatform can report alarms even though its Area is not deployed and running.

Alarm-event distributor objects maintain a list of all currently active alarms and inactive but unacknowledged alarms. They do not maintain a list of events, which are routed to clients that are currently subscribed at the time of the event.

You can configure a WinPlatform to act as an InTouch Alarm Provider in the runtime environment.

The WinPlatform sends an alarm through the InTouch Distributed Alarm System to InTouch clients when the WinPlatform loses communication with an Area that it subscribes to. This condition typically occurs during a network outage with computers hosting those Areas.

In a network outage, the WinPlatform InTouch Alarm Provider sends an alarm for each disconnected Area that it subscribes to, including all of its alarm distribution hierarchy. Each of these alarms is a high priority alarm that contains the name of the Area to which communication is lost. These communication problem alarms must be acknowledged.

Although they still appear in the historical record, any current alarms from the disconnected Area drop from the InTouch client’s summary list. They can no longer be acknowledged.

When communication to the disconnected Areas is restored, any unacknowledged alarms generated in those Areas are sent to the alarm client.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Subscribe to alarms and events from a client

Clients indicate interest in alarms and events by subscribing to an Area. When subscribing to an Area, the subscription is actually to all notification distributors within that Area.

For example, if an Area contains sub-Areas, those sub-Areas are subscribed to. If WinPlatforms, AppEngines or Device Integration objects belong to an Area, those objects are also directly subscribed to.

When a notification distributor receives an alarm and event subscription from a client, the notification distributor provides the client with the following:

- A list of all current alarm conditions, including unacknowledged return-to-normal conditions.
- An alarm condition state change. A state change includes transitions into or out of alarm (return to normal) and change in acknowledged flag.
- An event occurrence.

Alarm and event subscription requests do not include filters, for example, only show alarms greater than a specific priority value. All alarm and event messages received by the notification distributor are sent to all subscribed clients. Filtering is provided as a display option by clients.

Using InTouch HMI as the Alarm and Event Client

InTouch runtime clients subscribe to event reports from a Galaxy. Application Server reports alarms to the InTouch Distributed Alarm System, which subscribeS to alarm and event reports from a Galaxy.

An InTouch client application can visualize Application Server components. An InTouch alarm client can show alarm information for new, unacknowledged alarms, including all required fields.

The new alarm is in the unacknowledged state. An operator can view alarms, acknowledge alarms, disable alarms, and enable alarms from the client application running in InTouch WindowViewer.

Syntax of alarm queries

InTouch alarm queries subscribe to alarm and event information from objects within a Galaxy. The alarm and event queries can be in the form of user input or a script.

Alarm query syntax must be in one of the following forms:

\Provider!Area

or

\\Node\Provider!Area

You can have one or more references in a query separated by spaces.

You can also optionally append a tagname filter at the end, separated by another exclamation mark:

\Provider!Area!Filter

\\Node\Provider!Area!Filter

The filter can have a wildcard * character at the beginning or at the end, but not both.

The \\Node at the beginning is only important if you want to query for alarms from a provider on another computer. Otherwise, you can leave it off and the reference is assumed to be a provider on the local computer. The provider name Galaxy refers to alarms and events that get reported by the WinPlatform configured as an

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

InTouch alarm provider on that computer node.

Alarm query syntax when 'Register Using Galaxy_<GalaxyName>' is enabled

In the WinPlatform object, when you enable InTouch alarm provider, you can enable **Register using Galaxy_<GalaxyName> instead of Galaxy**. This option will register the platform to the alarm subsystem using the Galaxy name prefixed by "Galaxy_" instead of just the word "Galaxy". This allows an InTouch application to monitor alarms from multiple Galaxies and avoid name conflicts.

Syntax changes slightly when Galaxy_GalaxyName is enabled:

- Use \\ for computer name.
- Use \ for Galaxy or Galaxy_<GalaxyName>.
- Use ! for Area.

For example: \\Galaxy\MyGalaxy!Area001.

If Galaxy_GalaxyName is not enabled in WinPlatform, then the default behavior described in the previous section applies.

You can determine if Galaxy_<GalaxyName> has been enabled by monitoring the runtime attribute of the platform ITAlarmProvider.ProviderNameAsGalaxyNameEnabled.

Alarm query examples

- You can submit a query to get all alarms from Area1 and all other alarms within Area1, as reported by the WinPlatform object on the local computer.

  ```
  \Galaxy!Area1
  ```

  The query returns all alarms and events from all objects directly contained in Area1 and any sub-areas contained by Area1. This hierarchy is determined by what is configured in the Model View in the IDE.
- If Area1 and Area2 are two separate mutually exclusive areas, you can submit a query for alarms from both areas.

  ```
  \Galaxy!Area1 \Galaxy!Area2
  ```
- If you're on NodeA and the WinPlatform is on NodeB, you can submit a query for the alarms from the remote computer.

  ```
  \\NodeB\Galaxy!Area1
  ```
- You can submit a query for all alarms from objects whose name begins with "Tank" in the TankFarm1 area.

  ```
  \Galaxy!TankFarm1!Tank*
  ```

  The trailing wildcard character matches alarms from all objects with names that begin with "Tank" like Tank001, Tank002, TankUpper, or TankLower.
- You can submit a query for specific alarm types. For example, you can submit a query for all HiHi alarms in the TankFarm1 area.

  ```
  \Galaxy!TankFarm1!*.HiHi
  ```
- You can submit a query for all types of alarms from a specific object within an area.

  ```
  \Galaxy!TankFarm1!Tank752.*
  ```

  The trailing wildcard character matches all alarm types for Tank752.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Alarm requirements for InTouch client applications

For Application Server alarming to function, the following conditions must be met:

In Application Server:

- One or more Area objects are deployed and running.
- The source object is on scan.
- The source object's Area is on scan.
- Alarming must be enabled for the target object.
- An InTouch alarm provider on any WinPlatform in the Galaxy.

In InTouch:

- The InTouch client application is running in WindowViewer.
- An InTouch alarm ActiveX control is placed in a window and configured as an alarm consumer for the Galaxy.
- The user has logged into InTouch with security enabled and is authorized to acknowledge alarms for the object that is in the alarm state. If the user only wants to view alarms, security authorization is not required.

Application Server validates the user has sufficient security privileges to acknowledge the alarm.

If the user does not have privileges to acknowledge alarms, the user can attempt to acknowledge the alarm, but the Galaxy rejects the acknowledgment request.The alarm remains unacknowledged in the InTouch Alarm display.

The rejected alarm acknowledge event is recorded in InTouch Event History if the user attempting the acknowledgement has a valid Galaxy user account. Otherwise, the rejected acknowledgement is not recorded as an event.

You can acknowledge multiple alarms by providing a valid signature. At runtime, the Industrial Graphics SignedAlarmAck() script function checks a set of alarms and conditions to find out if any alarm from the list requires a signature in order to acknowledge it. If so, you must provide your user name, password and domain or your Smart Card details and PIN to authenticate yourself and acknowledge the alarms. To be able to provide the Smart Card details, your computer must be configured for Smart Card authentication.

Note: Smart Card authentication is not supported in multi-galaxy environments for read/write operations to remote galaxies.

If even one alarm in the list requires a signature, you must provide a signature to acknowledge the alarms. The SignedAlarmAck() function will acknowledge ALL the alarms in the list.

With the SignedAlarmAck( ) function, you can provide credentials and acknowledge an alarm or group of alarms even if you are not the logged-on operator.

Alarms and event implemenation in application server and runtime applications

Application Server and its runtime applications (AVEVA OMI and InTouch HMI) implement alarms and events. The following table shows the similarities and differences between the products.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Item	InTouch	Application Server/OMI
Alarm configured or detected by	Within a tag	Within an object
Alarm Classes (client column)	Only certain classes of alarms are supported or detected: DSC, VALUE, DEV, ROC, SPC.	No system-wide distinction for classes. Alarms are tied to a Boolean that can be triggered from any logic.
Alarm Type (Sub-class) (client column)	Discrete, LoLo, Lo, Hi, HiHi, MinorDev, MajorDev, ROC, SPC. Client column.	No sub-class. The Alarm feature name is the closest concept. For example, ".PVHiAlarm". Mapped from Category.
Priority (client column)	1-999 (1 most urgent)	Priority 0-999. 0 most urgent. 0 is mapped to 1 in InTouch.
Name (client column)	Alarm name = Tag name.	Object.attribute
Comment (client comment)	Separate alarm comment, which is different from the tag comment.	Object short description or alarm message where available.
Group	Alarm group allows client-side filtering. Sub-groups must be on same InTouch.	No alarm group. But Area provides mappable concept. Sub-Areas can be on different nodes.
State	No behavioral differences between Application Server/OMI and InTouch HMI. Four states, which are combinations of ACK/UNACK and ALARM/RTN. Optionally, a fifth state, Latched, can be enabled for the Galaxy. • UNACK/ALARM (usually displayed as UNACK) • ACK/ALARM (usually displayed as ACK) • UNACK/RTN (usually displayed as UNACK_RTN) • ACK/RTN (usually displayed as ACK_RTN) • LATCHED (optional) These states have a 1:1 correspondence with states of the Alarm feature, which keeps track of whether the alarm is InAlarm and IsAcked. Alarms in the state ACK/RTN are not shown in the SUMMARY alarm display because they do not need any further action from the operator unless Latched is enabled for the Galaxy. But, all four states appear in the HISTORY display, and in the Alarm Database. If the Latched state is enabled, ACK/RTN alarms remain visible in the	

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Item	InTouch	Application Server/OMI
	alarm display until the operator sends a Dismiss command.	
Value, CheckValue	Only static values sent with alarm message.	Static values and dynamic references are provided.
Ack	No behavioral differences between Application Server/OMI and InTouch HMI. All alarms are sent to the client and require acknowledgement regardless of priority.	
Dismiss	If the Latched alarm feature has been enabled for the Galaxy, all alarms in LATCHED state require a dismiss command.	
History	No behavioral differences between Application Server/OMI and InTouch HMI. Alarm state changes are logged to event history and shown on historical client.	

Configure plant state-based alarms

You can map alarm modes on a per-Galaxy basis to different plant operational states to control how alarms are reported. Five plant states are pre-defined, and have default alarm states associated with them:

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Plant state	Default alarm state	Available alarm states
Running	Enable	Enable
Maintenance	Disable	Enable / Silence / Disable
Startup	Silence	Enable / Silence / Disable
Shutdown	Disable	Enable / Silence / Disable
Testing	Silence	Enable / Silence / Disable

You can define new plant states, rename plant states, or remove existing plant states, except the "Running" state (you can, however, rename "Running"). The alarm state for Running is read-only and cannot be changed from Enable.

After you have defined plant state-based alarm configurations for the Galaxy, you can assign plant state-based alarming to area objects in the Galaxy. This is done by setting the PlantState attribute for each area object that will use plant state-based alarming. The area object will automatically update its PlantAlarmMode attribute to match the alarm state that is set for the PlantState currently assigned to it.

Attribute	Definition
PlantState	The name of the currently assigned plant state (Running, Maintenance, etc.).
PlantStateAlarmMode	The alarm state of the assigned plant state (enable, silence, or disable). This attribute is read-only at runtime.

Mapping alarm modes to plant states

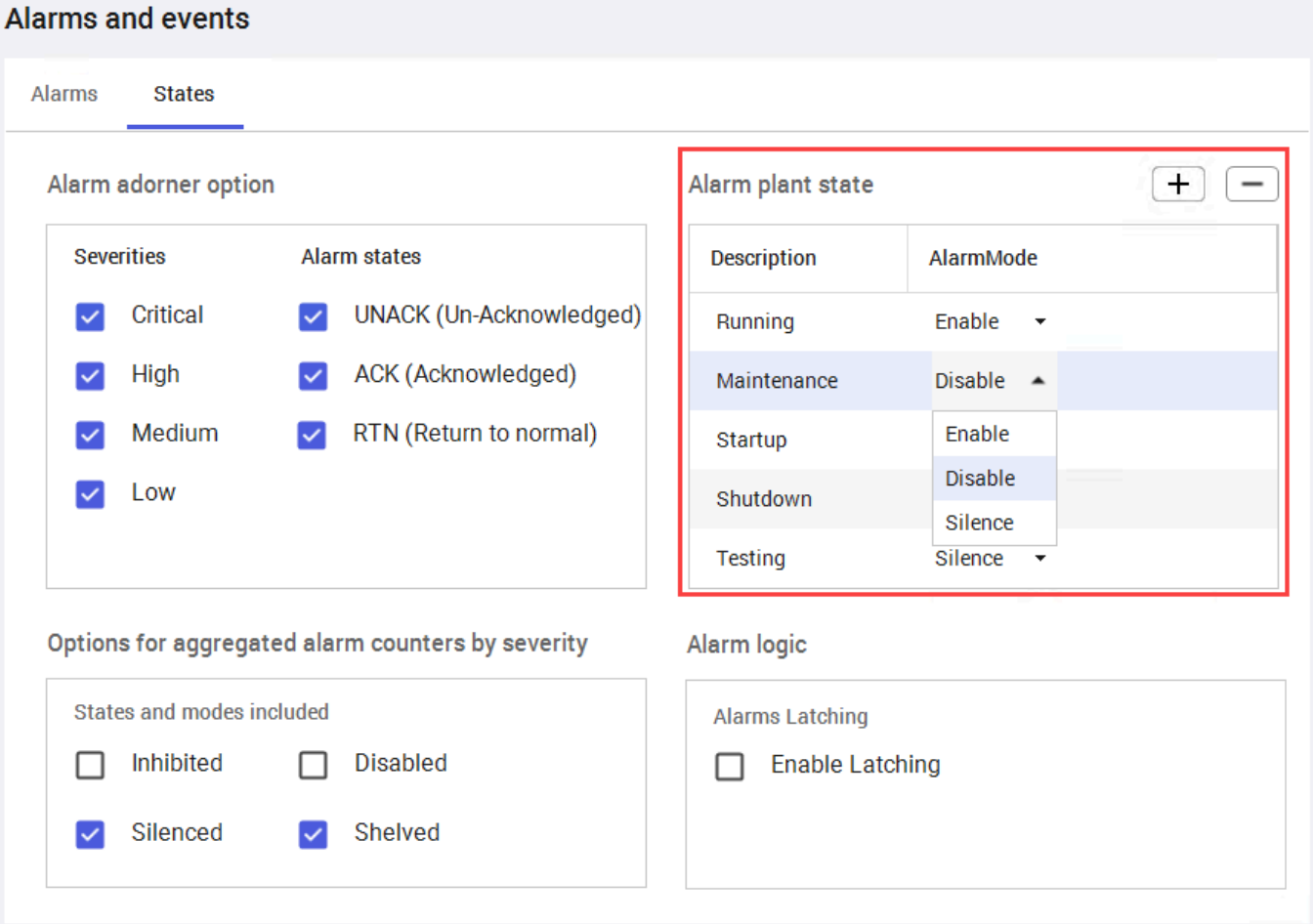
Define plant states and map them to alarm modes in the IDE. To access the configuration dialog:

1. From the ribbon, select **Galaxy,** then select **Configure.**
2. Select **Galaxy**, then select **Alarms**. The **Alarms** page appears.

   Note: Settings in this dialog are not shared across Galaxies in a multi-Galaxy environment. Each Galaxy in the environment will have its own alarms configuration.

To use the default values, you do not have to set anything in this dialog. Simply enable plant state-based alarms for each area object that is to use this feature. See Configure the state-based alarm mode on an area object for additional information.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.



Change alarm modes and modify plant states

- To change the AlarmMode value, click the **AlarmMode** associated with the Plant State that you want to change. Then, select the new value (enable, silence, or disable).

  Note: **AlarmMode** for Running is read-only and cannot be changed.

- To add a new Plant State, Click the **Add** (+) button. A new row is created at the bottom of the table.

  Enter the **Description** of the new plant state. The maximum length is 64 characters.

  Select a value for the **AlarmMode** (enable, silence, or disable).

- To delete an existing Plant State, click the **Remove (-)** button. The selected plant state and its associated value are removed from the table.

  Note: You cannot delete the first row of the **Alarm plant state** table. Running is the default PlantState **Description** in the first row.

- To change the description of an existing PlantState, double click the plant state you want to rename, then enter the new description. The maximum length is 64 characters.

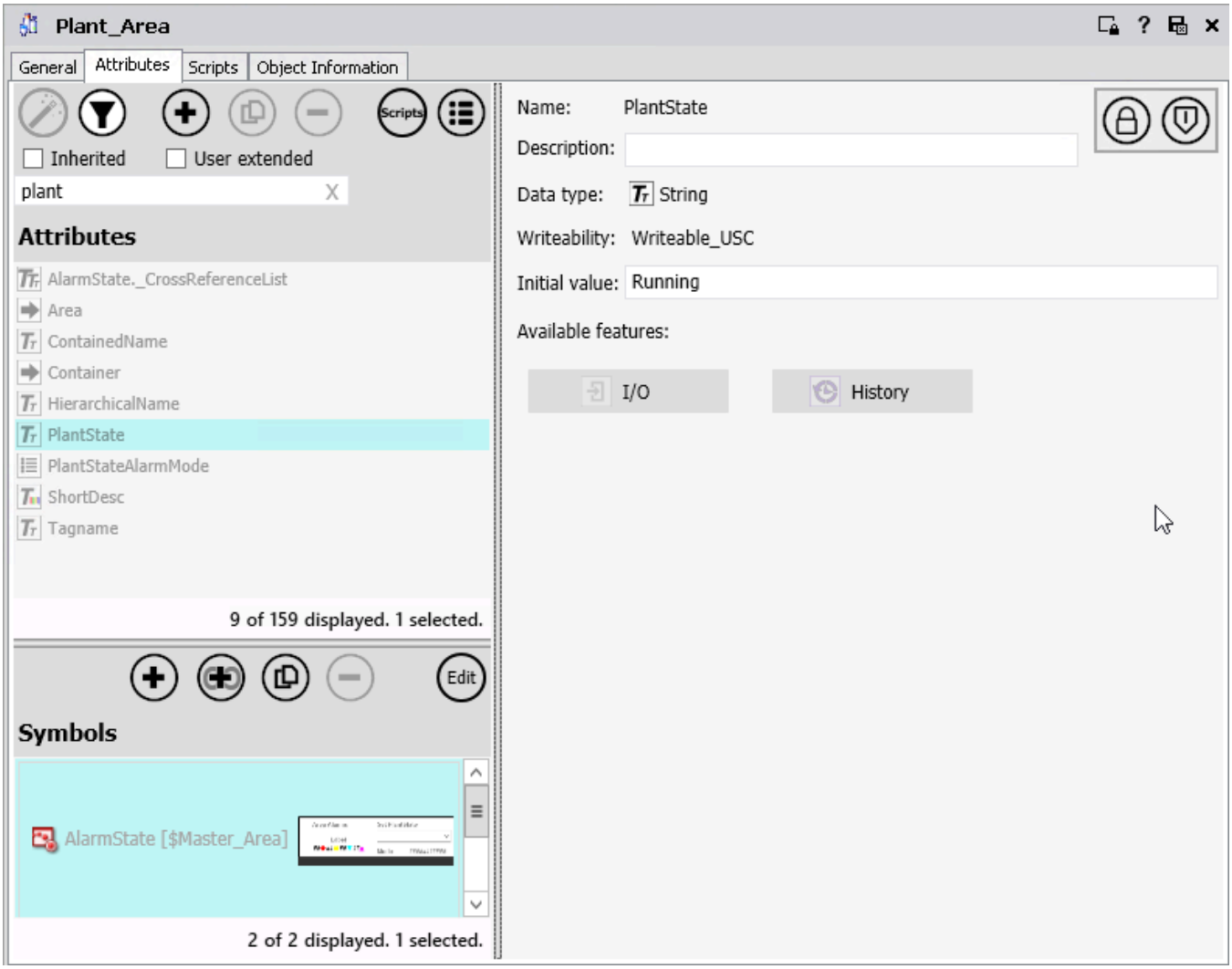
  Note: All Plant States, including Running, can be renamed.

Configure the state-based alarm mode on an area object

State-based alarm modes apply to String attributes, with writeability set to Writeable_USC (these cannot be

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

changed). To use state-based alarming for one or more areas, you must configure the PlantState attribute for each area that will use plant state-based alarming.



Configure plant state alarming

1. Open the area object and click the Attributes page.
2. Select the PlantState attribute.
3. Enable the **I/O** feature for PlantState and select Read, Read/Write, or Write.
4. Enable the **History** feature and configure it.
5. Select the PlantStateAlarmMode attribute.
6. Repeat steps 3 and 4 to configure the **I/O** and **History** features for PlantStateAlarmMode.

View plant state-based alarms at runtime

You can view plant state-based alarming at runtime through clients such as Object Viewer and InTouch applications in WindowViewer.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

User access and security at runtime

Operators with the appropriate permissions can change the PlantState of an area through a runtime client. An operator may have to change PlantState from Startup to Running, for example. Scripts can also be used to implement modifications.

To change a PlantState, the operator enters a string to match one of the defined PlantState values (for example, "Running"). Before a change to PlantState can be implemented, the system checks that the user belongs to a role that has the permission "Can Modify Plant State of an Area" and is therefore authorized to make the change. If the operator enters a string that does not match a defined PlantState, the change is rejected.

At runtime, the AlarmMode attribute is read-only. Therefore, the AlarmMode for a PlantState cannot be changed through a runtime client (for example, changing from enable to silence). It can only be changed through the IDE.

If an area is assigned to a PlantState and the PlantState is deleted through the IDE, the area will remain in that PlantState until it is changed. If AlarmMode for the deleted PlantState is anything other than enable, the AlarmMode will change to enable.

Note: Operators with appropriate permissions can use the AlarmModeCmd attribute to change AlarmMode of an area at runtime. However, the AlarmModeCmd can only be used to set a more restrictive condition than the AlarmMode of the area's corresponding PlantState. For more information about changing alarm modes, see Enable, silence, and disable alarms.

Configure priority ranges for alarm historization, mapping, and shelving

The **Alarms and Events Configuration** dialog lets you:

- Map alarm severities to priority ranges.
- Enable alarm shelving by priority range.
- Enable or disable historization of both alarm severities and event types. All alarms are historized by default.

Default Alarm Mapping and Historization Values					
Severity	**Description**	**Shelve**	**Historize**	**From Priority Range**	**To Priority Range**
1	Critical	N	Y	1	250
2	High	N	Y	251	500
3	Medium	Y	Y	501	750
4	Low	Y	Y	751	999

Shelving, historization and priority ranges are configurable.

Mapping alarm severity to priority

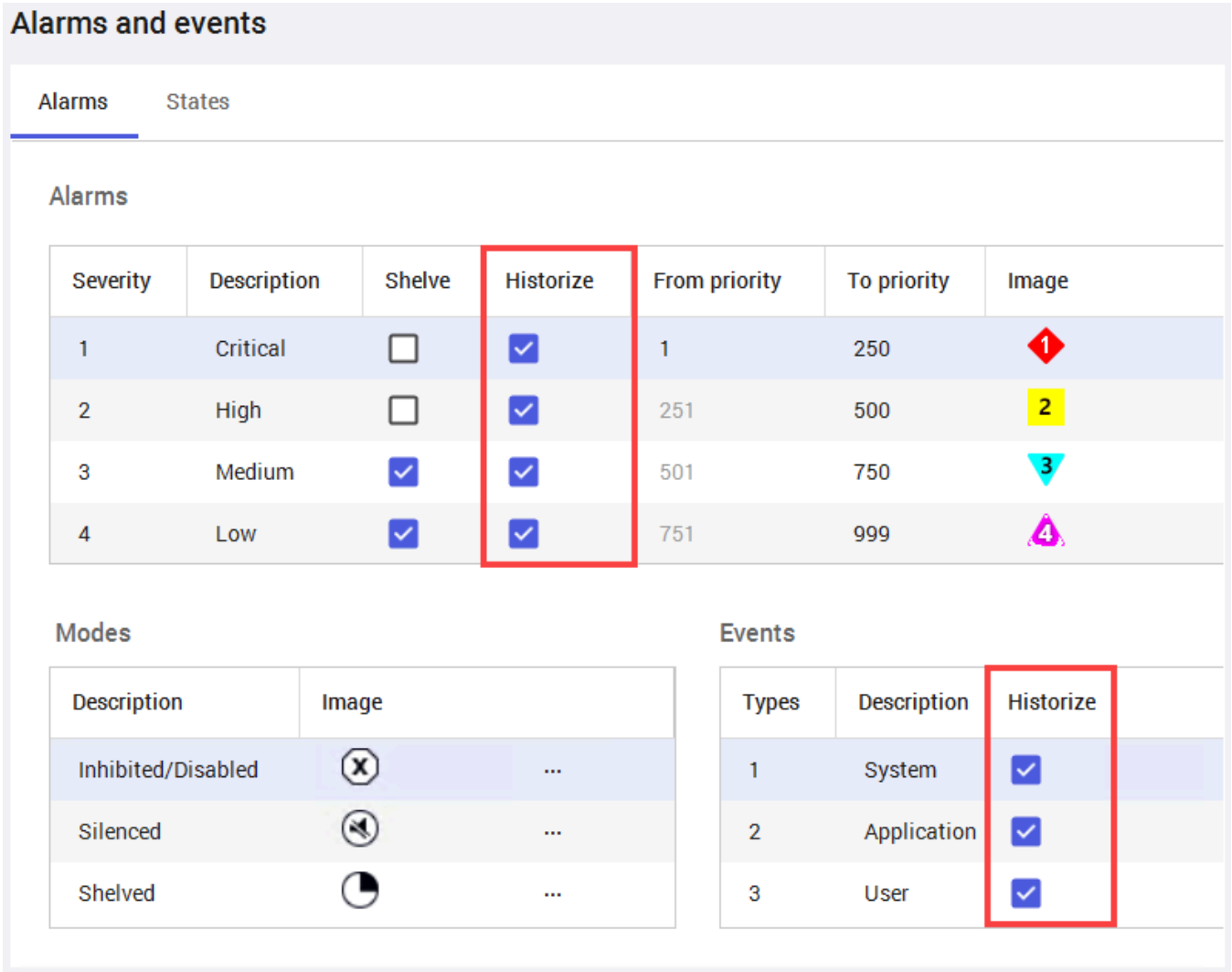
- Severity 1 starts at priority 1 by default. Severities can be mapped to priorities in ascending or descending order. For example, severity 1 can map to priority range 1–250 or it can map to priority range 999-751.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

- Severity 4 ends at 999 by default, but 999 is not required to be the end-of-range number. For example, severity 4 can end at 900, leaving a priority range above 900 unmapped to any severity level.
- A priority outside the configured priority ranges will return severity 0, not mapped. Since the priority is unmapped, it cannot be historized.

Note: Mapped alarm severities are not shared across galaxies in a multi-galaxy environment. Each galaxy in the environment has its own priority to severity mapping.

Configure historization for alarms and events



Historization is configurable for both events and alarms. Event types and the default historization setting for each type are:

Default event type historization values		
Types	**Description**	**Historize**
1	System	Y

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Default event type historization values		
2	Application	Y
3	User	N

Note: In prior versions of Application Server, historization of alarms and events was configured through the Alarm DB Logger Manager (an InTouch component) and utilized the SQL database "WWALMDB." The method described here replaces the Alarm DB Logger method.

To configure alarm and event mapping

1. From the ribbon, select **Galaxy**, then select **Configure**.
2. Select **Galaxy**, then select **Alarms**.

   The **Alarms** page appears.
3. Configure alarms:
   a. Edit the priority range for each severity level, or accept the default values.
   b. Accept the default (enabled) historization or click to clear the check box and disable historization for each severity level.
4. Configure events: Accept the default historization values, or click to clear the check box and disable historization for each event type.

If you want to change the default Alarm Border indicator images shown in the **Alarms and Events Configuration** dialog, see "Changing Alarm Border Indicator Icons" in the *Creating and Managing Industrial Graphics User Guide*.

Monitor alarm severities at runtime

Monitor alarms by severity, expressed as integers 1–4, using clients such as Object Viewer and AVEVA OMI ViewApps. You can monitor individual alarms and you can monitor alarms aggregated by containment level. For more information about aggregating alarm states, see Aggregated alarm state information .

Typically, alarm priorities and priority to severity mapping are not changed during runtime, but it is possible to change alarm priority configuration and severity mapping without closing the client application.

Alarm ranking at runtime

The most important alarm has the lowest severity number, but other criteria are taken into account when ranking alarms by urgency at run time.

Alarm Mode	Enabled is more urgent than silenced.
Alarm Shelved	FALSE (not shelved) is more urgent than TRUE (shelved).
InAlarm	TRUE (InAlarm) is more urgent than FALSE (normal).

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Acked	FALSE (unacked) is more urgent than TRUE (acked).
Severity Level	1 is most urgent; 4 is least urgent.

Aggregated alarm information

Alarm aggregation lets you see all the active alarms within an area object, as well as all the active alarms on all the area's contained objects, including contained area objects. This provides an efficient way at runtime to identify if there are any active alarms (in the InAlarm state), and the location and status of alarms, for example, unacknowledged and unacknowledged - return to normal. This makes it possible to follow a trail from one level to the next to find the underlying cause of a complex object's alarms.

To obtain aggregated alarm severity status information:

1. Map alarm severity levels to priority ranges. For more information, see Configure priority ranges for alarm historization, mapping, and shelving .
2. Configure alarms on objects. For more information, see Configure alarms.
3. Enable alarm aggregation by Area. For more information, see Configure alarm state aggregation
4. View aggregated alarm status information by means of runtime clients and applications. For more information, see Monitor alarm state information at runtime.

Configure alarm state aggregation

Configuring alarm state aggregation consists of normal alarm configuration procedures plus an added step of enabling the aggregation feature on each relevant Area object.

Note: Alarm aggregation is enabled by default on Area objects. Alarm aggregation cannot be disabled on application objects, such User Defined objects.

To configure alarm state aggregation

1. Accept the default settings or configure alarm severities for each of the four severity levels you want to aggregate. This is a global configuration.
2. Configure an object with one or more alarms. All alarms configured on all objects will be aggregated to the Area object that contains the objects.
3. Set the AlarmModeCmd of the object to enabled or silenced, but not to disabled.
4. Set the AlarmModeCmd of at least one alarm on the object to enabled or silenced, but not to disabled.
5. Set the AlarmModeCmd of the Area object to enabled or silenced, but not to disabled.
6. Check the **Aggregate Alarms** checkbox on the Area object editor to enable alarm aggregation. This sets the value of the AlarmAggregationStateCmd attribute to True.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.



Aggregated alarm state information

Runtime clients and applications, such as Object Viewer, AVEVA OMI ViewApps, and InTouch HMI applications, can show alarm aggregation statuses.

You can use animations such as the alarm border animation with Situational Awareness Library graphics or with Industrial Graphics in both Managed InTouch applications and AVEVA OMI ViewApps, and you can add alarm aggregation displays to navigation items in AVEVA OMI ViewApps

Alarm aggregation functionality can be described for an object, for an area, and for an attribute.

Object	Aggregation represents all alarms on the object, on all contained objects, and on their descendents down to the lowest level of the model view. Alarm aggregation values on child objects are added to the values of the parent object or objects. All objects have this alarm aggregation functionality.
Area	Aggregation represents the alarms on the Area object itself, on all objects assigned to the area, and on all sub-Areas, down to the lowest level of the model view. If the Area's AlarmMode is silenced, all alarms on all Objects in that Area will be silenced. For more information about setting alarm modes, see the following topics:

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

	• Alarm definition • Set alarm state with object attributes • Enable, silence, and disable alarms.
Attribute	Aggregation represents all the alarms on the attribute itself. This is the lowest level of aggregation.

Alarms are aggregated if they are in one of three states:

- UNACK_ALM
- ACK_ALM
- UNACK_RTN

Alarms in the ACK_RTN state are not aggregated.

Alarms in silenced mode are aggregated, even though they do not appear in a runtime client.

A set of five attributes provide runtime aggregated alarm status information:

AlarmMostUrgentSeverity	Displays the severity as an integer 1–4 of the most important current alarms on an object and its descendents. If no alarms are in the InAlarm state or waiting to be acknowledged, the value is 0.
AlarmMostUrgentMode	Displays the mode (enabled or silenced) of the most important current alarm. Alarms in disabled mode are not aggregated.
AlarmMostUrgentInAlarm	Displays the InAlarm status (true or false) of the most important current alarm.
AlarmMostUrgentAcked	Displays the acknowledgement status (true or false) of the most important current alarm.
AlarmCntsBySeverity	In AVEVA OMI and InTouch HMI runtime client applications, this attribute is used to show the number of alarms by status and severity. In the Object Viewer, this attribute displays an array that shows: • The total number of active alarms (UNACK_ALM + UNACK_RTN + ACK_ALM) at each severity level. • The UNACK_ALM alarm count. • The UNACK_RTN alarm count. • A decimal representation if any active alarm bits are set on the local object.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Note: By default, **AlarmCntsBySeverity** includes shelved and silenced alarms, but does not include hidden or disabled alarms. You can change this by changing the **AggregatedAlarmStatesIncluded** filter. For more information, see AggregatedAlarmStatesIncluded attribute

AlarmCntsBySeverity attribute

AlarmCntsBySeverity is a 13 element array, divided into three groups of four, plus one additional element to indicate local alarm severity and status. The first group of four elements shows the number of aggregated active alarms by severity (S1 through S4), the next group shows the aggregated number of unacknowledged alarms (UNACK_ALM) by severity, and the third group shows the aggregated number of return to normal (UNACK_RTN) alarms by severity. The final element of the array is a bit value that defines which, if any, of the alarms are on the selected (local) object. See Local alarm display in the Object Viewer for additional information.

You can determine the number of active, acknowledged alarms (ACK_ALM) by subtracting the number of unacknowledged alarms shown in the second group of elements from the active alarm count shown in the first group of elements.

Note: By default, **AlarmCntsBySeverity** includes shelved and silenced alarms, but does not include hidden or disabled alarms. You can change this by changing the **AggregatedAlarmStatesIncluded** filter. For more information, see AggregatedAlarmStatesIncluded attribute

AggregatedAlarmStatesIncluded attribute

AggregatedAlarmStatesIncluded is a four-element array that controls whether shelved, hidden, silenced, and/or disabled alarms are included in the values of the **AlarmCntsBySeverity** attribute. It is a galaxy-wide attribute. It is the same for all engines, areas, and objects.

The elements of **AggregatedAlarmStatesIncluded** are:

Element	Alarm condition	Default
1	Shelved	True
2	Silenced	True
3	Disabled	False
4	Hidden	False

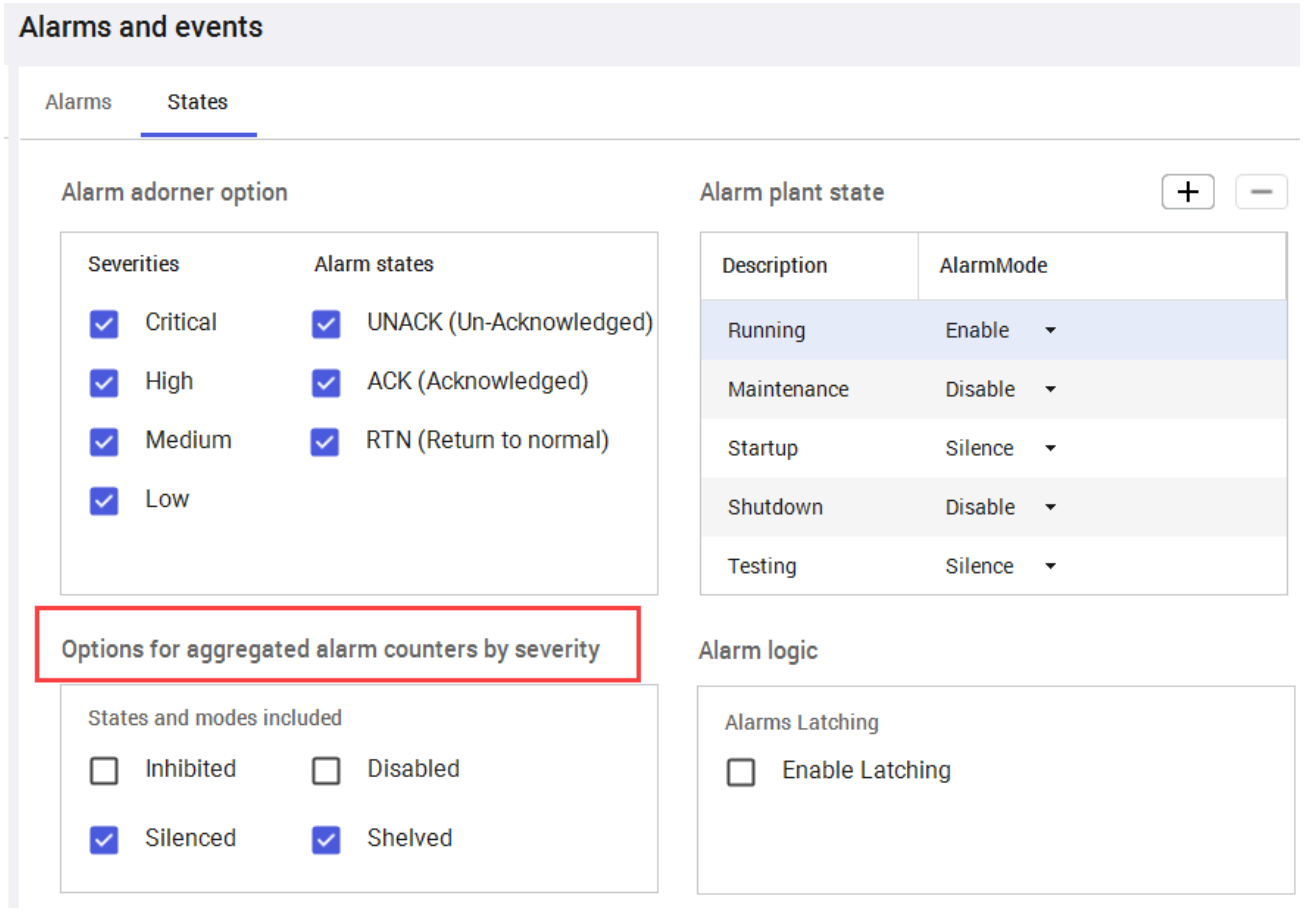
You can change the **AggregatedAlarmStatesIncluded** settings using the IDE. Follow the procedure below.

To configure the alarm states included in AlarmCntsBySeverity:

1. From the IDE main menu, select **Galaxy**, then **Configure > Galaxy - Alarms and events**. This opens the **Alarms and events** dialog.
2. Select the **States** tab.
3. In the **Options for aggregated alarms counters by severity** section, select the checkboxes for the alarm

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

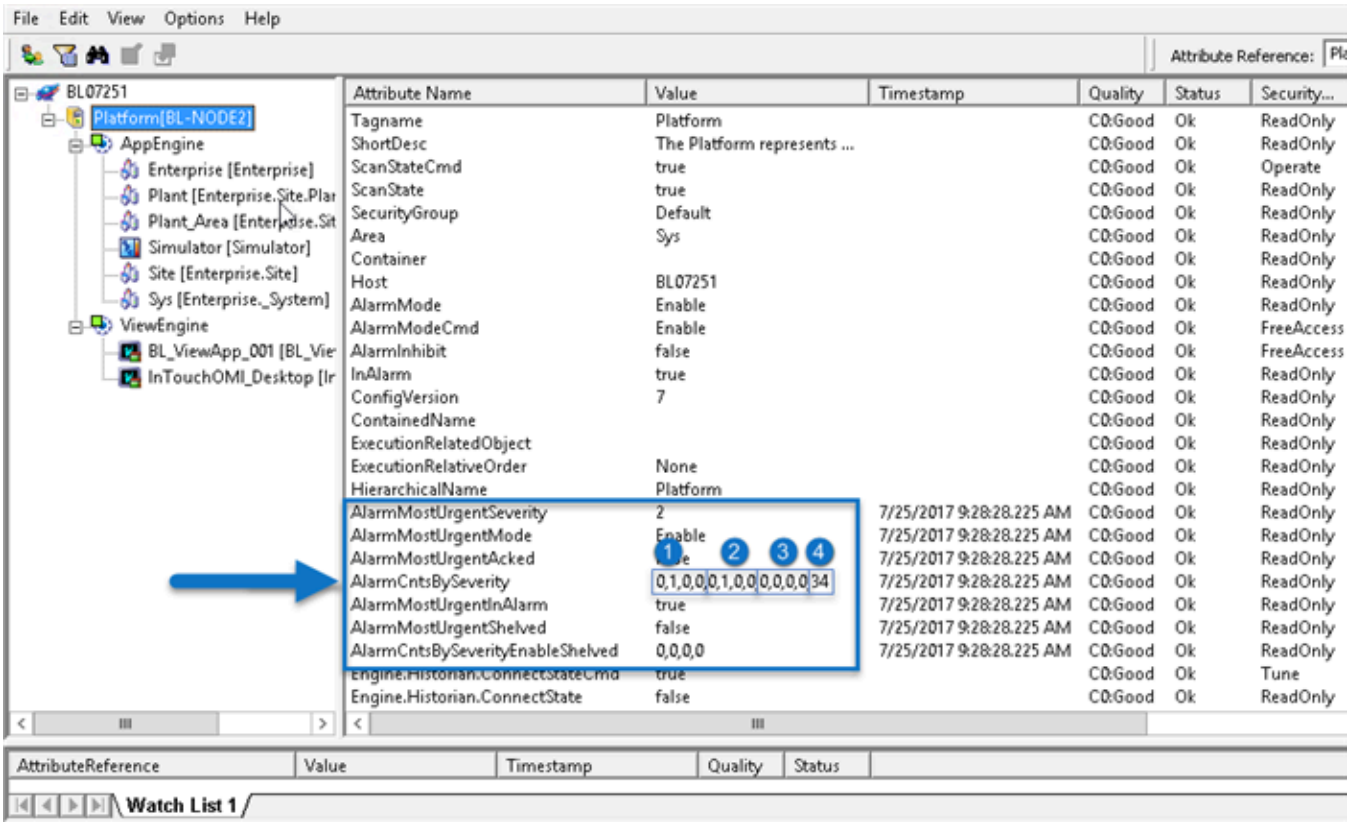
states to include in the **AlarmCntsBySeverity** totals and clear the checkboxes for those not to include.



Local alarm display in the Object Viewer

The Object Viewer shows thirteen values, separated by commas, for the **AlarmCntsBySeverity** attribute. The first twelve values (logically divided into groups of four) show how many alarms of each severity and type have been aggregated for the selected object or area:

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.



1. Number of active alarms, by severity.
2. Number of unacknowledged active alarms, by severity.
3. Number of unacknowledged alarms that have returned to normal, by severity.
4. Bit value that shows which alarm severity and status applies to the local object.

   This value is a sum of the bit values shown in the following table:

	ß (LSB) Active (1)				UnAck_ALM (2)				UnAck_RTN (3) (MSB) à			
	S1	S2	S3	S4	S1	S2	S3	S4	S1	S2	S3	S4
Bit value	1	2	4	8	16	32	64	128	256	512	1024	2048

In the Object Viewer example shown above, the local object bit value (4) is 34. This is the sum of the active S2 bit value plus the UnAck_ALM S2 bit value. Other examples:

- If the local object has one or more active S1 alarms that are unacknowledged, the bit value is 17 (S1 Active Alarm = 1; S1 UnAck_ALM = 16).
- If the local object has one or more active S4 alarms that are UnAck_RTN, the bit value is 2056 (S4 Active Alarm = 8; S4 UnAck_RTN = 2048).
- If there are no local alarms, the bit value is 0.
- If there are multiple local alarms with different severities and unack statuses, the bit value is the sum of each bit that is set. For example, if there are two unacknowledged S1 alarms, but one has returned to normal, the bit value is 273 (two S1 Active Alarms = 1; one S1 UnAck_ALM = 16; one S1 UnAck_RTN = 256). Note that

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

each bit that is set is only counted once, even if more than one alarm exists at that severity or status.

When alarm aggregation is shown on a navigation item in an AVEVA OMI ViewApp, the bit value is used by the navigation item to determine the alarm location (selected object or a contained object). See Aggregated alarm display in OMI ViewApp navigation for additional information.

Note: By default, **AlarmCntsBySeverity** includes shelved and silenced alarms, but does not include hidden or disabled alarms. You can change this by changing the **AggregatedAlarmStatesIncluded** filter. For more information, see AggregatedAlarmStatesIncluded attribute

Aggregated alarm display in OMI ViewApp navigation

The **Alarms and events** dialog is used to configure which aggregated alarms are displayed on a Galaxy-wide basis. You can set whether or not display alarm information in individual AVEVA OMI ViewApps by enabling/disabling the **ShowAlarms** attribute. AVEVA OMI ViewApps display alarm indicators and alarm counts for aggregated alarms on NavigationApp controls (NavigationTree and BreadcrumbControl), if the **ShowAlarms** attribute is enabled in the ViewApp. To enable this attribute, select the **Show Alarms** option when you configure the control in the **Layout Editor**.

Note: Alarm indicators ("adorners") used in AVEVA OMI navigation are FrameworkElement Adorners, part of the Microsoft .NET Framework used in creating and configuring apps for AVEVA OMI ViewApps.

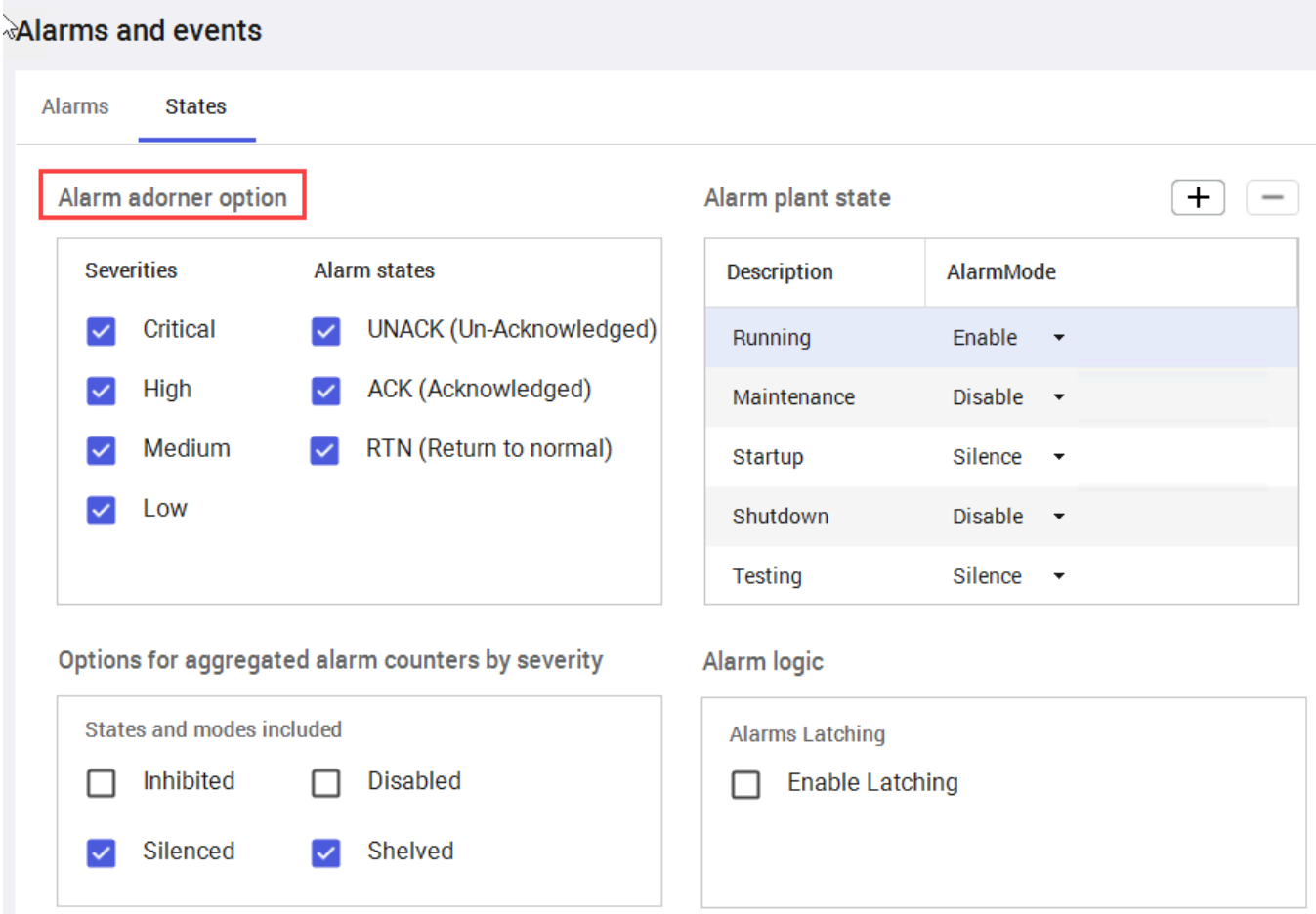
The opacity of the alarm indicator shows if the alarm is on the selected navigation node or on a contained object. When enabled, the following alarm severities and alarm statuses are shown:

- If the color of the alarm indicator is opaque (solid), the alarm is on the selected (local) object. If enabled, the severity (S1 to S4) and status of the alarm (UNACK_ALM, UNACK_RTN, ACK_ALM) is shown graphically, along with a count for each alarm severity that has been enabled.
- If the color of the alarm indicator has some transparency, the alarm is on a contained object.
- The color of the blinking line under the object name shows the highest-severity (most urgent) alarm.
- An indicator on the NavigationTree or BreadCrumbControl shows if the data quality of the object is bad. Bad quality may result if communication has been lost with the object.

You can also determine which navigation node or asset contains the alarm by selecting the alarm icon. This displays a tooltip that indicates the node(s) or asset(s) sending the alarm(s).

Set the **Alarm adorner option** in the **Alarms and events** dialog (**Galaxy** > **Configure** > **Galaxy - Alarms and events**, then select the **States** tab). This configures which aggregated alarms are displayed on NavigationApp controls and any custom apps that subscribe to the aggregated alarm count during runtime.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.



The following **Alarm Adorner Options** are set through the dialog. The corresponding attribute settings are propagated to runtime nodes (all attribute names are prefixed by MyViewApp.Alarms). For additional information about these attributes, see Alarm adorner attributes.

Alarm severities to display	Attribute name (alarms namespace)	Description
Critical	ShowCriticalAlarms	When enabled, the ViewApp displays critical alarms
High	ShowHighAlarms	When enabled, the ViewApp displays high alarms
Medium	ShowMediumAlarms	When enabled, the ViewApp displays medium alarms
Low	ShowLowAlarms	When enabled, the ViewApp displays low alarms

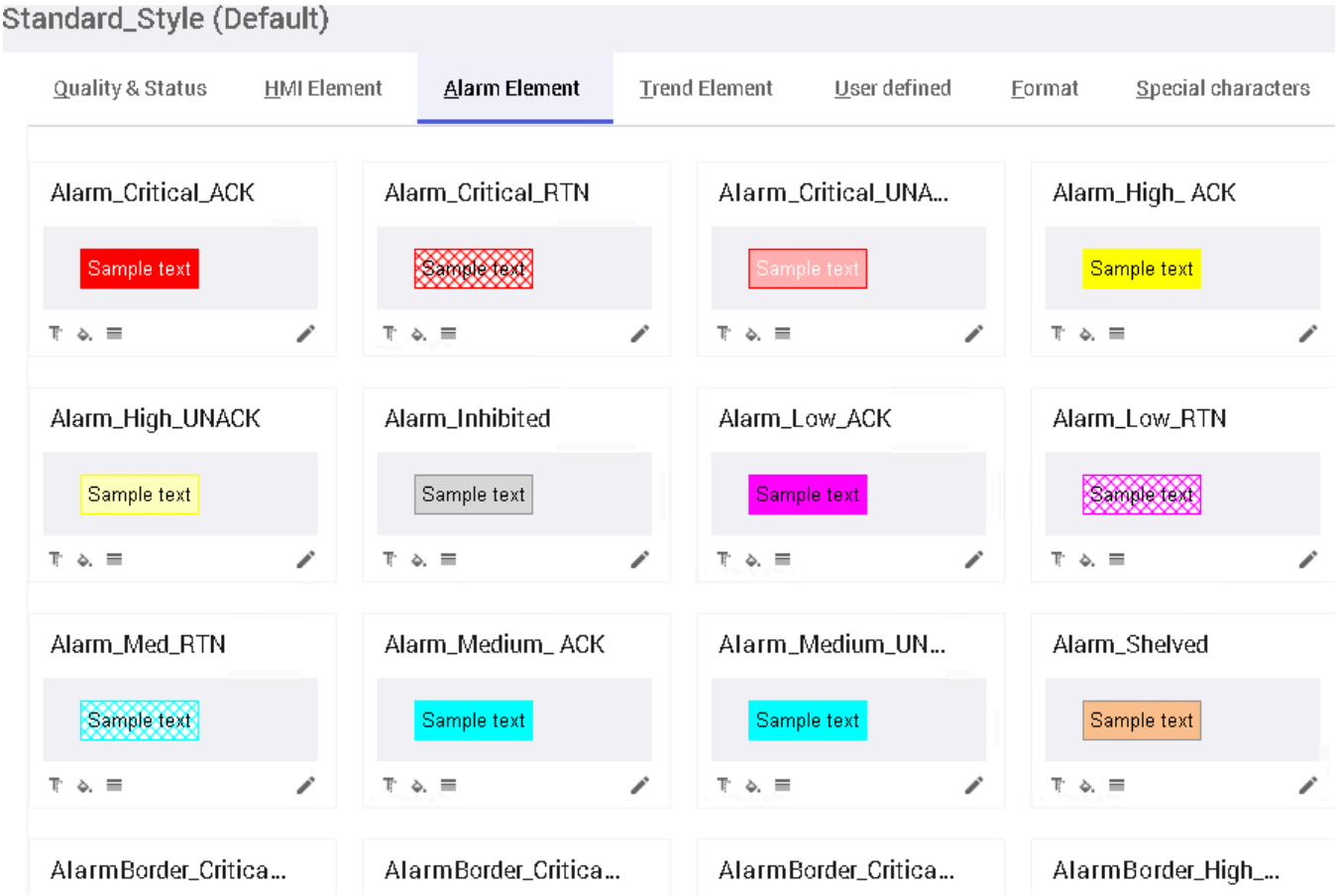
Alarm states to display	Attribute name (alarms namespace)	Description
UNACK (Unacknowledged)	ShowUnacknowledgedAlarm	When enabled, the ViewApp

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

		displays UNACK alarms
ACK (Acknowledged)	ShowAcknowledgedAlarms	When enabled, the ViewApp displays ACK alarms
RTN (Return to Normal)	ShowReturnToNormalAlarms	When enabled, the ViewApp displays RTN alarms
LATCHED	ShowLatchedAlarms	Latched alarm state must be enabled for the Galaxy for Latched alarms to show. When enabled, the ViewApp displays Latched alarms.

Alarm indicator appearance

The appearance of alarm indicators is configured through the **Alarm Element** tab of the **Galaxy Styles** page.



Alarm adorner attributes

Alarm Adorner attributes are available only for use in AVEVA OMI ViewApps. All Alarm Adorner attributes operate in the MyViewApp.Alarms namespace. Attributes for the Alarm Adorner are specified with the MyViewApp.Alarms prefix in the form MyViewApp.Alarms.*attribute_name*. Alarm Adorner attributes cannot be

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

used in an InTouch application because the MyViewApp.Alarms prefix is regarded as a configuration error.

Attribute name	Data type	Attribute type	Default value	Description
ShowCriticalAlarms	Boolean	Read/ Write	True	If True, the ViewApp displays critical alarms
ShowHighAlarms	Boolean	Read/ Write	True	If True, the ViewApp displays high alarms
ShowMediumAlarms	Boolean	Read/ Write	True	If True, the ViewApp displays medium alarms
ShowLowAlarms	Boolean	Read/ Write	True	If True, the ViewApp displays low alarms
ShowUnackedAlarms	Boolean	Read/ Write	True	If True, the ViewApp displays UNACK alarms
ShowAckedAlarms	Boolean	Read/ Write	True	If True, the ViewApp displays ACK alarms
ShowReturnToNormal Alarms	Boolean	Read/ Write	True	If True, the ViewApp displays RTN alarms

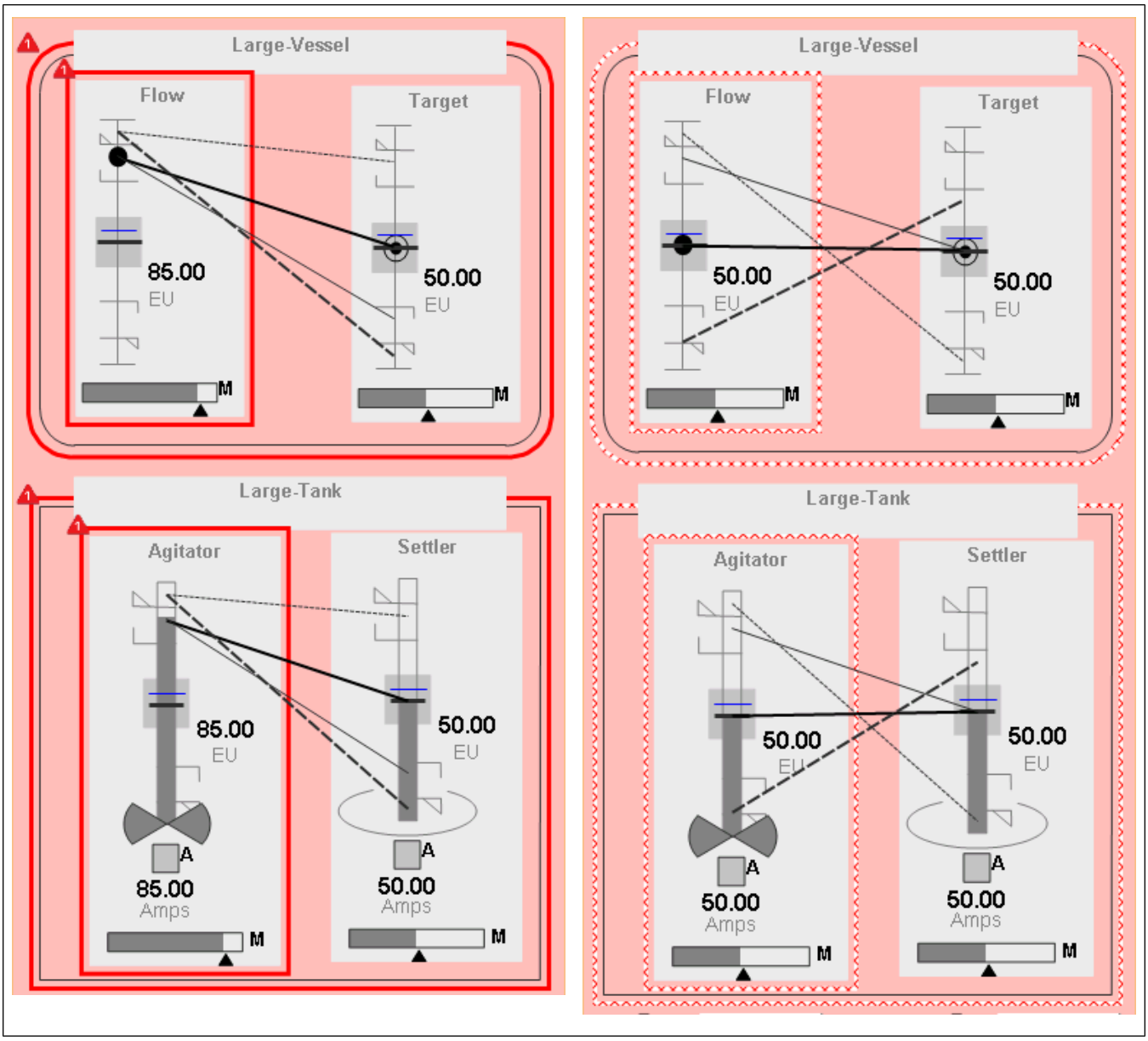
Monitor alarm state information at runtime

You can obtain runtime alarm state information, including aggregated alarm information, using the Object Viewer or InTouch Tag Viewer. You can visualize the objects in an AVEVA OMI ViewApp or InTouch HMI application using the AlarmApp for AVEVA OMI or Alarm Client for InTouch HMI, along with the alarm border animation to represent the aggregated alarms at each level of containment. You can also enable aggregated alarm information for the NavTree

Note: AlarmApp is not supported in InTouch HMI applications. Similarly, AlarmClient is not supported in AVEVA OMI ViewApps. Use AlarmApp for AVEVA OMI ViewApps to allow alarm aggregation to follow selected objects and areas.

An alarm border animation applied to a group of graphics can indicate a priority 1 alarm with a red border, flashing or solid for different alarms. The alarm border animation can also indicate a return to normal (RTN) for the same graphics.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.



Alarm clients configured in InTouch applications will display the severity of each alarm in the **User1** field. You can change "User1" to a more descriptive heading, such as "Severity".

For information about configuring alarm animations, see "Configuring an Alarm Border Animation" in the *Creating and Managing Industrial Graphics User Guide*.

Object wizards

Object Wizards consist of a series of user-selectable choices and options that are used to customize a deployable instance. Each choice and option may have one or more attributes, graphics, other content types and/or scripts associated with it.

An Object Wizard can be added to any derived template, and provides a simplified user interface for configuring instances (assets) from the template. To access the Object Wizard editor:

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

1. Open a template in the **Object Editor.**
2. If the template does not open to the **Attributes** page by default, select it.
3. If necessary, select the **Configure Wizard Options** icon (1) to open the Object Wizard editor. The **Attributes** tab divides into multiple panes that are used for building the Object Wizard.



Object Wizards can be used either within the IDE, or from the **Configure New Asset** editor that also lets you configure an associated graphic as you configure the object.

Note: The **Configure New Asset** editor can only be used if 1) the template contains a graphic, and 2) your security setting allows you to edit graphics. It cannot be used with other content types, such as layouts and external content.

A single template with an Object Wizard can replace a number of derived templates for configuring a variety of similar instances. You can add an Object Wizard to any derived template.

Depending on their level of permissions, users can:

- Create and configure instances from within the IDE, and add them to the Galaxy, or modify existing instances. A user simply opens an instance and answers a series of prompts or questions. See Configure instances for additional information.
- Use the **Configure New Asset** editor to create and configure instances and a representative graphic, simply by dragging an associated (linked or embedded) graphic into another graphic, and then answering a series of prompts or questions. See Create a new instance graphically for additional information.

  Note: The drag and drop method is only valid for graphics, and not for other linked content types such as layouts and external content.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

- For more information about linking to a graphic, see Link to shared content in the visualization folder.
- For more information about embedding a graphic, see Add a new object-owned symbol.

Important! InTouch HMI does not support directly adding assets to an InTouch HMI window if the asset is derived from an object wizard with a linked Symbol Wizard, such as the graphics in the Situational Awareness Library. **With InTouch HMI, the Symbol Wizard must be contained in an overview graphic** when deriving assets from an object wizard. In contrast, AVEVA OMI does support the the use of linked Symbol Wizards when adding assets to AVEVA OMI layouts from an object wizard.

Object Wizard features

An Object Wizard contains choices and options that guide users through the process of configuring instances. You can replace many derived templates by using a single template with an Object Wizard, since the Object Wizard can contain a wide variety of configuration possibilities. Attributes and features contained in the Object Wizard that are not needed in a particular instance are removed and are not a part of the runtime object. Finally, and perhaps most importantly, an Object Wizard is easy to use and can reduce user training and system maintenance costs.

Object Wizards provide the following benefits:

- Simplified workflow for configuring instances
- Automatic propagation of changes, without having to lock templates
- Unnecessary attributes can be trimmed from instances to reduce the footprint of deployed objects
- Unused graphics can be hidden from users to reduce clutter
- Reduced system maintenance
- Reduced training requirements for plant personnel

Create a basic Object Wizard

An Object Wizard can include Choice Groups, each with two or more Choices, and/or Options. A Choice Group is a radio button list that lets the user select one choice. An Option contains a checkbox that a user can select or clear.

To add an Object Wizard to a template

1. In the IDE, open a derived template in the Object Editor.
2. If the **Attributes** page is not already selected, select it now. See Create derived templates.

   Note: Only templates derived from the $UserDefined base template open in the Object Editor with the **Attributes** page selected. The Object Editor always opens the first page of the object. For example, the Object Editor opens the **General** page of the $AnalogDevice template.
3. Configure the template by adding:
   - Attributes
   - Graphics or other content such as links to documents and videos
   - Scripts

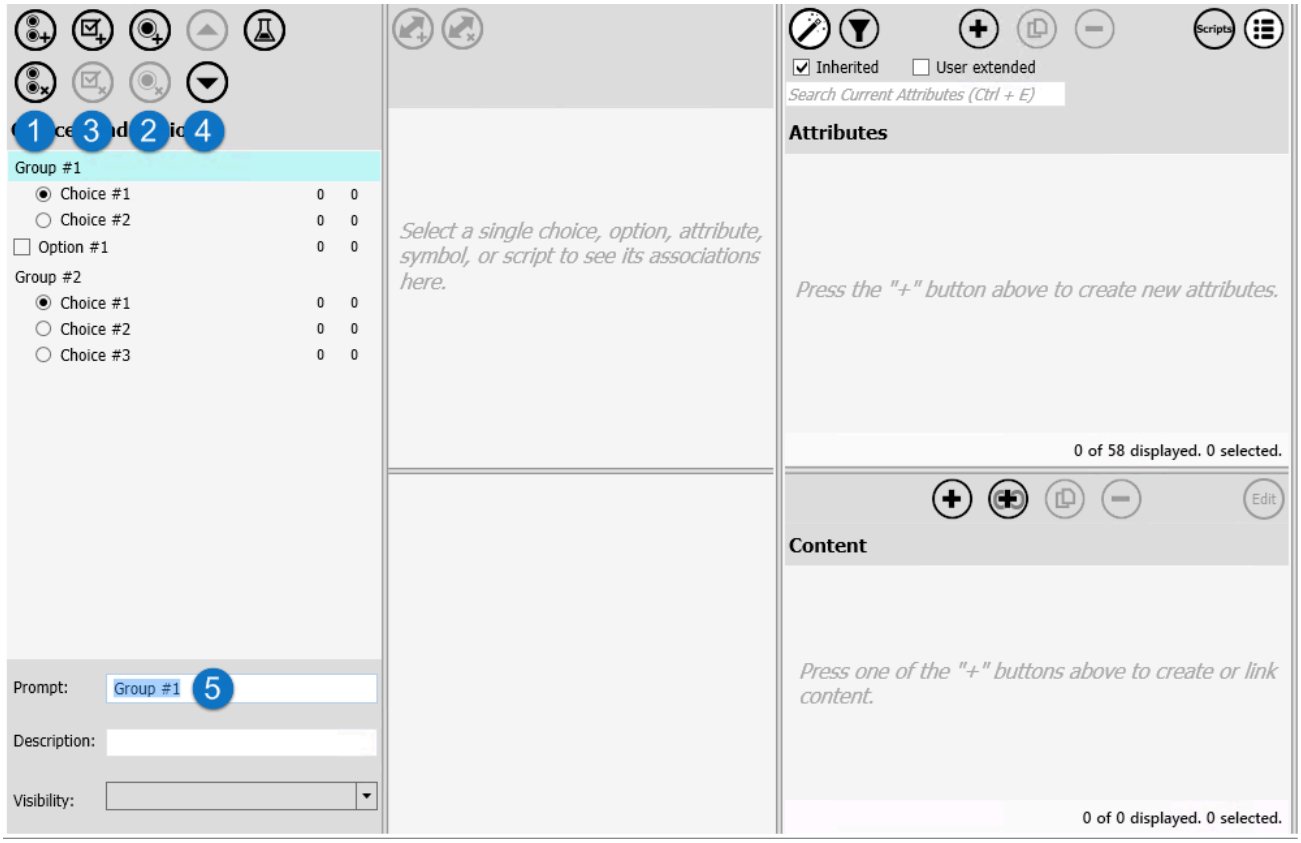
   See Add content for use with an object wizard for additional information.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

4. If the Object Wizard editor is not already open, select the **Configure** W**izard Options** button.

   See Object wizard GUI panes for additional information.

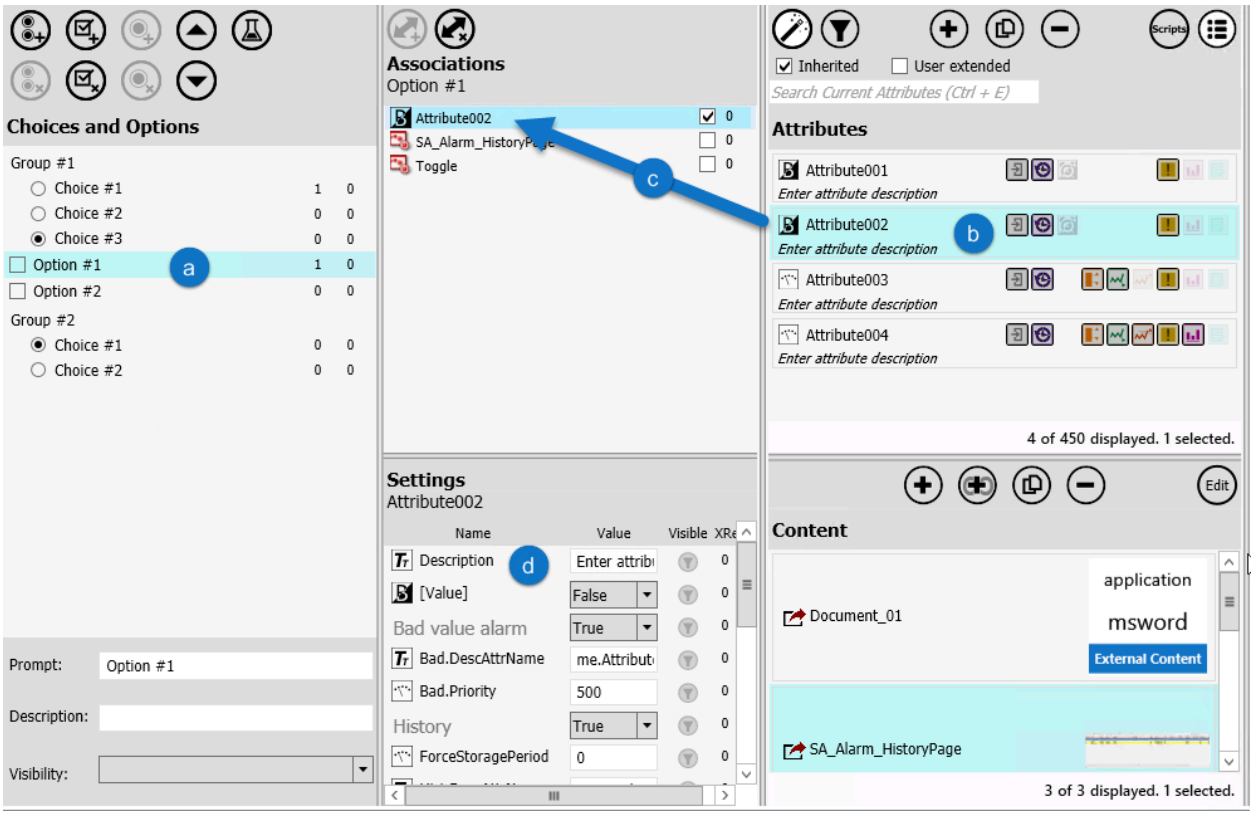
5. Add a Choice Group by selecting the **Add Choice Group (1)** icon. With the Group name selected, change the name of the Group shown in the **Prompt (5)** text field. See Add a Choice Group for more information.



6. To add more Choices to a Choice Group, select the Choice Group and then select the **Add Choice (2)** icon. To change the name of a Choice, select the Choice and enter the new name in the **Prompt** field. See Add a Choice to a Choice Group for more information.
7. Add Options by selecting the **Add Option (3)** icon. Change Option names the same way that you change Choice Group and Choice names. See Add an Option for more information.
8. Use the **Up and Down arrow (4)** icons to change the order of Choice Groups, Choices (within a Choice Group), or Options.
9. To associate an attribute or content with a Choice or Option:
   a. Select the Choice or Option.
   b. Select the attribute or content (graphic or other type of content) you want to associate with the Choice or Option.
   c. Select the **Add Association** icon (or drag the attribute or content to the **Associations** pane).
   d. Settings for the selected attribute or content are shown in the **Settings** pane, once the association is made. Configure visibility and value overrides, as needed.

   Note: No settings are shown for graphics that do not contain custom properties or a Symbol Wizard.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.



Object wizard configuration best practices

An Object Wizard can be added to any derived template. However, to maximize the benefits of using Object Wizards, add your Object Wizard to the template as close to the top level of the derivation hierarchy as possible. To build an Object Wizard you must:

- Add and configure attributes, graphics, links to external content, and scripts before you build the Object Wizard.
- An Object Wizard requires at least one Choice Group or Option. See Object wizard components for additional information.
- Configure Choices and Options by associating attributes,graphics/content, and scripts with Choices and Options. We recommend that you limit the number of items you associate with each Choice or Option to avoid configuration mismatches.
- When configuring Choices and Options, keep the workflow as simple and direct as possible. Remember that users will be working sequentially through the Object Wizard. If you enable a feature at one level of the Wizard hierarchy, you cannot disable the same feature at a subsequent level.

  For example, if a graphic has Custom Property "X" enabled and is associated with a Choice, do not associate the same graphic with "X" disabled to an Option further down in the hierarchy. Instead, leave "X" disabled and override the setting when deriving an instance.
- Configure the Conditional visibility expressions for choice groups, choices, and options of Choices and Options in derived instances.
- Configure settings for the associated attributes and graphics. Scripts and external content do not have settings that can be configured.

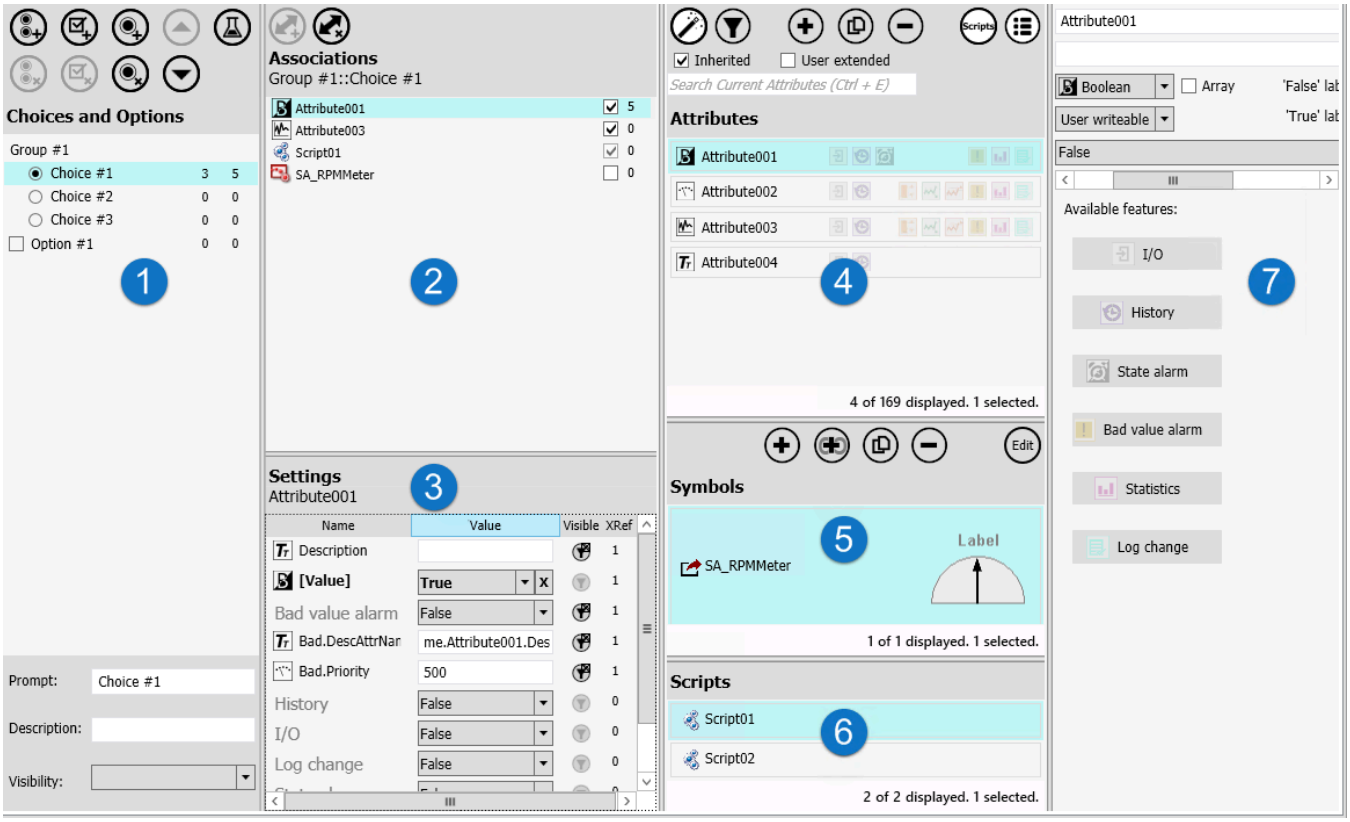
© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

- Select whether each attribute, graphic, and other content can be Object wizard trimming from derived instances. Scripts are always configured as trimmable.

Object wizard GUI panes

The user interface for configuring Object Wizards is accessed through the **Attributes** page within the Object Editor of the Integrated Development Environment (IDE).

	Select the **Configure Wizard Options** icon to open the Object Wizard user interface. The **Attributes** page divides into multiple panes for use in creating and editing an Object Wizard. See Object wizard user interface details for additional information.
	Select the **Scripts** icon (located at the top right of the **Attributes** pane) to open the **Scripts** pane.



Ref	Name	Description
1	**Choices and Options pane**	Add, rename, reorder, and delete Choice Groups, Choices, and Options that comprise the Object Wizard in the **Choices and Options pane (1)**. You can also set the logic for displaying Choices and Options.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

2	**Associations pane**	Drag and drop attributes, content, and scripts to the **Associations pane (2)** to associate them with a choice or option; you can use the **Add** button at the top of the pane instead of dragging and dropping. You can select multiple attributes, graphics, or scripts at once. You will also enable and disable trimming of associated attributes and graphics in this pane. See Object wizard trimming for additional information. You can view associations for individual attributes, graphics, and scripts in this pane by selecting an attribute, graphic, or script. Every association of the item is displayed. Select an association to view the item's configuration.
3	**Settings pane**	Configure attribute and graphic overrides in the **Settings pane (3)**. Overrides are set individually for each choice or option with which the attribute or graphic is associated. Default settings for attributes are set in the **Details pane (7)**. You can resize columns by selecting a column header (in this case, **Value**) and moving its edges to widen or narrow it.
4	**Attributes pane**	The **Attributes pane (4)** lists the attributes contained in the template. You can create new attributes in this area. To associate attributes with the Object Wizard, highlight a Choice or Option, and then select and drag one or more user-created attributes from the **Attributes pane** to the **Associations pane (2)**. See Associate Attributes with Choices and Options for additional information. Default configuration settings for a selected attribute are displayed to

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

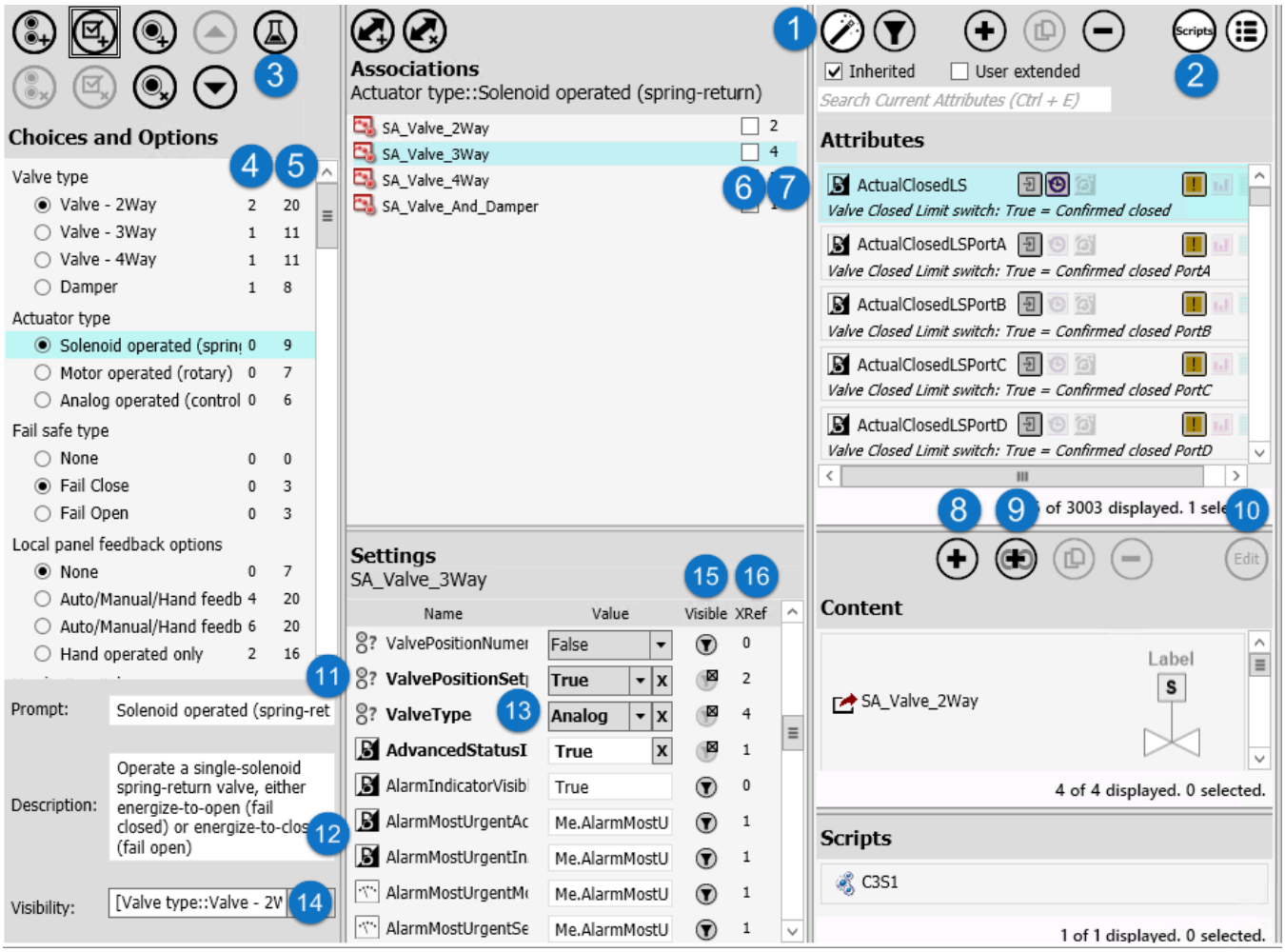
		the right of this pane, in the **Details pane (7)**. See Attributes page for additional information. Configured settings for the selected attribute that apply to a selected Object Wizard Choice or Option are shown in the **Settings pane (3)**. You can also enable attribute features (historization, I/O, alarms, etc. in the **Attributes pane**. Multi-select can be used to enable features for attributes, as well as to associate attributes. To use multi-select for features, select the attributes in the **Attributes pane** and enable the feature in the **Details pane**.
5	**Content pane**	The **Content (5)** pane lists graphics and external content items contained in, or linked to, the object. To associate graphics with the Object Wizard, highlight a Choice or Option, and then select and drag one or more items from the **Content pane** to the **Associations pane (2)**. You can add a new graphic to the object from this pane, or link to a graphic in the Visualization folder. See Associate content with a choice or option for additional information.
6	**Scripts pane**	To display the **Scripts pane (6)**, select the **Scripts** button. The **Scripts pane** lists scripts contained in the object. To associate a script with the Object Wizard, highlight a Choice or Option, and then select and drag one or more scripts from the **Scripts pane** to the **Associations pane (2)**. See Associate content with a choice or option for additional information. To add a script to the object, select the **Scripts tab** at the top left of the **Object Editor**.
7	**Details pane (Attribute Editor)**	The **Details pane (7)** displays details of a selected attribute,

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

		graphic, or other content. This pane lets you configure individual attributes, but is not usually used while configuring an Object Wizard. See Configure objects for additional information.

Object wizard user interface details

The Object Wizard user interface provides a number of visual elements to help you as you build an Object Wizard.



Ref	Icon	Name	Description
1		**Configure Wizard Options button**	Opens/closes the **Object Wizard** editor (**Choices and Options**, **Associations**, **Settings**, **Attributes**, and **Content**

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

			panes.
2		**Scripts button**	Opens/closes the **Scripts pane** so you can associate scripts with Object Wizard Choices and Options.
3		**Test mode button**	Changes the **Object Wizard** editor to test mode. This lets you verify the behavior of the Object Wizard. See Object wizard test mode for additional information.
4	<#>	**Number of associations**	Number of trimming-enabled attributes, graphics/content, and scripts associated with a Choice or Option. This can help in the maintenance of complex Object Wizards by letting you quickly identify where trimming has been enabled.
5	<#>	**Total configured settings**	Total number of configured settings for all attributes and graphics that are associated with each **Choice or Option**. This can help in the maintenance of complex Object Wizards by letting you quickly identify where overrides have been configured.
6	☐ ☑	**Trimming checkbox**	Place a check in the checkbox to enable trimming for an attribute or graphic. Trimming is always enable for scripts, and is enabled by default for attributes. See Object wizard trimming for additional information.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

7	<#>	**Number of configured settings**	Number of configured settings for each **attribute** or **graphic** that is associated with the selected Object Wizard Choice or Option. Scripts and external content items do not have configurable settings.
8		**Add graphic button**	Add (create new) graphic button. The new graphic is owned by the object. See Add content for use with an object wizard for additional information.
9		**Link content button**	Link to an existing graphic or external content item in the Visualization folder. The graphic/content can be shared with other objects. See Add content for use with an object wizard for additional information.
10		**Edit graphic button**	Opens the selected graphic in the **Graphic Editor**.
11		**Symbol Wizard icon**	Symbol Wizard settings are indicated by a unique icon. See **Attribute and Graphic Icons**, below.
12	 	**Attribute / Custom Property icons**	Attribute and graphic custom property icons are displayed for Boolean, integer, float, double, string, time, elapsed time, and internationalized string. An icon is also displayed for Symbol Wizard settings. See **Attribute and Graphic Icons**, below.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

13		**Setting Override indicator**	Attribute, custom property, and Symbol Wizard settings that contain an override are indicated by **bold** text and an X next to the value. To remove the override, select and clear the X. See **Override Icons**, below.
14	<text field>	**Visibility Expression**	Sets the conditional visibility of Object Wizard elements. See Conditional visibility expressions for choice groups, choices, and options for additional information.
15	**Visibility indicator icons**		When the icon is unmuted, the setting is visible to the user when configuring instances. When muted, the setting is hidden. See Attribute and graphic overrides and Configurable settings for attribute and graphic values for additional information. A small x in the top right corner of the icon indicates that the visibility setting is an override, and may differ from the setting in the parent template. **Note:** In derived templates, visibility indicators replace visibility expressions for Object Wizard choices and options. See Object wizard derivation for additional information.
		Visible	
		Visible - override	
		Hidden	
		Hidden - override	
16		**XRef**	Total number of configured overrides for the selected graphic or attribute. If the attribute or graphic is associated

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

			with multiple Object Wizard Choices or Options, hover over the XRef number to see which Choices and Options have overrides configured.

Attribute and Graphic Icons

Icon	Description / Data Type	Attribute	Graphic
	Symbol Wizard Choice or Option		X
	Boolean data type	X	X
	Integer data type	X	X
	Float data type	X	X
	Double data type	X	X
	String data type	X	X
	Time data type	X	X
	Elapsed time data type	X	X
	Internationalized string data type	X	X
[No icon]	Large type with no icon indicates that the item is an attribute feature. Attribute features are selected in the **Details pane** (see Object wizard GUI panes) and include I/O, History, State Alarms, Bad Value Alarms, Hi/Lo Alarms, etc.	X	

Note: For custom properties, string, time, and elapsed time data types can be toggled between reference/expression and string literal. This does not apply to attributes.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

	Reference/expression
	String literal

Override Indicators

Indicator	Description
	This is the default setting (no override). If the setting is changed in a parent template, this value will also change.
	Setting has been overridden. Changes made in a parent template will not cause the this value to change. The value is shown in **bold** and an X appears next to the value.
	No override: Setting is visible and is not overridden. If the visibility of this item is changed in a parent template, the visibility of this setting will also change.
	No override: Setting is hidden and is not overridden. If the visibility of this item is changed in a parent template, the visibility of this setting will also change.
	Override has been applied: The visibility setting ("visible") is locked in this template. If this item is hidden in a parent template, this setting will remain visible (setting will not change).
	Override has been applied: The visibility setting ("hidden") is locked in this template. If the this item is made visible in a parent template, this setting will remain hidden (setting will not change).

Object wizard components

An Object Wizard consists of Choice Groups and Options. Choice Groups contain at least two Choices, and Object Wizard users must select one Choice from every Choice Group.

Once you create Choice Groups, Choices and Options, you can associate attributes, graphics, layouts, external content, and scripts with the wizard to create an Object Wizard workflow. Typically, you create and configure the required attributes, scripts and other content prior to adding an Object Wizard to the template. See Configure objects for additional information.

Choices are chosen by selecting the radio button next to the Choice. Only one Choice can be selected from a Choice Group. When users select a Choice, they are also selecting any elements (attributes, etc.) associated with

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

the Choice, even though those elements may not be visible to them.

Options are selected by placing a check in the Option's checkbox. Options can be either true (checked) or false (unchecked). Attributes, graphics, other content, and scripts can only be associated with the true state. No action is implemented when an Option is set to false.

You can rename Choice Groups, Choices, and Options as needed and add a description (the description is visible to users as a tooltip). Choice Groups and Options must be uniquely named; a Choice Group and an Option cannot have the same name. Choices within a single Choice Group must also be uniquely named.

Add content for use with an object wizard

You add attributes, graphics, layouts, scripts, and other content to a template object with an Object Wizard the same way that you add content to any other object. After adding the content, you can associate individual attributes and other content to individual Choices and Options. This association allows content that is not needed to be trimmed when you derive instances from the template. This helps to reduce the footprint of runtime objects. See Object wizard trimming for additional information.

Note: Only user-created attributes can be associated with Object Wizard Choices and Options.

When you associate content with Choices and Options, the propagation of these items to derived instances is determined by:

- User selection of Choices and Options during configuration.
- Trimming configuration for attributes and graphics, as they apply to the selected Choices and Options.

Content is trimmed from a derived instance when the instance is saved. Note that all items are propagated to derived templates, regardless of configuration and trimming settings, since unlike instances, templates are not runtime objects.

Both attributes and graphics can have unique settings for each association. The visibility of these settings, which determines if the setting can be overridden by a user deriving instances from the template, can also be configured.

For detailed information about custom properties and Symbol Wizards, see Using custom properties with Application Server and Symbol Wizard Editor.

Content can either be added or linked to an object.

	When you select the **Add Content** button, a new graphic is created that is owned by the object. Any changes made to the graphic are only applicable to the owning object. Changes to the graphic in the parent object will cascade to child objects. The **Add Content** button does not let you create an object-owned external content item.
	When you select the **Link Content** button, the Galaxy Browser opens the Visualization folder. Selecting a graphic, layout or external content object from the Visualization folder creates a link between the object and the content, but the object does not own the content. Note that relative references will resolve

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

	correctly for the linked graphic or layout. Changes to the content will cascade to every object with which the item is linked, regardless of the hierarchical relationship of the objects.

For information on adding scripts to a template, see Write and edit scripts.

The settings configured in the **Attribute Editor**, **Graphic Editor**, and **Layout Editor** provide the default settings for each attribute and content item. You can override the default values for attributes and graphic custom properties in the **Settings pane**. Note that you can set different values for each Choice or Option with which the content is associated.

In addition, you can allow the user to override the default values in derived instances. See Attributes page and Add attributes to an object for information about adding and configuring attributes.

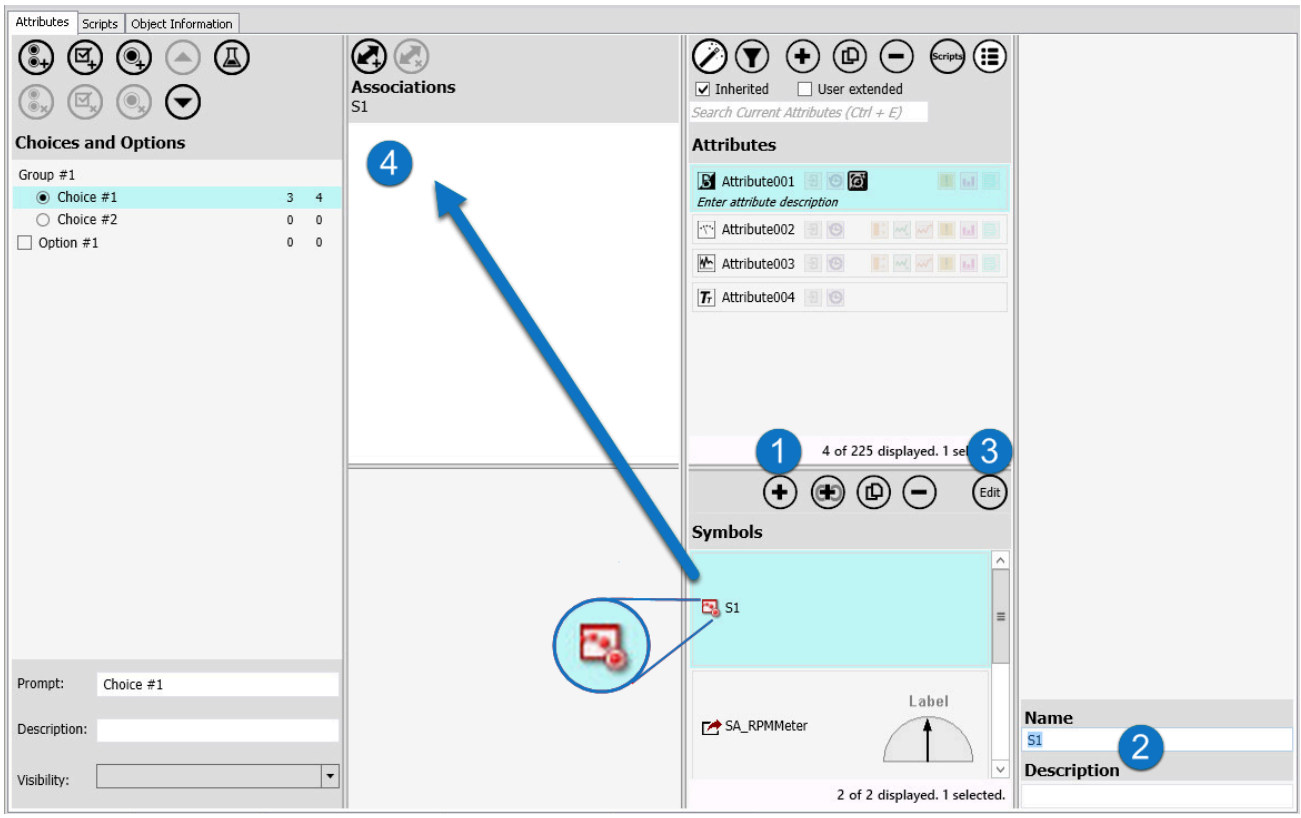
Add a new object-owned symbol

To add a new object-owned graphic

1. In the **Content** pane, select the **Add Content (1)** button. The graphic is added with a default name (based on the previously added graphic, if any).
2. In the **Details (2)** pane, edit the name of the graphic and add a description.
3. In the **Content** pane, select the **Edit (3)** button to open the **Graphic Editor**. See *Creating and Managing Industrial Graphics User Guide* for information about using the editor to create graphics. When you are done editing the graphic, save and close the Graphic Editor.
4. Associate the graphic with one or more Choices or Options, as described in Associate content with a choice or option.

   The icon next to the graphic name in the **Content** pane indicates that the graphic is owned by the object.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.



Link to shared content in the visualization folder

You can link an object to content in the Visualization folder. Content can be a graphic, a layout or an external content object.

To link to an existing content item in the Visualization folder:

1. In the **Content** pane, select the **Link Content** button .

   The **Galaxy Browser** opens. Select an existing graphic, layout, or external content object from the **Visualization folder**.

2. In the **Details** pane, edit the default name of the content as needed. The edited name applies only to the object to which it is linked. The name of the content item remains unchanged in the Visualization folder. You can also add a description .

   Note: If you choose to link to the same content item more than once, it is automatically renamed within the object with an incremental number appended to the original name. Its name in the Visualization folder is not changed, and is displayed in brackets in the following format: `OriginalName1 [OriginalName]`. In most cases, linking to the same item more than once is not an efficient use of an Object Wizard.

3. **Optional:** To edit the linked content, select the **Edit** button to open the **Graphic Editor** (for graphics), the **Layout Editor** (for layouts), or the **External Content Editor** (for external content items)**.** Any change you make to the linked content item changes the configuration of the content, and the change affects any object that links to the item.

   Note: Editing protected graphics, such as graphics contained in the **Situational Awareness Library**, is not permitted. If you edit an item in the **Visualization folder**, the changes will affect every object that links to the

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

item.

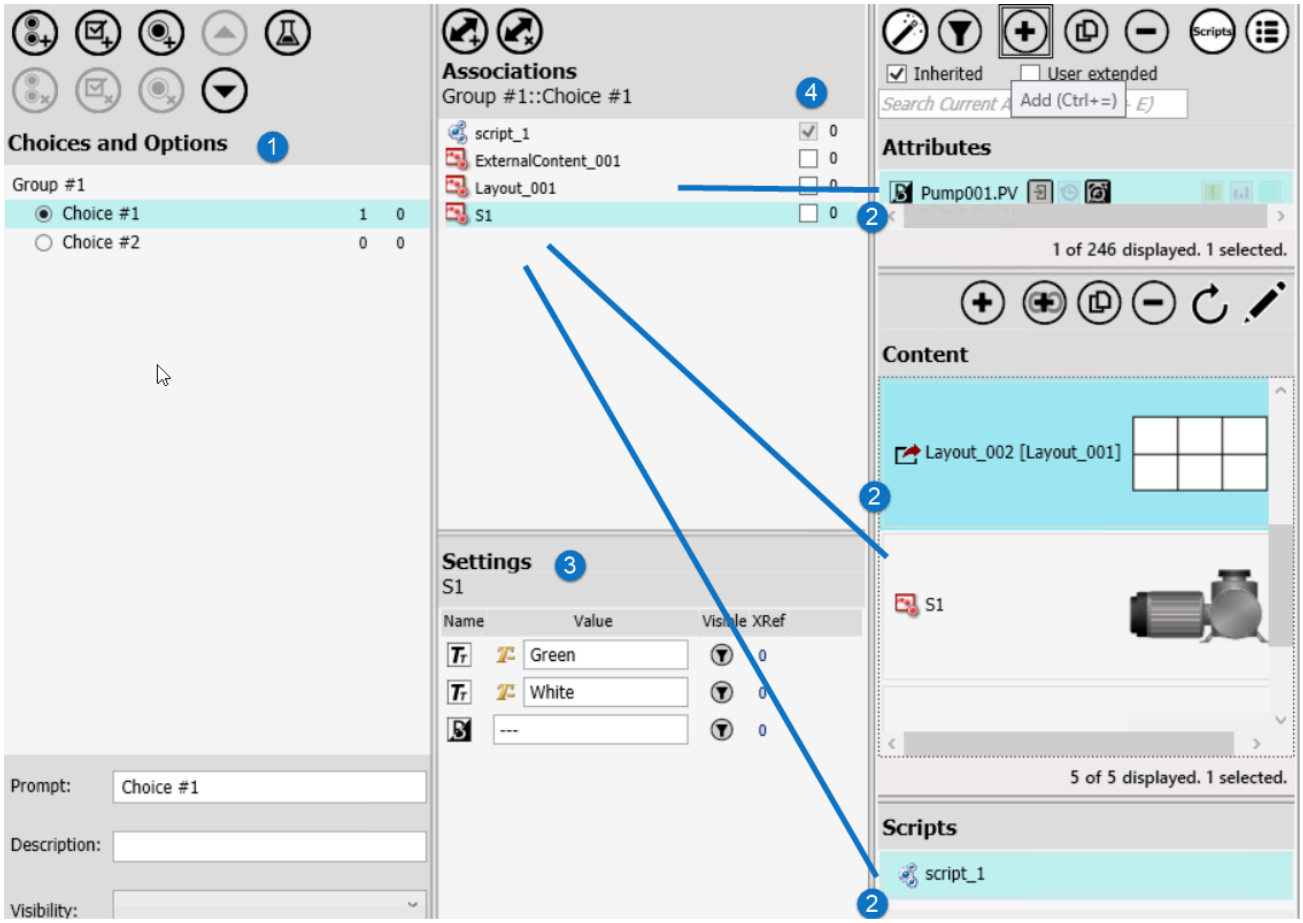
Associate content with a choice or option

In most cases, you will have added content to the object prior to adding the Object Wizard. Once you have begun configuring the Object Wizard, you can associate the following types of content with Object Wizard Choices and Options:

- Attributes: shown in the **Attributes** pane
- Graphics: shown in the **Content** pane
- External Content Objects: shown in the **Content** pane
- .NET Controls: shown in the **Content** pane
- Scripts: shown in the **Scripts** pane

To associate content with a Choice or Option:

1. Highlight a Choice or Option in the **Choices and Options** pane.
2. Select and drag one or more item from the **Attributes**, **Content**, or **Scripts** pane to the **Associations** pane. Use the **Shift** or **Ctrl** key to select multiple items in a single pane.



Note: Individual items can be associated with more than one Choice or Option. However, a Choice or Option does not have to have any items (attributes, etc.) associated with it. For example, a Choice labeled "No" or

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

None" typically would not have any content associated with it.

Note: A content item (attribute, graphic, layout, etc.) does not have to be associated with an Object Wizard Choice or Option. Unassociated items are simply a part of the object and are propagated to all derived objects.

3. If applicable, highlight an attribute or graphic and enter overrides for the default values. The overrides affect only the Choice or Option for which they are entered. For example, a graphic can have one label for one Choice, and a different label for another. The override applies only to the associated Choice or Option. See Attribute and graphic overrides for additional information.

   Note: You can resize columns in the **Settings** pane by selecting the column header and moving its edges to widen or narrow the column.

4. If applicable, override the default trimming setting for each associated item.
   - Trimming is disabled by default for graphics, layouts, and external content (checkbox is unchecked).
   - Trimming is enabled by default for attributes (checkbox is checked).
   - Trimming is enabled for scripts and cannot be disabled.
   - Note that the trimming defaults for graphics and attributes are different. See Object wizard trimming for additional information.

Disassociate content from a choice or option

To remove the association of a content item (attribute, graphic, etc.) with a Choice or Option:

1. Locate the Option, or the Choice within its Choice Group and select it.
2. In the **Associations** pane, select the content item you want to remove from the Choice or Option.
3. Select the **Remove Association** button. The item is no longer associated with the Choice or Option.

   Note: Removing an association does not remove the content item. The content remains a part of the template, and if the content is associated with other Choices or Options, those associations remain intact. However, removing an association may affect the trimming of content within instances (see Object wizard trimming for additional information). If the content is completely unassociated with any Object Wizard Choice or Option, it is always included in derived instances.

   To completely remove an attribute from a template, see Delete content from an object.

Multiple content item associations

Attributes, graphics, and other content can be associated with more than a single Choice and/or Option. Each association can use different override values (settings) for a particular attribute or graphic. To change a setting for a particular association, enter overrides of the defaults in the settings pane.

You can use different overrides for each Choice or Option with which an attribute or graphic is associated. For example, an integer attribute could have a default value of 10, but could have an override of 5 for one Choice and an override of 20 for a different Choice.

Map attribute and graphic configurations to a choice or option

You can associate a particular attribute, layout, or graphic configuration with an Object Wizard selection (Choice

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

or Option). In this way, pre-configured attributes and content can be added to the derived instance simply by selecting an Object Wizard Choice or Option. Different configurations can be applied to different Choices and Options.

Mapping these configurations to Object Wizard Choices and Options may require additional planning and testing. This is especially important if multiple configurations of the attribute or content are being mapped. Mismatches between configurations can result in different behaviors in the derived instance than intended. Use **Test mode** to catch potential mismatches.

Attribute and graphic overrides

Override values for an attribute or graphic (with a Symbol Wizard or custom properties) can be configured for each associated Choice or Option. When you associate an attribute or graphic, the default values are listed in the **Settings** pane. To configure values that apply to the selected Choice or Option, enter the values in the **Settings** pane. If the attribute or graphic is associated with multiple Choices or Options, different values can be used for each one. Values for the Choices and Options are saved when you save the object. If the setting is marked as visible, users can override your setting (or override the default, if you have not configured a setting) as they derive and configure instances from the Object Wizard.

Graphics with overrides that affect their shape, for example, certain Situational Awareness Library graphics, are rendered with the applicable shape-changes as defined by the overrides.

Note: If you override a Symbol Wizard setting, the only way to make the setting visible again is to remove the override. Attributes and custom properties with overrides can be made visible again by selecting the visibility checkbox.

A configured setting is defined as any change from the default configuration of the attribute or graphic. This includes changing a value or visibility. For example, you might change the default Boolean value from true to false, or you might change an integer value from 5 to 10, or change engineering units from liters to cc. Changing the visibility of a setting also counts as a configured setting.

- The default visibility setting for attribute values is hidden (Visibility icon is muted (grayed-out) in the **Settings** pane).
- The default visibility setting for Symbol Wizard and custom property values is visible (Visibility icon is unmuted in the **Settings** pane).
- Visibility overrides are indicated by a small X in the upper right quadrant of the visibility icon.

The total number of configured settings for each individual attribute and graphic is displayed in the **Associations** pane. The total number of configured settings for ALL attributes and graphics associated with an individual Choice or Option is displayed in the **Choices and Options** pane.

Delete content from an object

You can delete (completely remove) a content item from an object. When you delete content, all settings and associations to the items are also removed.

If there are any instances or templates derived from the template that use the deleted item, the deletion of the item will propagate to all the derived objects. That is, all objects derived from the template will no longer contain the deleted content.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Delete an attribute or content from a template

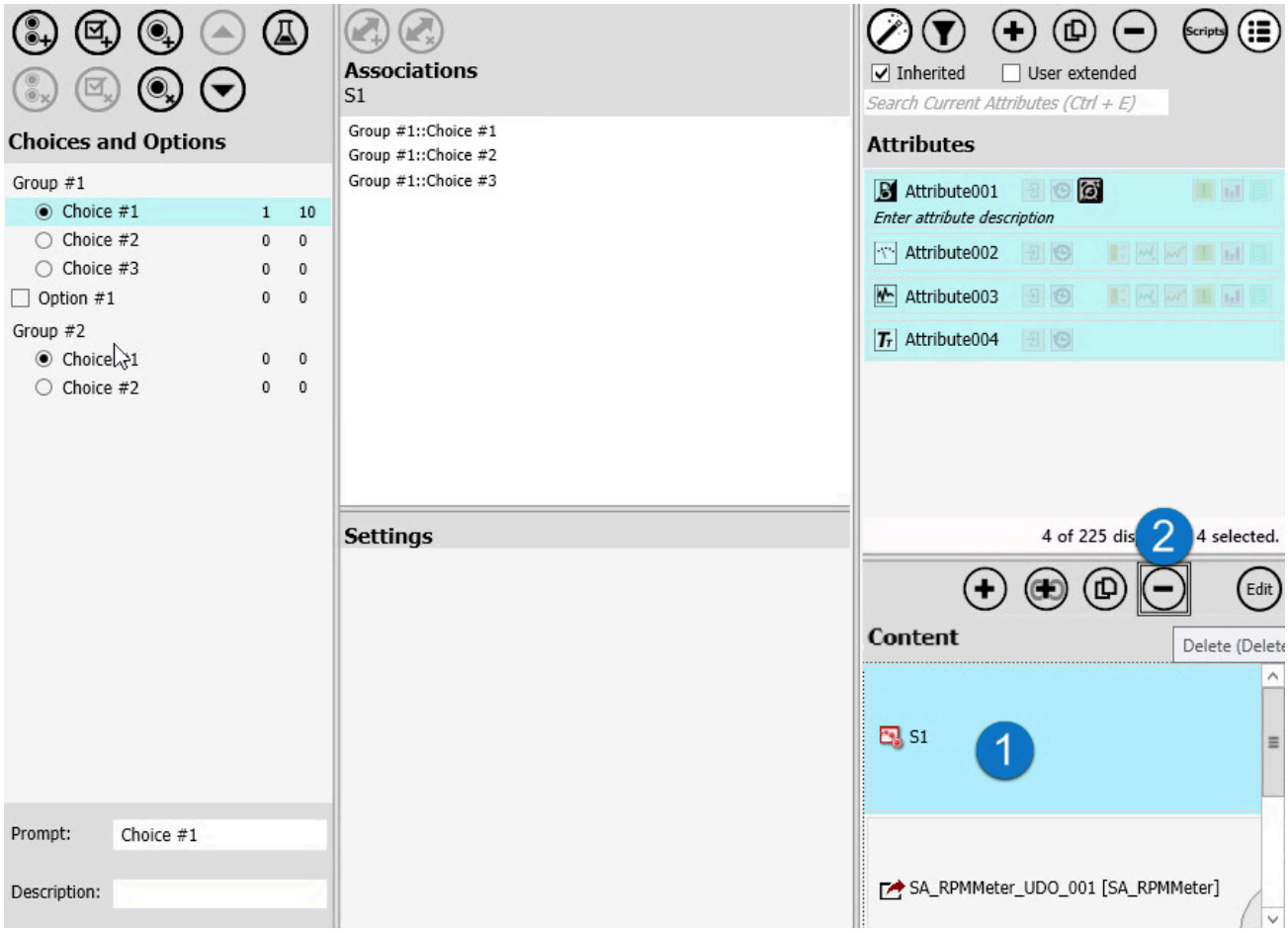
You can delete an attribute or content (object-owned or linked) from a template that has an Object Wizard. If there are objects derived from the Object Wizard that use the deleted attribute or content, the deletion propagates to all the derived objects.

Linked graphics, layouts, and external content items will still exist in the Visualization folder. Deleting an object-owned graphic completely removes it. Once you save the template that contains the deleted object-owned graphic, the graphic cannot be retrieved.

To delete an attribute or content from a template:

1. Select the content item from the **Content** pane. To delete an attribute, select it from the **Attributes** pane. You can select more than one attribute or graphic at a time.
2. Select the **Delete** button. This removes the selected graphic from the object. Use the **Delete** button in the **Attributes** pane to delete attributes.

Note: Added (object-owned) graphics are permanently deleted once the object is saved. Linked content is removed from the object but remains in the Visualization folder.



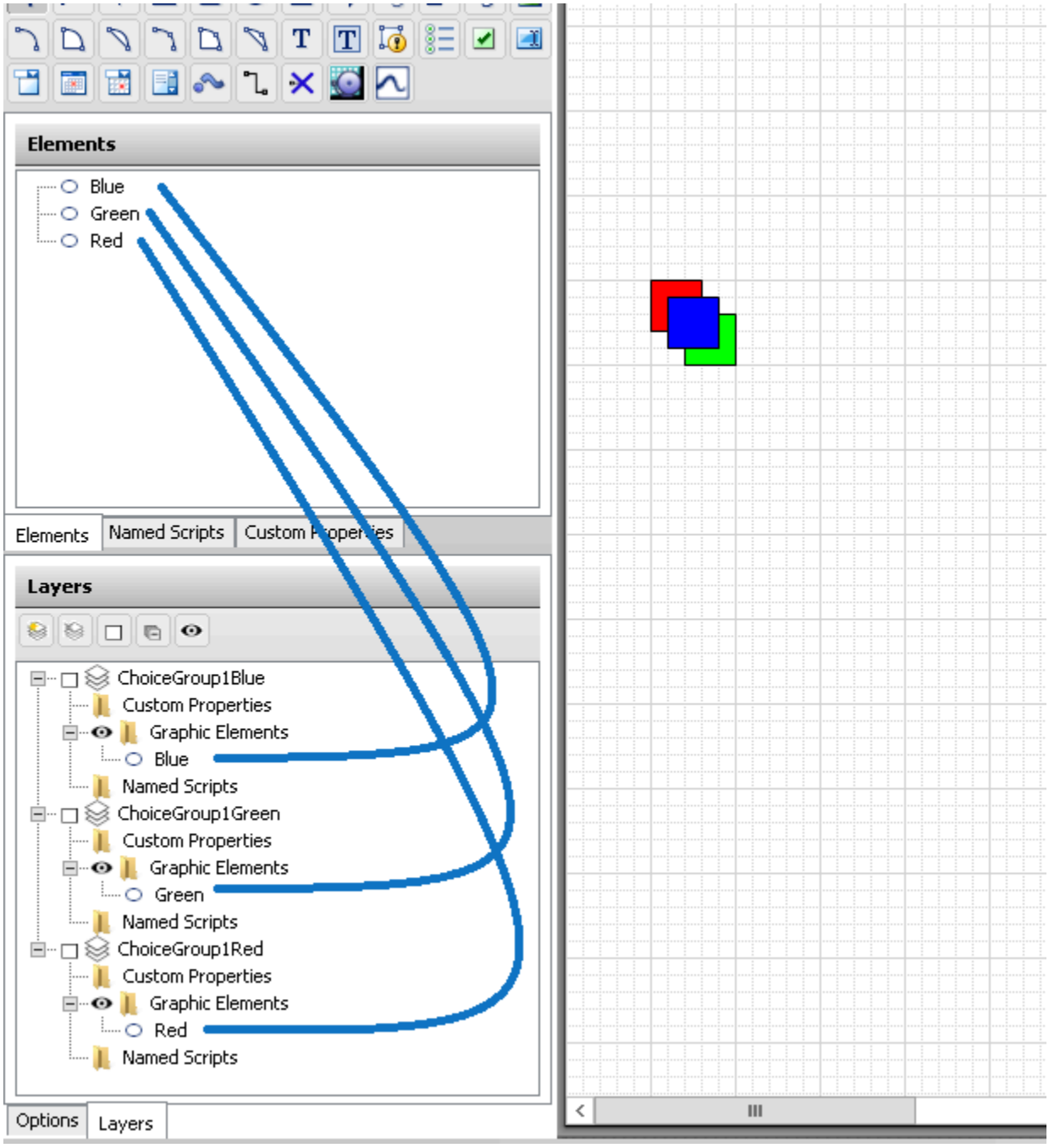
Example1: Map a symbol wizard to an object wizard

This example shows how to create a simple Symbol Wizard, set configurations, and map the configurations to

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Object Wizard Choices.

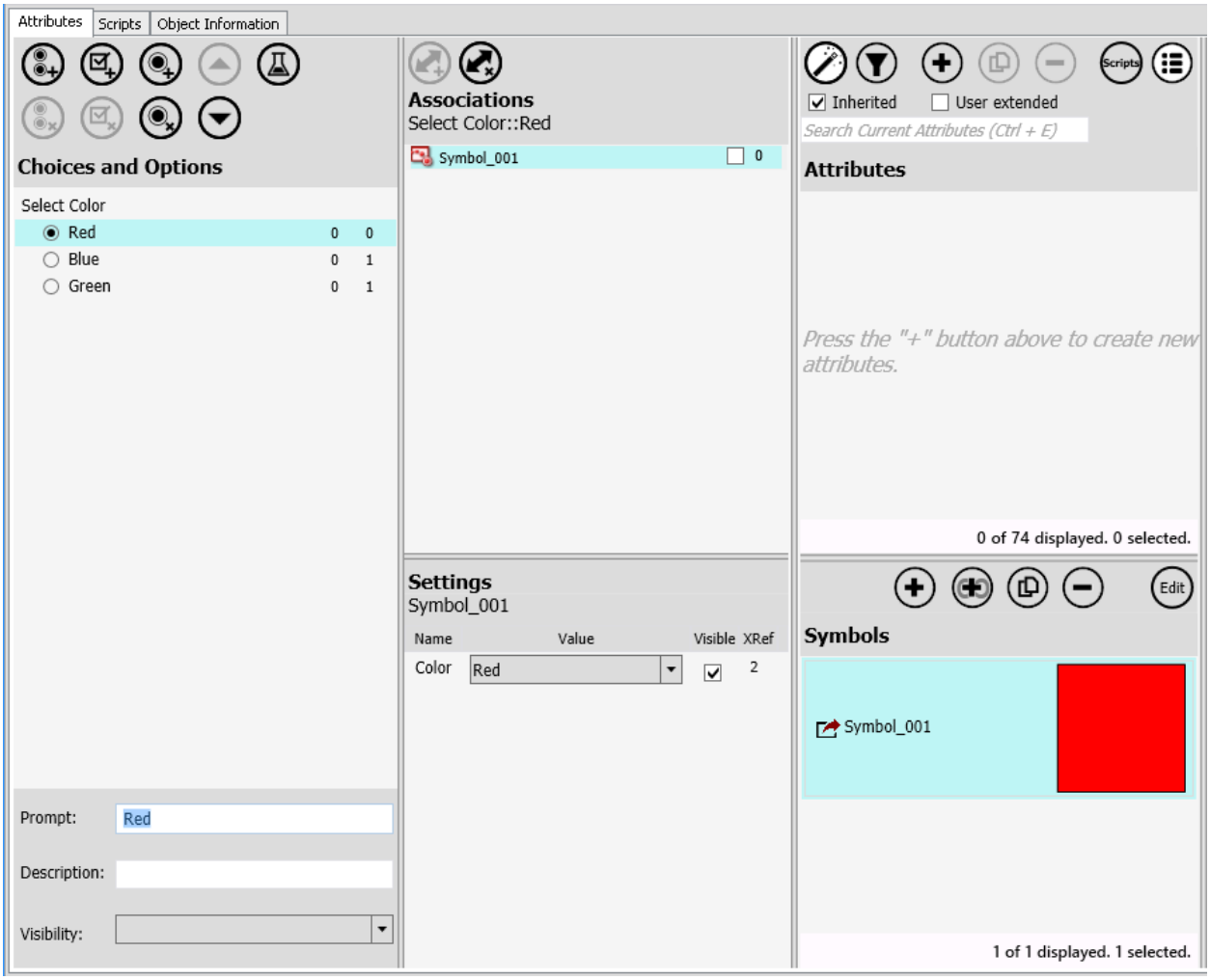
1. Create a new graphic in the Visualization folder.
2. Build a simple Symbol Wizard in the **Graphic Editor**. by adding a Choice Group with three Choices named Red, Green, and Blue.
3. Add three shapes, filled with colors that correspond to the Choices. Rename the shapes so you can identify them.
4. Assign each shape to the corresponding Choice by dragging the shape name from the **Elements tab** to the corresponding Choice in the **Layers tab**.



5. Save and close the **Graphic Editor**.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

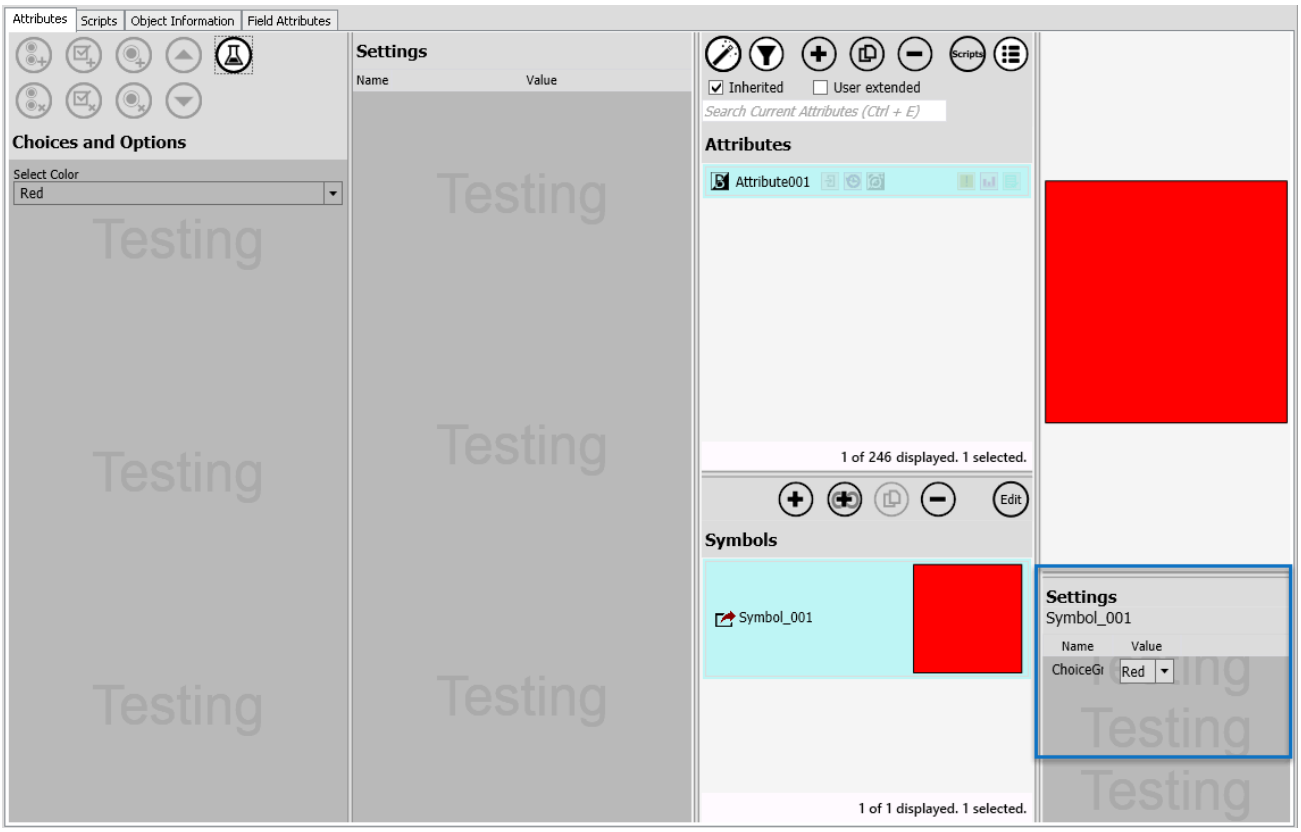
6. Derive a new template from the Template Folder, and add an Object Wizard with:
   - One Choice Group. Change the name of the Choice Group to "Select Color."
   - Three Choices: Red, Green and Blue.
7. Link the Symbol Wizard you created in step 1 to the derived template.
8. Configure each Symbol Wizard association to match the corresponding Object Wizard Choice.
   - Red Symbol Wizard Choice selected for Red Object Wizard Choice.
   - Blue Symbol Wizard Choice selected for Blue Object Wizard Choice.
   - Green Symbol Wizard Choice selected for Green Object Wizard Choice.



Note: Only the default selection can be marked as visible (you can remove the checkmark to hide the default selection). Selections with override are not visible. Users cannot modify selections that are not visible.

9. Switch to Test Mode.
10. With the default Option Wizard Choice selected (Red, in this example), you should see something like this:

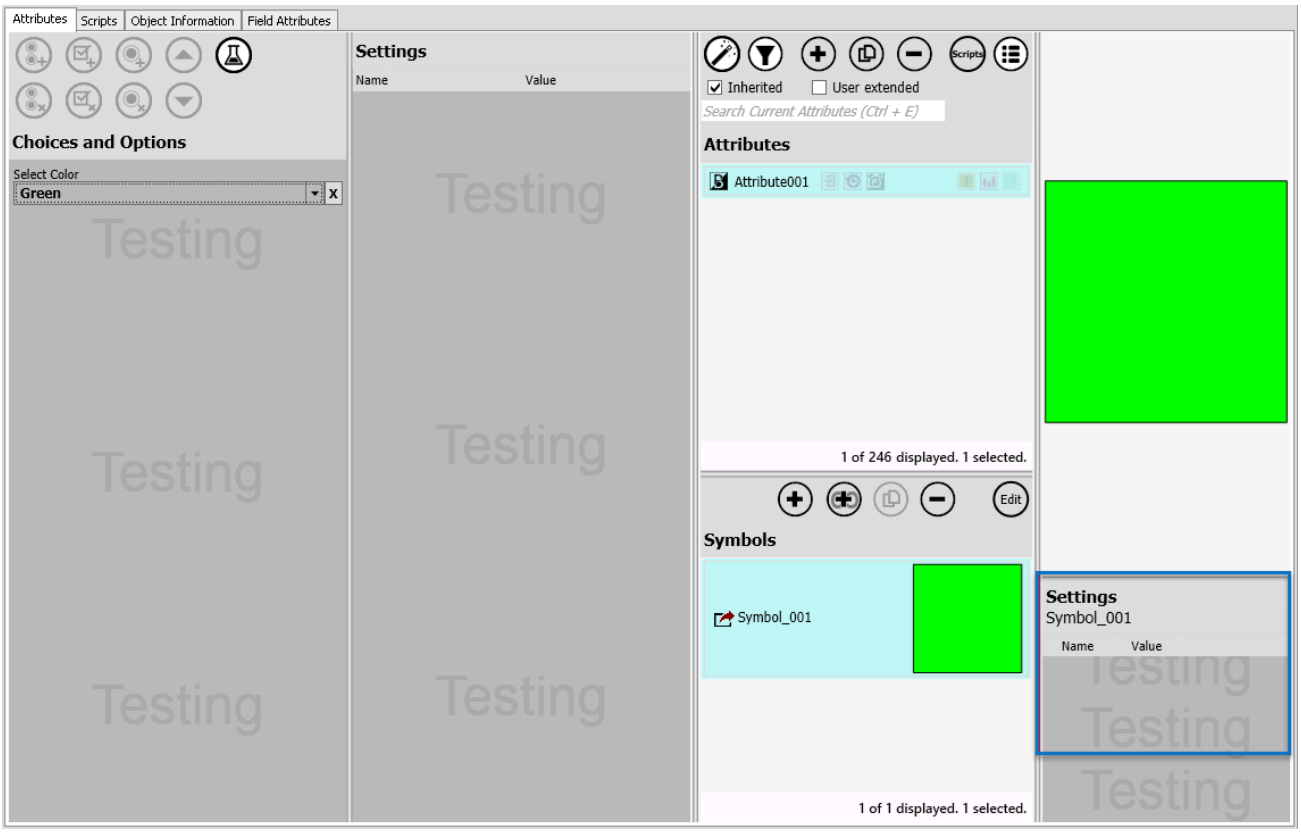
© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.



The Symbol Wizard Choice Group can be changed in the **Symbol Settings** pane, since we did not turn off visibility for the default selection.

11. With any of the non-default Option Wizard Choices selected, you should see something like this:

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.



In this case, Symbol Wizard settings cannot be changed because a non-default configuration of the Symbol Wizard is associated with the Green Object Wizard Choice. When a non-default selection is made in the Symbol Wizard, its visibility setting is turned off and disabled.

Example2: Configure a linked graphic associated with multiple choices

This example shows the steps required to link and configure a graphic from the **Visualization folder**, set configurations, and associate different configurations with Choices and Options. This example uses a graphic from the **Situational Awareness Library**. You could also link other types of content, such as layouts and external content items. However, only graphics allow you to configure and associate different configuration settings within the Object Wizard.

Note: Most graphics in the **Situational Awareness Library** are Symbol Wizards.

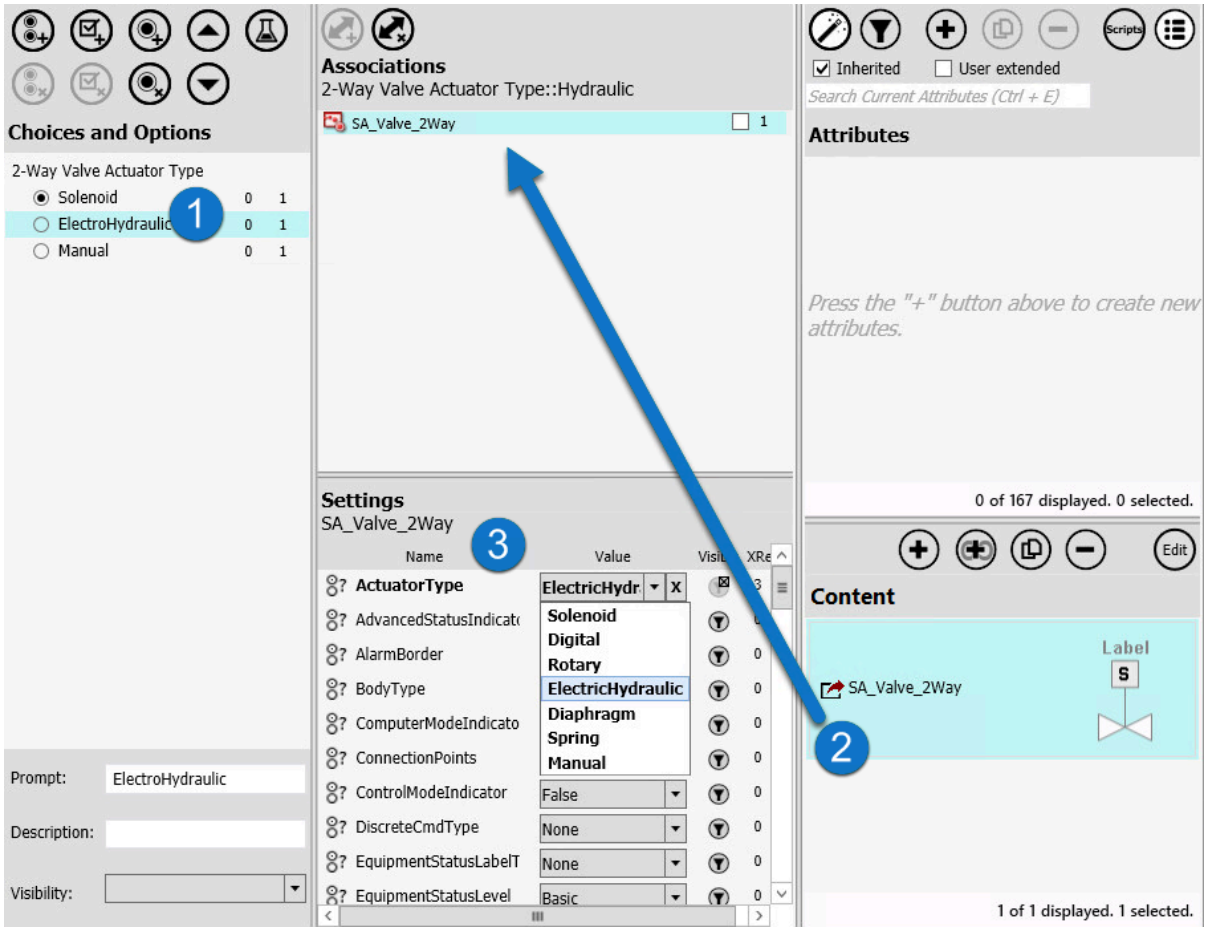
To configure a linked graphic associated with multiple choices

1. Derive a template from the $UserDefined template, then:
   a. Add an Object Wizard Choice Group and name it "2-Way Valve Configuration."
   b. Add three Choices, named: "Solenoid," "ElectroHydraulic," and "Manual."
2. Navigate to the Equipment folder of the Situational Awareness Library (in the **Visualization** folder), and link to the SA_Valve_2Way graphic. The graphic is added to the **Content** pane.

   Associate the linked graphic with each of the Choices created in step 2, and set the ActuatorType in the **Settings** pane to match the associated Object Wizard Choice.
3. Set visibility and overrides for each association:

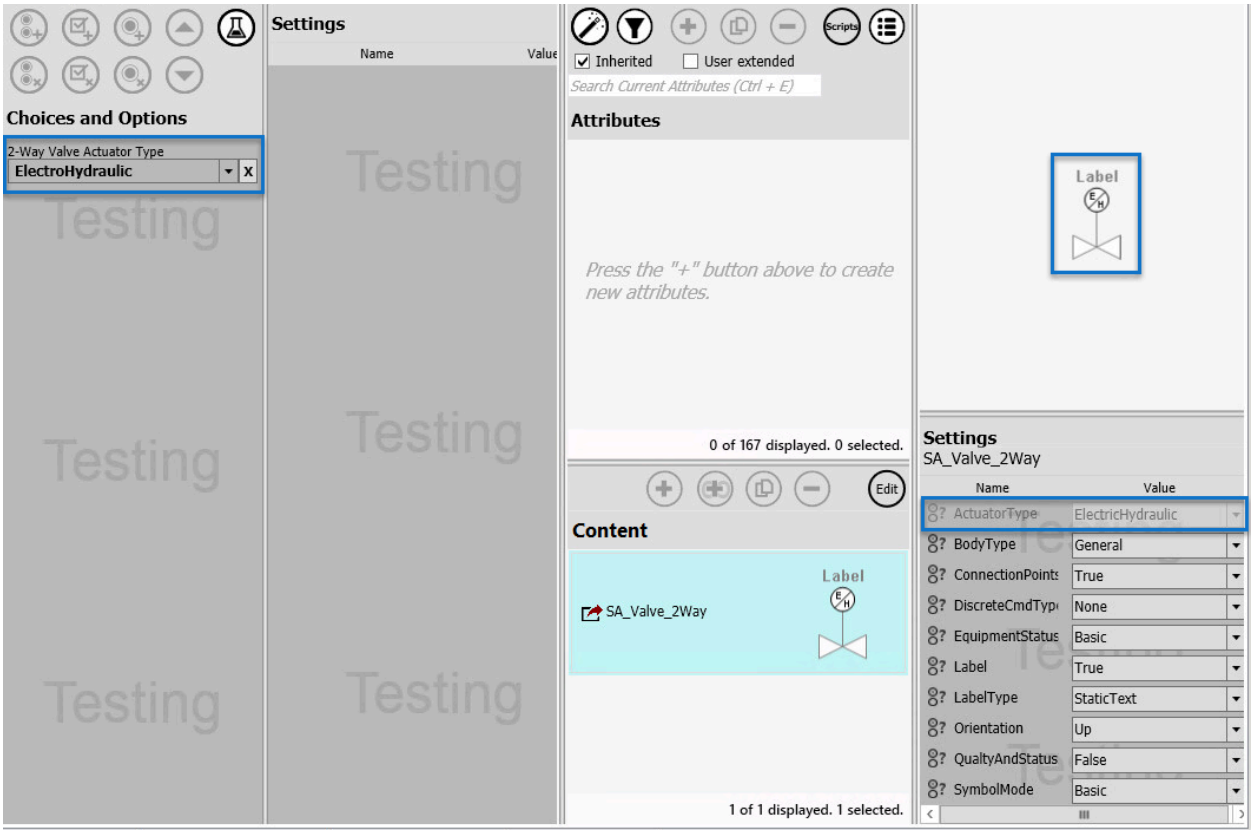
© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

- For Solenoid:
  - Select the Solenoid Choice from the **Choices and Options** pane.
  - Select SA_Valve_2Way from the Associations pane.
  - In the Settings pane, make sure that ActuatorType is set to the default (Solenoid), then turn off visibility.
- For ElectroHydraulic and Manual:
  - Select the ElectroHydraulic or Manual Choice from the **Choices and Options** pane.
  - Select SA_Valve_2Way from the Associations pane.
  - In the **Settings** pane, set the ActuatorType to match the Choice in the **Choices and Options** pane. Since both ElectroHydraulic and Manual are overrides, visibility of these settings automatically changes to hidden.



4. Switch to **Test mode** and check each of the Choices in the **Choices and Options** pane.
   - ActuatorType under graphic **Settings** should be grayed-out to indicate it is hidden
   - The graphic setting should match what you have selected in the **Choices and Options** pane**.**
   - The graphic thumbnail should match the selected ActuatorType.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.



Conditional visibility expressions for choice groups, choices, and options

You can set whether or not a user will be able to see individual Choice Groups, Choices and Options as they use the Object Wizard.Visibility expressions are scripted and use simple logical expressions to determine if users will be able to see a particular Option or Choice when configuring instances. The visibility expression determines if the Object Wizard component will be visible or hidden, based on the user's previous Object Wizard selections. For example, selecting Choice #1 may hide (or show) Choice Groups, Choice or Options that follow that Choice in the Object Wizard.

Note: You cannot add a visibility expression to the first listed item (Choice Group or Option) in an Object Wizard; the first item is always visible.

When a user uses the Object Wizard to configure an instance, the default behavior of any elements hidden by visibility expressions remains in effect. For example, if Choice #1 is the default of a Choice Group, and the Choice Group is hidden, Choice #1 is still selected for the configured instance, even though it is not visible to the user. If the default state of an Option is True but the Option is hidden, the Option remains True for the configured instance.

Visibility is set through logic expressions assigned to Choices and Options. These logic expressions use the Boolean operators AND, OR, and NOT.

In derived templates that contain an Object Wizard from the parent object, icons are used to implement visibility settings that are in addition to the settings made in the parent object. Note that the visibility settings made in the parent Object cannot be changed in the derived template. See Object wizard visibility settings in a derived template for additional information.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Add a basic visibility expression

To add a basic visibility expression to a Choice Group, Choice or Option:

1. Highlight the item for which you are setting visibility (Choice Group, Choice or Option). You can select any item except the initial one. That is, you cannot make the visibility conditional of the first item (Choice Group or Option) in the Object Wizard. If the first item is a Choice Group, you cannot make it or any of its Choices conditional.
2. Place the cursor in the Visibility text box of the selected item, and type " **[** " (open bracket character). A drop-down list of all preceding items (Choices and Options) is displayed.

   Note: Choices are listed in format "*Choice Group Name::Choice Name*", for example, `Choice Group #1::Choice #1`. Options are listed in the format "*Option::State*", for example `Option #1::True`.

   Note: While you can simply select an item for a simple visibility expression without typing an open bracket, we recommend that you always type the open bracket as a matter of habit. As you begin building more complex visibility expressions, you risk deleting the expression if you select an item from the drop-down without first typing an open bracket.
3. Select the item from the drop-down list that will be used to control the visibility of the selected item. For example, you can make Option #2 visible only if Choice #1 of Choice Group #1 has been selected (Group #1::Choice #1), and/or if Option #1 is set to True (Option #1::True).

   For additional information, see Operators, rules, and behavior for visibility expressions.

   Note: When a user configures an instance, the default behavior of elements remains in effect, even if they are hidden from the user. The default Choice of a hidden Choice Group is still selected, and hidden Options remain in their default states.

Add a visibility expression with logical operators

To add a visibility expression that uses logical operators to a group, choice or option:

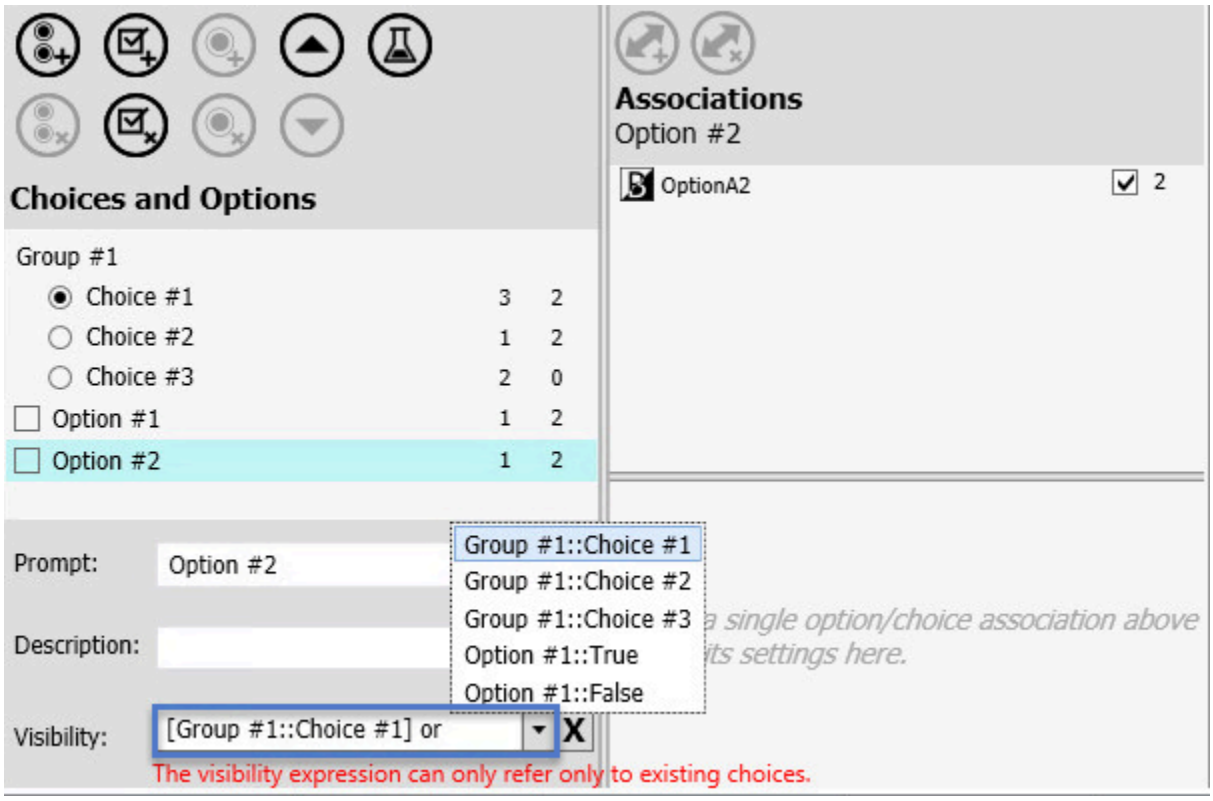
1. Highlight the item for which you are setting visibility (Choice Group, Choice or Option). You can select any item except the initial one. That is, you cannot make the visibility conditional of the first item (Choice Group or Option) in the Object Wizard. If the first item is a Choice Group, you cannot make it or any of its Choices conditional.
2. Place the cursor in the Visibility text box of the selected item, and type " **[** " (open bracket character). A drop-down list of all preceding items (Choices and Options) is displayed.

   Note: Always type an open bracket before selecting an item from the drop-down list. Simply selecting an item without first typing an open bracket will overwrite the existing expression.

   Note: Choices are listed in format "*Choice Group Name::Choice Name*", for example, `Choice Group #1::Choice #1`. Options are listed in the format "*Option::State*", for example `Option #1::True`.
3. Select the item upon which you are basing the visibility of the selected item. It will be entered in the visibility expression. To expand the expression:
   - Place the cursor at the front of the expression (before the open bracket) and enter the NOT operator, or
   - Place the cursor at the end of the expression (after the close bracket) and enter the AND or OR operator.
4. Type " **[** " (open bracket character) and select another item from the drop-down. All Object Wizard items must be enclosed in brackets. Enclose items and operators in parentheses as needed to ensure that the

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

expression resolves as intended.



5. Add additional operators, parentheses, and Object Wizard items as needed to complete the visibility expression. For example:

```
[ChoiceGroup1::Choice1] AND ([Option1::True] OR [Option2::True] OR [Option3::False])
```

Note: When a user configures an instance, the default behavior of elements remains in effect, even if they are hidden from the user. The default Choice of a hidden Choice Group is still selected, and hidden Options remain in their default states.

For additional information, see Operators, rules, and behavior for visibility expressions.

Operators, rules, and behavior for visibility expressions

As you build an Object Wizard, you can create visibility expressions that selectively hide Choice Groups, Choices, and Options. This can simplify the workflow by only showing Choices and Options that are applicable, based on Choices and Options that the user has already made. When an Option or Choice Group is hidden, the system applies the configured default values for attributes, graphics, and scripts associated with the hidden Option or Choice Group. You can leverage this capability to hide advanced functions from users tasked with configuring instances from templates that contain Object Wizards.

The **Visibility** prompt lets you assemble expressions to hide or show Choice Groups, Choices, or Options based on what the user has previously selected. Note that Visibility cannot be conditional for the first Object Wizard item (Choice Group or Option). The first Object Wizard item is always visible. Available logical operators are AND, OR, and NOT.

Observe these basic rules and behaviors when building visibility expressions:

- Always start by typing the " [" (open bracket character) in the visibility text box. This will bring up a list of all

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Choices (from other Choice Groups) and Options that precede the selected item. Do not simply highlight an element without first typing the open bracket character.

- Choices are listed with their Choice Group. Each Option is listed twice, for both states (True and False). Use fully qualified Option/Choice selections surrounded by brackets. The separator between Choice Groups and Choices, and between Options and their states is "::" For example:

  [Valve Type::3-Way] OR [Valve Type::4-Way] AND [Option1::True]

- Valid logical operators:
  - Valid binary operators are "AND" and "OR".
  - Valid unary operator is "NOT".
  - "AND" has a higher precedence than "OR".
  - "NOT" has a higher precedence than "AND" or "OR".
  - You can use "&" or "&&" instead of "AND".
  - You can use "|" or "||" instead of "OR".
  - You can use "!" instead of "NOT".
- Parentheses can be used to build complex expressions. For example:

  ```
  [ChoiceGroup1::Choice1] AND ([Option1::True] OR [Option2::True] OR [Option3::False])
  ```

  Note that without parentheses, this expression would resolve very differently, since "AND" has a higher precedence than "OR". If parentheses were not used, the expression would resolve this way:

  ```
  [ChoiceGroup1::Choice1] AND [Option1::True] OR [Option2::True] OR [Option3::False]
  ```
- Literals and operators are not case sensitive.
- Spacing between literals and operators is optional.
- The visibility expression editor validates the logical expression as you enter it, but does not check if the expression is possible. For example, the expression "[Valve Type::3-Way] AND [Valve Type::4-Way]" would not be flagged since it is logically correct, even though the condition is impossible (you cannot select more than one Choice from a single Choice Group).

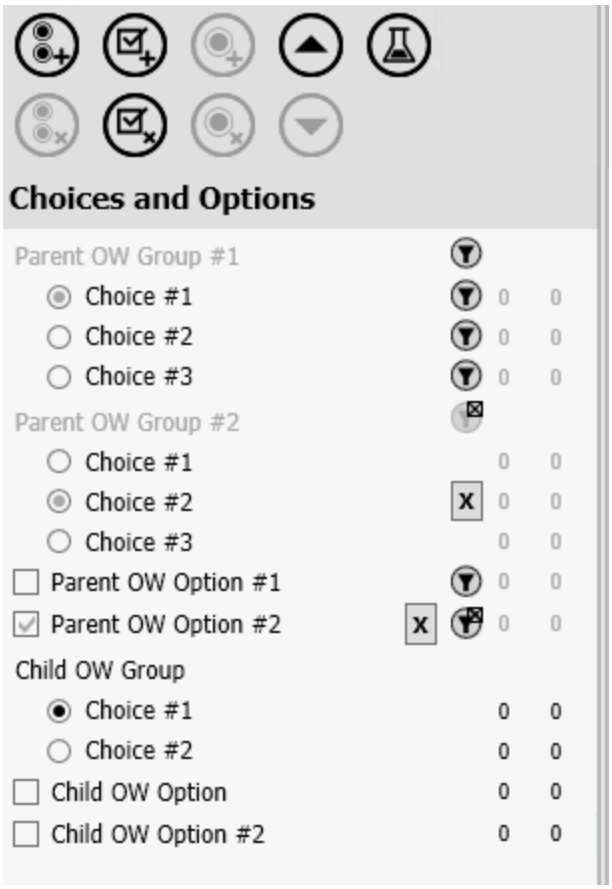
Object wizard visibility settings in a derived template

When you derive a template from a template with an Object Wizard, derived Object Wizard elements can no longer be edited, and Object Wizard group names are grayed out. Other noticeable changes are:

- A visibility icon appears next to each derived wizard item.
- You can override visibility settings by clicking on the visibility icon.
- The item counts are reset to zero; changes made in the derived object will increment the counters.
- If you change a default selection, the setting override indicator appears next to the changed setting.

See Object wizard user interface details for more information about visibility icons and the setting override indicator. For additional information about derivation and visibility, see Object wizard derivation.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.



The visibility expressions that were configured in the parent Object Wizard remain in effect. You can, however, hide Choice Groups, Choices, and Options. Instead of using Boolean expressions to set the visibility of Groups, Choices, and Options, use the visibility icon . When changing the visibility settings of a derived Object Wizard:

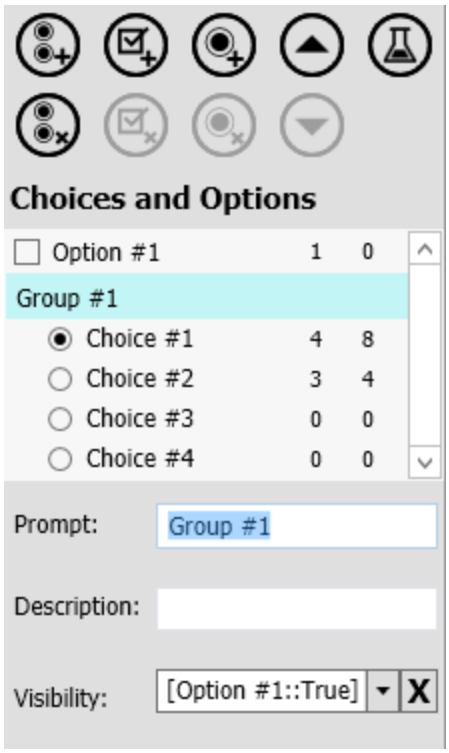
- The behavior of a hidden Choice or Option does not change when configuring an instance from the derived Object Wizard. The default state of a hidden Option remains in effect when the instance is configured. If the Option's default state is True, any attributes, graphics, or scripts associated with it are automatically configured for the instance. Similarly, the default Choice in a hidden Choice Group is automatically selected, along with its associated attributes, graphics, and scripts.
- You cannot hide (turn off the visibility of) a default Choice. You can, however, hide the entire Choice Group.
- If every Choice except the default is hidden, the entire Choice Group is hidden.
- If you change a default choice, the entire Choice Group is automatically hidden. You can turn visibility on again for the Choice Group. Once the Choice Group has been made visible, you can hide individual Choices.

Example 1: Basic visibility expressions

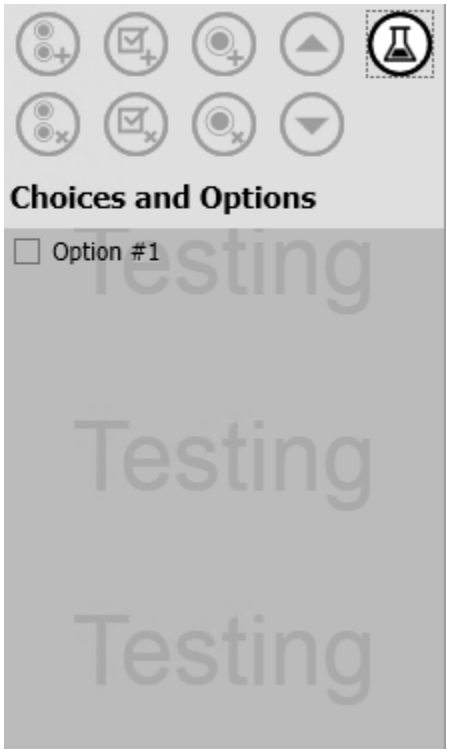
This example shows how to build a basic visibility expression. In this case, we want to hide a Choice Group unless an Option is True.

1. Select Group #1 and type [(open bracket) in the Visibility text box.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

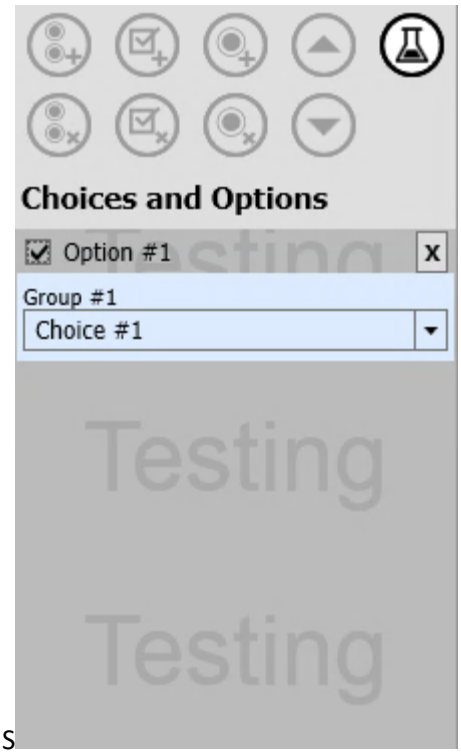


2. Select Test mode.
3. With Option #1 unselected (false), Group #1 is hidden.



4. Set Option #1 to True. Group #1 is now visible.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.



S

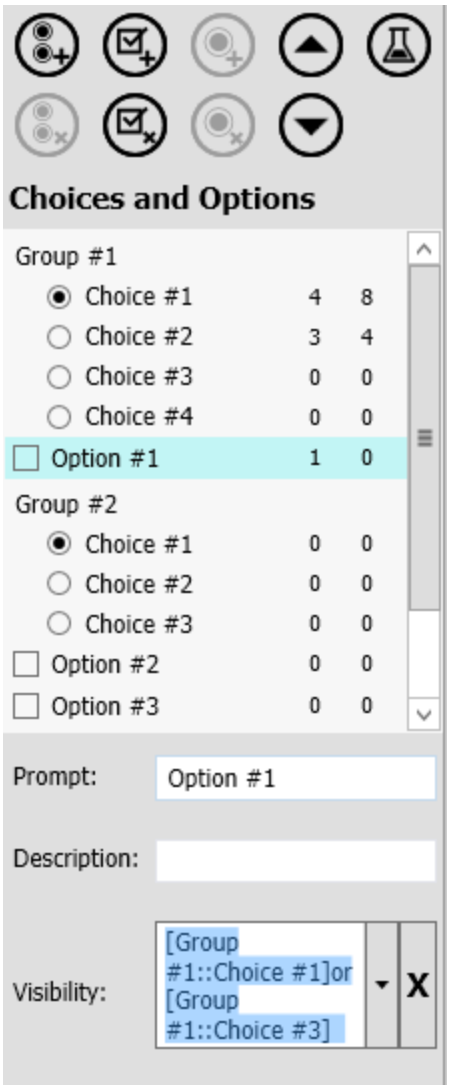
Example 2: Complex visibility expressions

Setting more complex rules for the visibility of Choices and Options requires the use of the logical operators AND, OR, or NOT, and may also necessitate the use of parentheses to ensure that the expression is parsed correctly. The rules for using logical operators are listed in Operators, rules, and behavior for visibility expressions.

Scenario 1: Visibility of option is contingent on specific choices

Consider the following scenario. We want to allow a user to select Option #1 only when Choice #1 or Choice #3 is selected from Group #1.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.



1. Select Option #1 and type [(open bracket) in the Visibility text box.
2. Select Group #1::Choice #1 from the drop-down list.
3. Place the cursor in the text box after] (close bracket) and type OR.
4. Type [and select Group #1::Choice #3 from the drop-down list. The Visibility text box should contain:

```
[Group #1::Choice #1] or [Group #1::Choice #3]
```

5. Select Test mode.
6. Select Group #1::Choice #1. Option #1 is visible
7. Switch to Group #1::Choice #3. Option #1 remains visible.
8. Switch to Group #1::Choice #2. Option #1 is now hidden.

Scenario 2: Visibility of option is contingent on a combination of choices and options

In this case, we want allow a user to select Option #3 only when Group#1::Choice #1 and Group#2::Choice #1 or Group#2::Choice #2 is selected and Option #2 is true. This expression can be interpreted in different ways:

- (Group#1::Choice #1 and Group#2::Choice #1) or (Group#2::Choice #2 and Option #2::True)

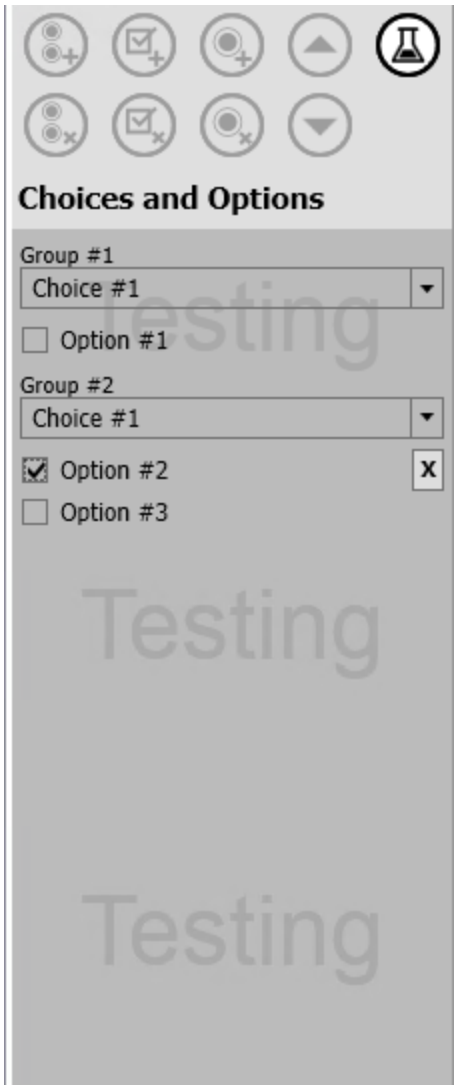
© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

- Group#1::Choice #1 and (Group#2::Choice #1 or Group#2::Choice #2) and Option #2::True

Let's assume that we want the latter interpretation.

1. Select Option #3 and type [(open bracket) in the Visibility text box.
2. Select Group #1::Choice #1 from the drop-down list.
3. Place the cursor in the text box after] (close bracket) and type AND.
4. Type [and select Group #2::Choice #1 from the drop-down list.
5. Place the cursor in the text box after] (close bracket) and type OR.
6. Type [and select Group #2::Choice #2 from the drop-down list.
7. Place the cursor in the text box after] (close bracket) and type AND.
8. Type [and select Option #2 from the drop-down list.
9. Position the cursor in the text box enclose Group#2::Choice #1 or Group#2::Choice #2 in parentheses.
10. The Visibility text box should contain:
    ```
    Group#1::Choice #1 and (Group#2::Choice #1 or Group#2::Choice #2) and Option #2::True
    ```
11. Select Test mode.
12. Select Group #1::Choice #1. and Group#2::Choice #1. Option #3 is hidden.
13. Set Option #2 to True. Option #3 is now visible.

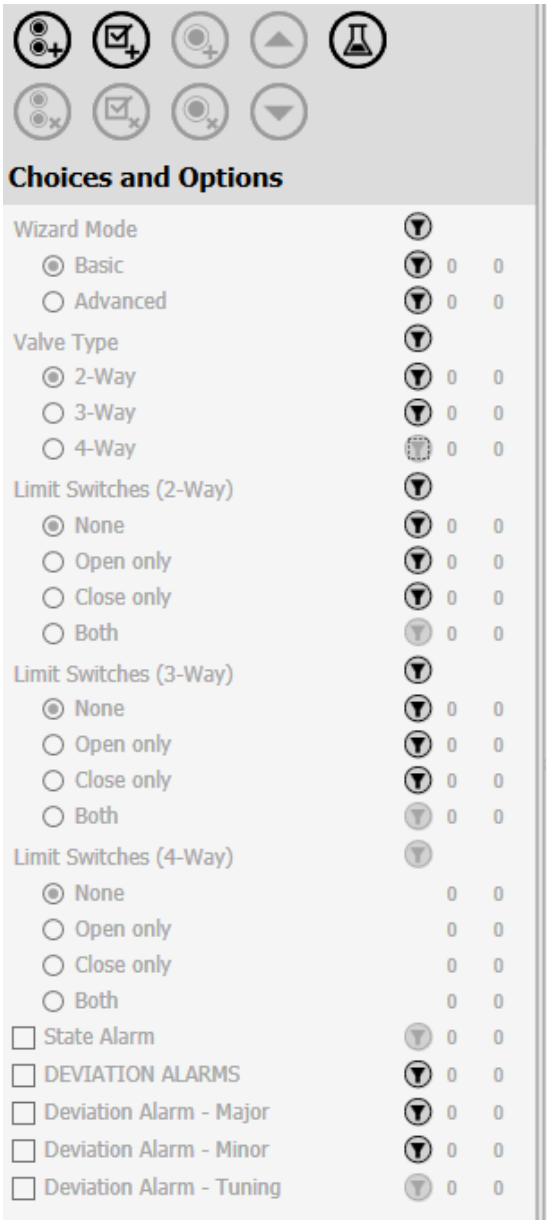
© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

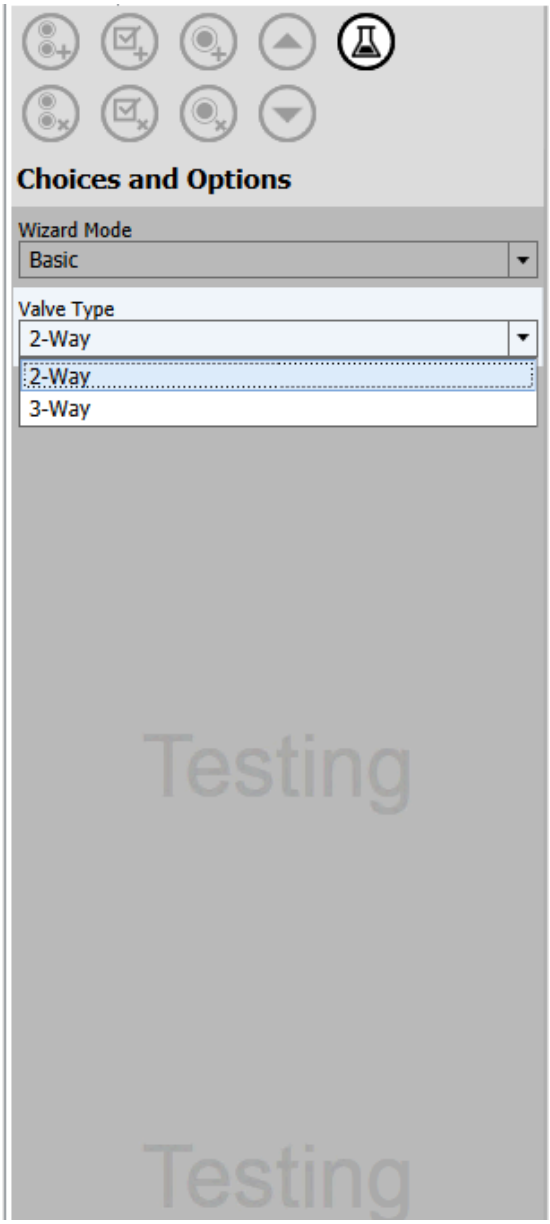


Example 3: Visibility of groups, choices, and options in a derived object wizard

This example illustrates how a derived Object Wizard with hidden Choices and Options is translated at run time and test mode:

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.





In the example above, the following items are hidden:

- Group: Limit Switches
- Choice: 4-Way (in the Group Valve Type)
- Choice: Both (in the Groups Limit Switches (2-Way) and Limit Switches (3-Way))
- Option: State Alarm
- Option: Deviation Alarm - Tuning

Object wizard trimming

Trimming refers to the ability of the Object Wizard to remove unneeded attributes, graphics, and scripts from derived instances. By removing unneeded items, the size of instances an be minimized, thus allowing more efficient utilization of runtime system resources. This can also result in faster deployments.

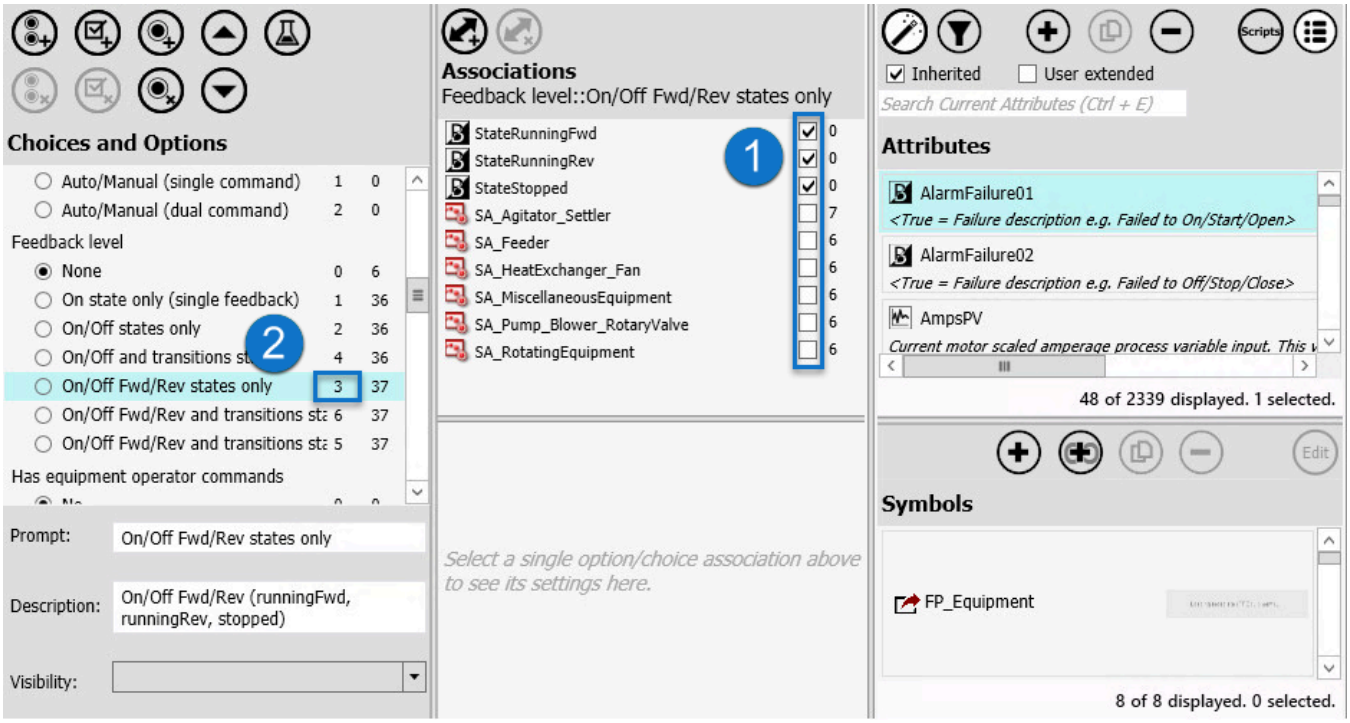
© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

- When you associate an attribute with a Choice or Option, the **Trim** checkbox (1) is checked by default. This means that trimming is enabled, and the attribute will be evaluated for trimming.
- When you associate a graphic with a Choice or Option, the **Trim** checkbox is cleared by default. This means that trimming is DISabled, and the graphic will NOT be evaluated for trimming.
- When you associate a script with a Choice or Option, the **Trim** checkbox is checked. Trimming cannot be disabled for scripts.

Note: Graphics do not exist as runtime objects, but do exist as part of a ViewApp. Trimming unnecessary graphics can help to reduce system resource requirements.

The total number of trimmable items (attributes, graphics, and scripts) associated with each Object Wizard Choice and Option is displayed in the **Choices and Options** pane (2). Items that are not enabled for trimming are not included in this count. The default trim state of associated graphics is disabled.

Note: The count of trimmable items resets to 0 in derived templates.



Enable trimming

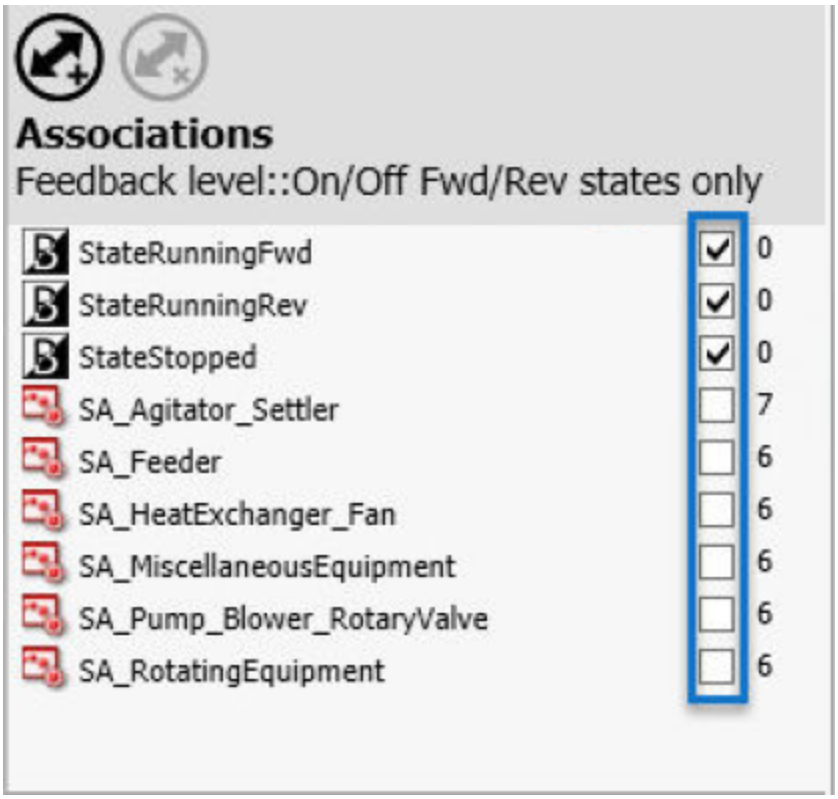
Whether or not an attribute or graphic associated with a Choice or Option will be evaluated for trimming is determined by the state of its trimming checkbox in the **Associations** pane. See Object wizard user interface details for additional information.

Note: Scripts are always evaluated for trimming.

To enable trimming:

- Select the **Trim** checkbox in the **Associations** pane.
  - When checked, trimming is enabled.
  - When unchecked, trimming is disabled.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.



Graphics and attributes are evaluated for trimming when an instance is derived, in accordance with the following behaviors (scripts are always evaluated for trimming):

- If none of its associations have trimming enabled, the item is not evaluated for trimming and is always propagated to derived instances.
- The item is propagated to the instance if any Choice or Option where trimming is enabled is selected.
- The item is trimmed if at least one of its associations has trimming enabled, and none of the associations with trimming enabled are selected (even if associations with trimming disabled have been selected).
- Trimming occurs only when the instance is saved.

Note: Trimming is not applicable to derived templates, since these are not runtime objects.

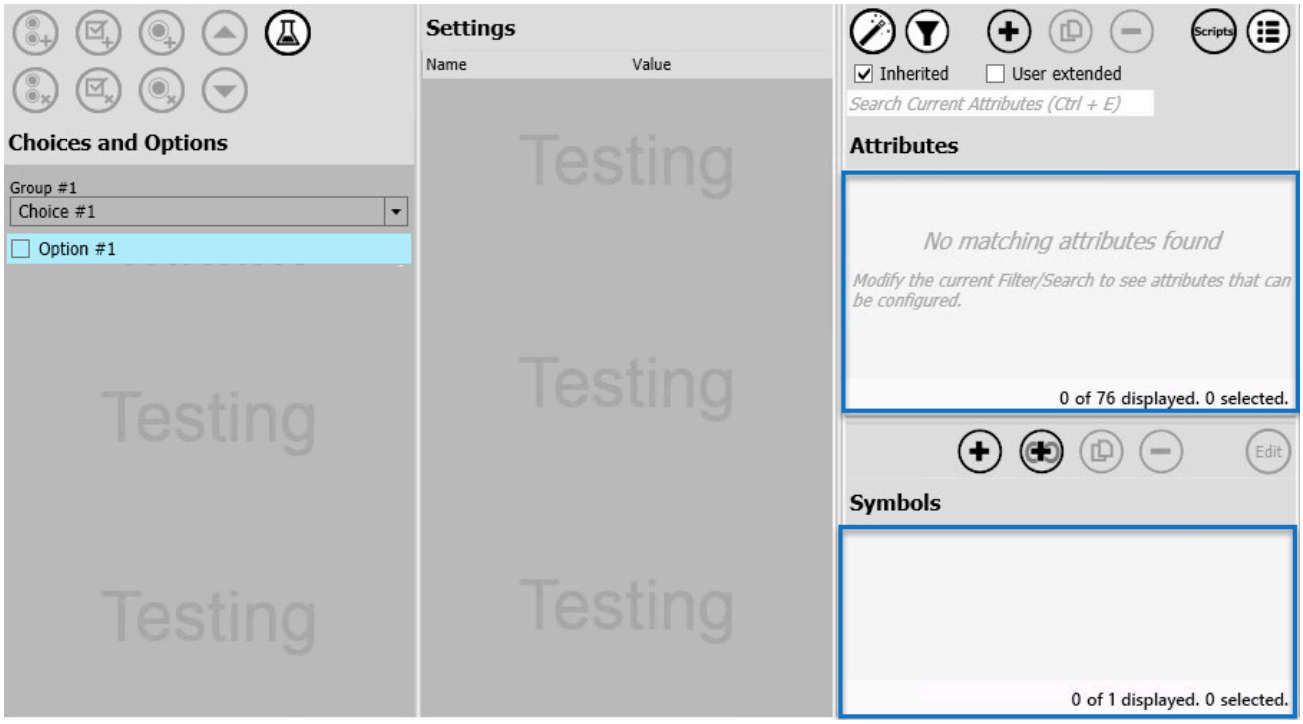
Example: Enable trimming for attributes and graphics

When you associate an attribute with a Choice or Option and set it as trimmable, you can use Test mode to see how attributes and graphics will be added to, or removed from, instances that are configured from the Object Wizard. For example:

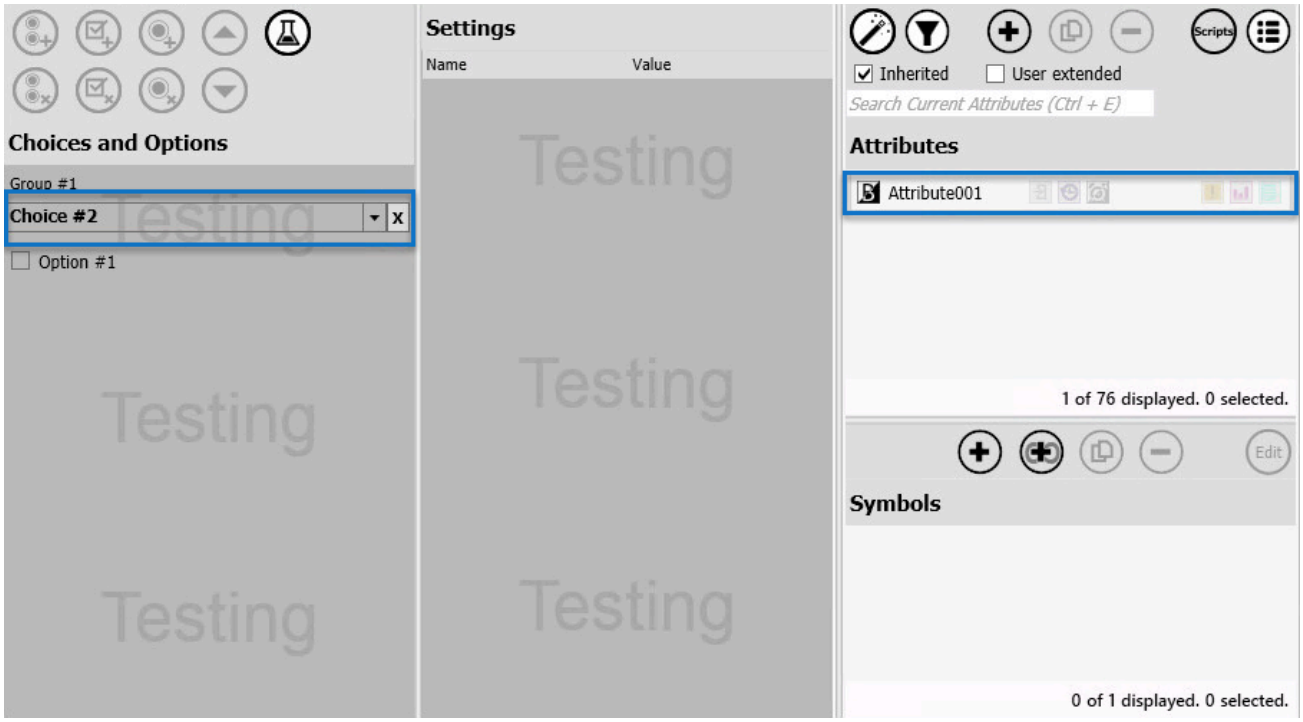
1. Create a derived template and open it in the Object Editor.
2. Add an attribute: Attribute001.
3. Add a graphic: S1.
4. Add a Choice Group (with two Choices) and one Option. Leave Choice #1 as the default choice, and leave the default state of Option #1 as not selected.
5. Associate Attribute001 and S1 with Choice #1, and disable trimming for both.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

6. Associate Attribute001 with Choice#2, and enable trimming for the attribute.
7. Associate S1 with Option #1, and enable trimming for the graphic.
8. Select **Test mode**. Note that with the default selections, Attribute001 does not appear in the **Attributes** pane, and S1 does not appear in the **Symbols** pane.

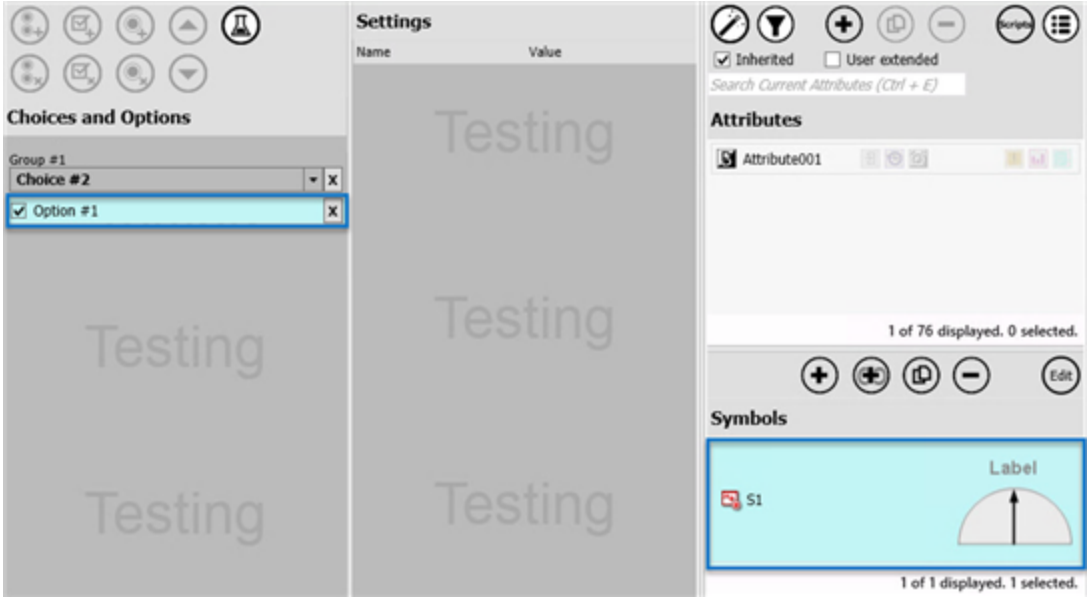


9. Select Choice #2. Note that Attribute001 now appears in the **Attributes** pane.



10. Enable Option #1. Note that S1 now appears in the **Symbols** pane.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.



Attribute and graphic overrides

You can associate different configurations of graphics and attributes with different Object Wizard Choices and Options. You can also allow users to override values that you previously set as they configure derived objects. See Configurable settings for attribute and graphic values for additional information about setting overrides and visibility for Symbol Wizard and custom property values. For detailed information about Symbol Wizards and custom properties, see Using custom properties with Application Server and "Working with Symbol Wizards" in the *Creating and Managing Industrial Graphics User Guide*.

The **Settings** pane lets you configure setting overrides and visibility for associated attributes and graphics.

Attribute Overrides

For attributes, you can override default settings and set visibility for:

- Attribute description
- Data type
- Initial value
- Engineering units (if applicable)
- Activate/deactivate features, such as I/O and alarms
- Feature values

Attribute Setting Visibility

When you associate an attribute with an Object Wizard Choice or Option, its settings are **hidden by default**. You can make settings visible by selecting the checkbox next to the setting. You can also make overrides visible.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Graphic Overrides

For graphics, you can override default settings and set visibility for:

- Symbol Wizard Choices and Options
- Custom properties

Graphic Setting Visibility

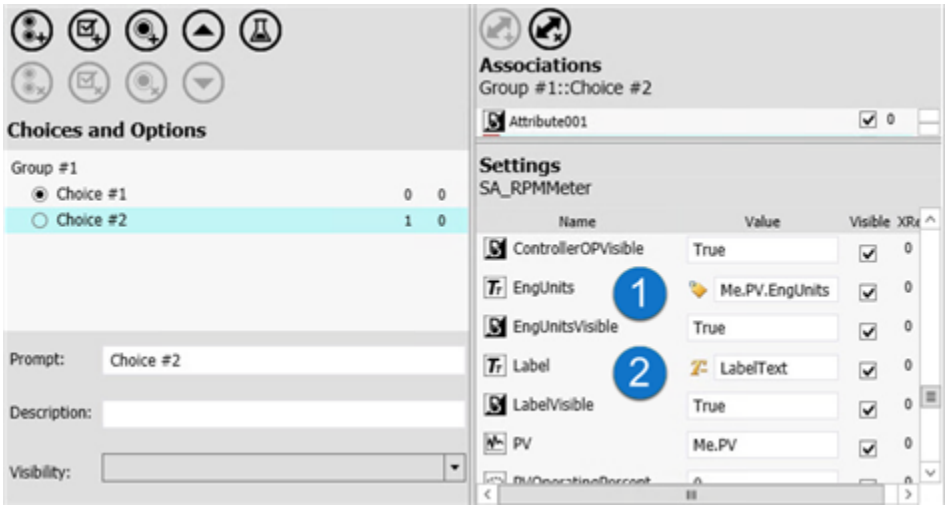
When you associate a graphic with an Object Wizard Choice or Option, Symbol Wizard and custom property settings (if any) are **visible by default**. If you override a graphic custom property or Symbol Wizard setting, its visibility automatically turns off (the setting is hidden). You can make an overridden custom property setting visible again by selecting the checkbox next to the setting.

Note: If you override a Symbol Wizard setting, the only way to make the setting visible again is to remove the override.

Custom properties can be Boolean, integer, float, string, etc. See Object wizard user interface details for additional information. For detailed information about custom property data types, see Using custom properties with Application Server. When configuring string, time, and elapsed time custom properties, a toggle is provided in the **Settings** pane to configure the custom property as either:

	Reference string or an expression (1). Reference strings will be evaluated to determine their meanings. For example, `Me.Tank1.InletValve.PV`.
	String literal (2). String literals are passed as static text.

To change the string type, select the icon next to the text field.



Enter overrides and set visibility for attribute and graphic values

For both attributes and graphics that contain a wizard or custom properties, you can enter overrides that apply only to the associated Object Wizard Choice or Option. While the following procedure shows how to configure a

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

graphic, the procedure for configuring attributes is virtually identical. See also Attribute and graphic overrides.

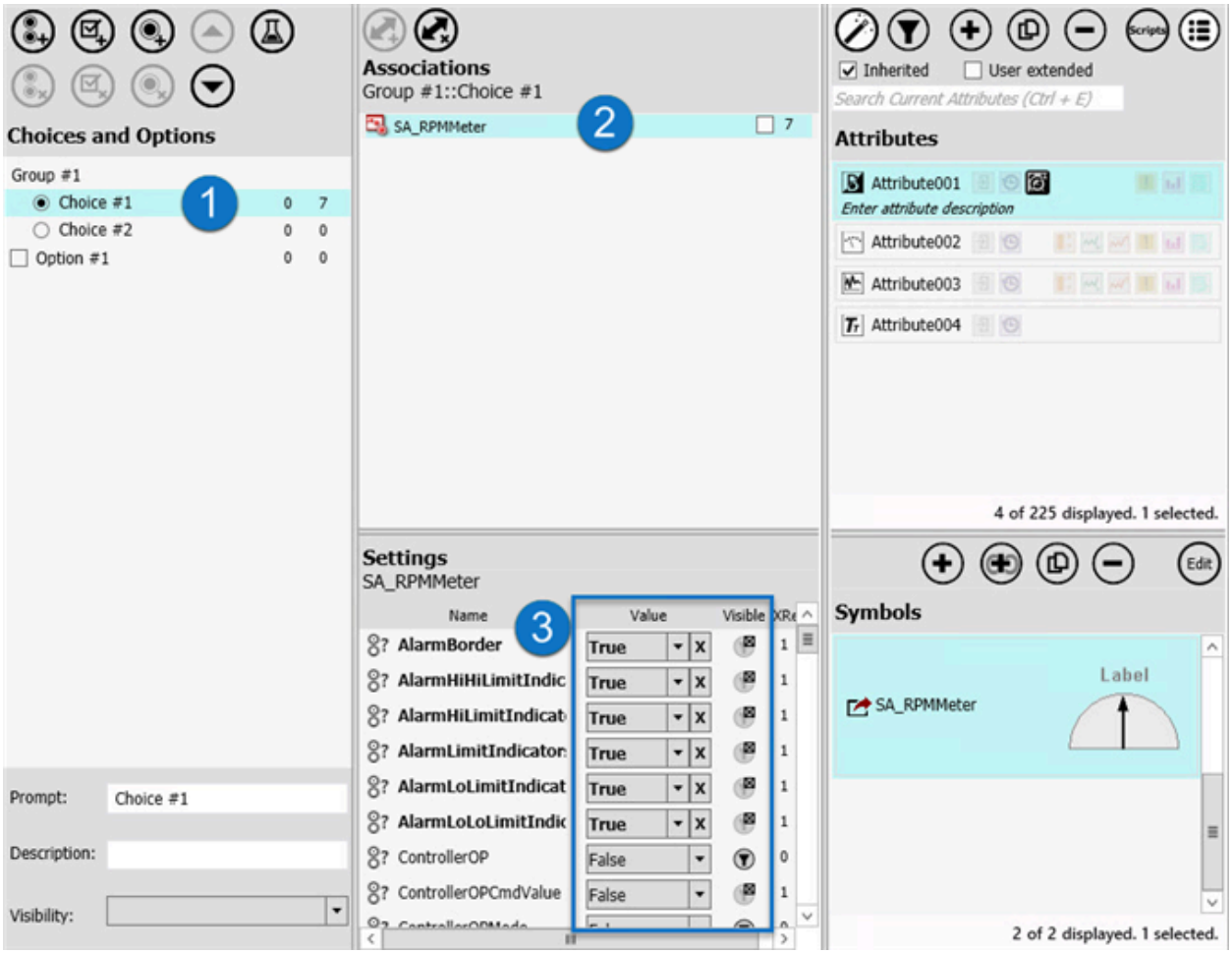
To configure an Override for a Choice or Option

1. Select the Choice or Option you want to configure (**Choices and Options** pane).
2. Select a graphic or attribute (**Associations** pane) associated with the Choice or Option. See Associate Content with a Choice or Option for information about associating graphics with Object Wizards.

   This does not apply to scripts since they can only be associated with a Choice or Option. There are no configuration or visibility settings that can be configured.
3. In the **Settings** pane, override the default values as needed. Overrides are shown in **bold** and by an **X** next to the new value. See
   - You can resize columns in the **Settings** pane as needed by selecting and moving the boundary between column headers.
   - Symbol Wizard and Custom Property settings are visible by default. Overrides are automatically hidden from users.
     - You can only restore the visibility of a Symbol Wizard setting with an override by removing the override.

       Note: This behavior is unique to Symbol Wizard overrides. You can change visibility settings for both Custom Property overrides and attribute overrides.
     - You can restore the visibility of a Custom Property with an override by selecting the Visibility icon until it is unmuted. The icon will have a small X in its upper right quadrant to indicate that Visibility has been overridden.
   - Switching a Custom Property string type between expression/reference and static text is an override (value is displayed with **bold** type with an **X** next to the value and in the Visibility setting).
   - Attribute settings are hidden by default. You can override the Visibility settings for attributes that use their default setting, as well as for attributes that have an override.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.



Note: Attribute, Symbol Wizard, and Custom Property values that are marked as visible can be overridden by users as they configure instances.

Note: The total number of configured settings (visibility settings and value settings) is shown in the **Associations** pane, and is reflected in the **Choices and Options** pane. See Attribute and graphic overrides for additional information.

Attribute and graphic overrides

Override values for an attribute or graphic (with a Symbol Wizard or custom properties) can be configured for each associated Choice or Option. When you associate an attribute or graphic, the default values are listed in the **Settings** pane. To configure values that apply to the selected Choice or Option, enter the values in the **Settings** pane. If the attribute or graphic is associated with multiple Choices or Options, different values can be used for each one. Values for the Choices and Options are saved when you save the object. If the setting is marked as visible, users can override your setting (or override the default, if you have not configured a setting) as they derive and configure instances from the Object Wizard.

Graphics with overrides that affect their shape, for example, certain Situational Awareness Library graphics, are rendered with the applicable shape-changes as defined by the overrides.

Note: If you override a Symbol Wizard setting, the only way to make the setting visible again is to remove the override. Attributes and custom properties with overrides can be made visible again by selecting the visibility

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

checkbox.

A configured setting is defined as any change from the default configuration of the attribute or graphic. This includes changing a value or visibility. For example, you might change the default Boolean value from true to false, or you might change an integer value from 5 to 10, or change engineering units from liters to cc. Changing the visibility of a setting also counts as a configured setting.

- The default visibility setting for attribute values is hidden (Visibility icon is muted (grayed-out) in the **Settings** pane).
- The default visibility setting for Symbol Wizard and custom property values is visible (Visibility icon is unmuted in the **Settings** pane).
- Visibility overrides are indicated by a small X in the upper right quadrant of the visibility icon.

The total number of configured settings for each individual attribute and graphic is displayed in the **Associations** pane. The total number of configured settings for ALL attributes and graphics associated with an individual Choice or Option is displayed in the **Choices and Options** pane.

Configurable settings for attribute and graphic values

Graphics and attributes can be configured for each Choice or Option with which they are associated. When you initially associate an attribute or graphic with a Choice or Option, the default settings of the attribute or graphic are listed in the **Settings** pane.

Note: Icons for attribute and graphic settings are defined in Object wizard user interface details.

To override default values, enter the new values in the **Settings** pane **(1)**. If the attribute or graphic is associated with multiple Choices or Options, different overrides can be used for each one. These configured (override) values are saved when you save the template.

An override is defined as any change from the default graphic configuration. For example, you might change the default Choice of a Symbol Wizard, or change a custom property or attribute value. Changing the visibility of a setting is also considered an override. By default, graphic settings are visible in derived objects, while attribute settings are hidden by default.

If the setting is marked as visible, users can enter overrides when they use the **Create New Asset** editor to configure a new instance. If the setting is hidden, it is not exposed to users and therefore the default value cannot be overridden.

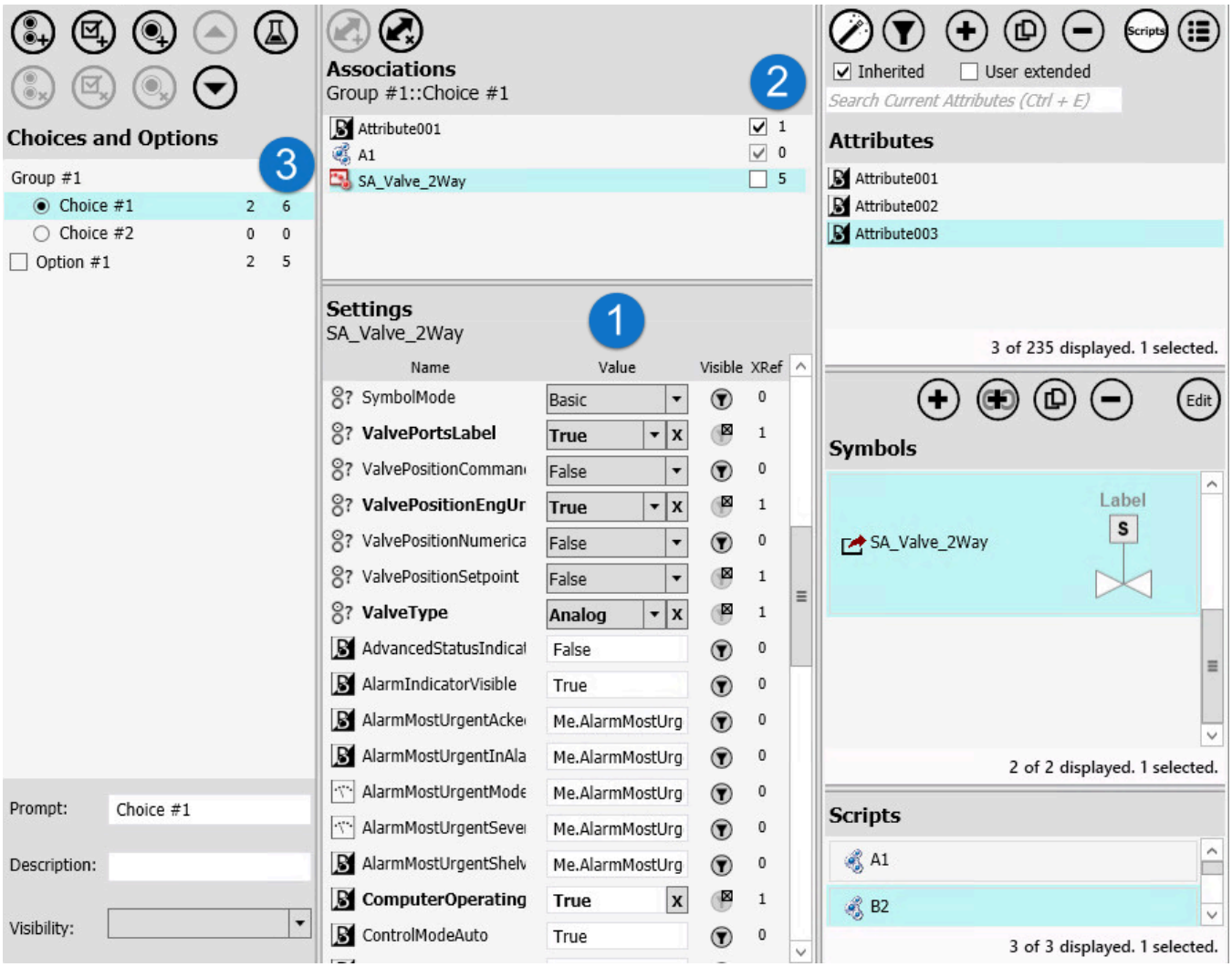
Symbol Wizard overrides are automatically hidden and cannot be changed back to visible, except by restoring the original setting. The names of configured settings are displayed in bold type, and have an "**X**" next to the setting. To restore the original setting, clear the **X**.

Configured visibility settings have a small x in the upper right quadrant of the visibility icon. To restore the original, click on the icon twice to clear the override. If the original visibility state was visible, the icon will cycle through override - hidden, override - visible, and then return to its original state of visible (no override).

Custom property and attribute overrides are also hidden automatically. However, you can override the visibility setting for a setting that has an override value and make it visible again.

The total number of configured settings for each individual attribute and graphic is displayed in the **Associations** pane **(2)**. The total number of overrides for ALL attributes and graphics associated with a Choice or Option is displayed in the **Choices and Options** pane **(3)**.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.



Object wizard derivation

You can derive child templates from templates with Object Wizards. Creating a second (or deeper) level of derivation from a template with an Object Wizard lets you extend the functionality of the original Object Wizard, and make specialized versions for specific applications. See Reasons to derive a template from an object wizard for additional information.

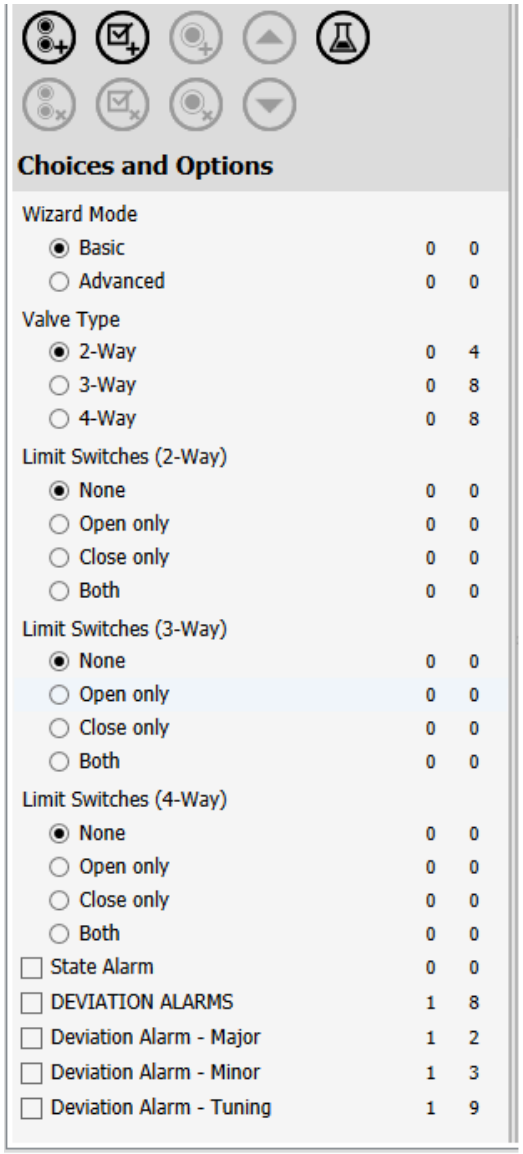
Derive a child template from a template with an object wizard

To create a derived template from a template with an Object Wizard:

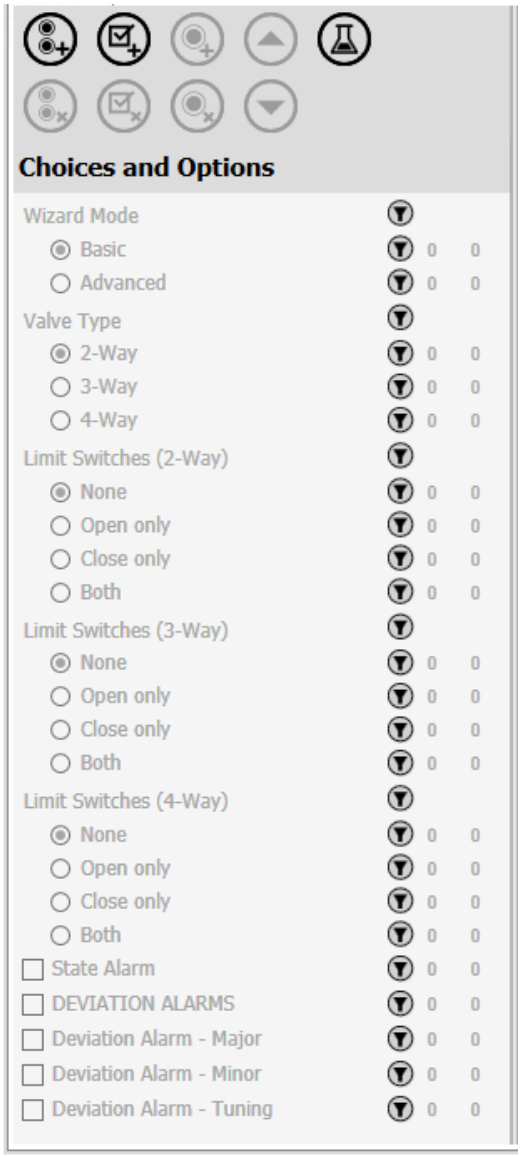
1. Select the parent template with Object Wizard in the Template Folder.
2. Do one of the following:
   - Right click on the template and select **Derived Template**.
   - Use the keyboard shortcut: **Ctrl + Shift + N**.

   You will notice that the child template looks different than the parent template.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.



Original Object Wizard



Derived Object Wizard

See Object wizard visibility settings in a derived template for additional information.

Set visibility of choices and options in a derived template

The original Object Wizard cannot be changed in the child derived template. However, you can set the visibility of Choice Groups, Choices, and Options. You can also set different defaults for Choice Groups, and you can turn Options on or off.

To set Object Wizard visibility in a derived template, do any of the following:

1. Select the visibility icon next to a Choice Group, Choice, or Option.
   - When you hide a Choice Group, all its Choices are hidden as well.
   - When you hide a Choice, the remaining Choices in the Choice Group are still visible.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

- When only one Choice is visible in a Choice Group, the entire Choice Group is hidden.

  Note: The system applies the default values and actions of hidden Options and Choice Groups when a user derives an instance from the template. Visibility does not change configuration, it only affects aspects of the wizard that are visible to the user.

2. Change the default Choice in a Choice Group. This hides the Choice Group.

   You can select the visibility icon to make the Choice Group visible again. You can then set visibility of the Choices individually.

3. Change the default state of an Option. This hides the Option.

   You can select the visibility icon to make the Option visible again.

Reasons to derive a template from an object wizard

There are two main reasons for deriving a child template from a template that contains an Object Wizard:

1. To extend the existing Object Wizard and thus, make it more capable. Available actions to accommodate the specific needs of a target solution include:
   - Add Choice Groups or Options.
   - Add attributes, graphics, or associations.
2. To create specialized versions of templates that preconfigure choices, and thus simplify the task of configuring instances. For example, you can define common configuration aspects that are applicable to a set of similar instances, or modify some of the characteristics of the existing Object Wizard. Available actions to simplify configuration include:
   - Change default Choices or settings.
   - Hide Choice Groups, Choices, or Options.
   - Enter overrides for attribute or graphic settings, or change their visibility settings.

Using derived templates lets you propagate changes made to a parent template, as long as the child template does not have an override for what has changed. See Propagation of attribute and graphic settings for additional information.

If you are familiar with using Application Server, you will notice one other advantage of using derived Object Wizards: there is no need to lock objects. Changes automatically propagate to derived objects (provided the child template does not have an override for what changed in the parent), without explicitly locking attributes in the parent object.

Changes to object wizards in derived templates

After you create a derived template from a template with an Object Wizard, you can only make certain changes to the existing Object Wizard in the derived object.

- You can set overrides for attributes and graphics.
- You can hide or show settings for attributes and graphics.
- You can change visibility and defaults for Choices and Options.
- You can add new Choice Groups and Options (you cannot add new Choices to an existing Choice Group).

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

- You can add new attributes, graphics, and scripts, and associate them with the existing Object Wizard.
- You cannot remove existing associations from the Object Wizard.
- You cannot remove any existing Choice Groups, Choices, or Options.
- You cannot change the order of any existing Groups, Choices, or Options.
- You cannot change attribute, graphic, or script associations with any existing Choices or Options.

When a Choice or Option is changed from the default, the change is indicated by by an **X** next the modified Choice or Option. To revert to the default, select the **X**.

Similarly, when a visible setting for an associated attribute or graphic has been changed from the default, the change is indicated by an X next to the modified setting. To revert to the default, select the **X**.

Behavior of object wizard components in a derived template

The following list summarizes actions that you can and cannot perform in derived templates that contain an Object Wizard.

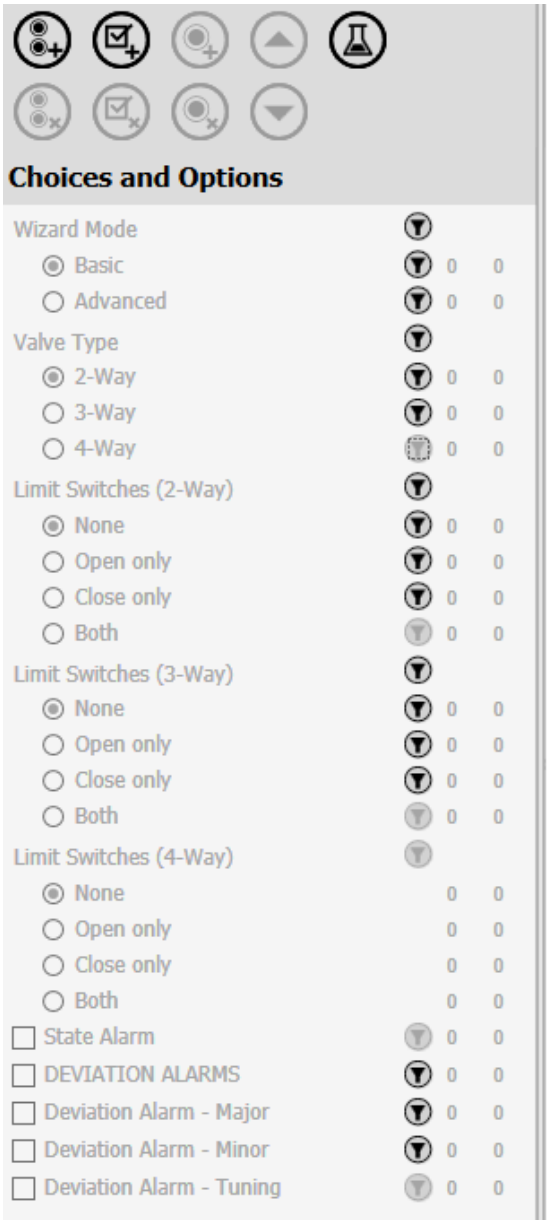
- Inherited Groups, Choices and Options cannot be edited.
  - Groups, Choices and Options cannot be renamed or deleted.
  - Their order cannot be changed (cannot be moved up or down).
  - Their prompts, descriptions, and visibility rules cannot be edited.
- You can change the default Choice, or the default state of an Option.
- Associations of inherited items (attributes, graphics, and scripts) with Choices and Options cannot be changed, and inherited associations cannot be removed.
- You can associate an inherited item with additional inherited Choices and Options
- Inherited attribute and graphic values and visibility settings can be overridden.
- Inherited values and visibility settings with overrides can be overridden.

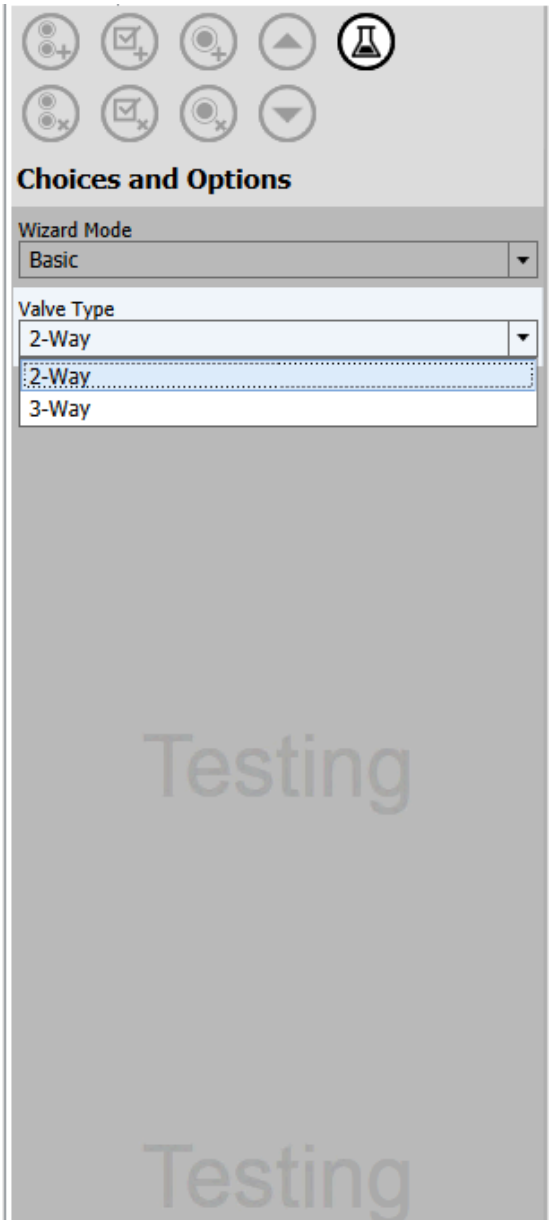
  Note: Even if a value is overridden in the parent object, the value will appear as a default value in the child object. This is because the override from the parent becomes the default for the child.

Example: Set visibility of derived wizard choices and options

The following example illustrates how a derived Object Wizard with hidden Choices and Options is interpreted in test mode. This simulates instance configuration.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.





In the example, the following items are hidden:

- Group: Limit Switches
- Choice: 4-Way (in the Group Valve Type)
- Choice: Both (in the Groups Limit Switches (2-Way) and Limit Switches (3-Way))
- Option: State Alarm
- Option: Deviation Alarm - Tuning

Object wizard test mode

Test mode lets you simulate how an Object Wizard will behave when a user configures a derived instance. Use Test mode as you develop your Object Wizard to ensure that it behaves as expected. For complex Object Wizards, associated attributes and graphics may not always appear as you had planned. Test mode lets you verify

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

that derived instances will behave as intended.

- Select the **Test** button to enter Test mode.

- Deselect the button to exit Test mode.

Visibility expressions determine which Choice Groups, Choices, and Options are visible or hidden (along with associated graphics, attributes, settings, and configured overrides), based on previous selections.

Verify behavior of visibility expressions

1. Open the template, containing the Object Wizard to be tested, in the **Object Editor**.
2. Select the **Test** mode button. The background color of the **Choices and Options** and **Settings** panes darkens, and the **Associations** pane is no longer shown. Choice Groups and Options are displayed, based on their visibility settings.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.



3. Check that the Choice Groups and Options that you expect to be visible are shown.
   - Regardless of the construction of the Object Wizard, the first item (Choice Group or Option) in the Object Wizard is always visible.
   - Choice Groups are displayed as drop down lists, with only the default Choice visible.
4. Check that the expected default Choice for each visible Choice Group is shown.
5. Check that the default state (checked or unchecked) of each visible Option is correct.
6. Cycle through different Choice and Option selections, and verify that subsequent dependent items are hidden or shown as designed.
7. Verify that the settings for associated attributes and graphics are shown in the **Settings** pane, and that settings that should be visible are marked as visible. Verify that settings with visibility turned off are marked

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

as hidden.

About object wizard behavior in test mode

At the most basic level, you can check that Object Wizard elements (Choice Groups, Choices, and Options) operate as intended. Choices and Options should appear in their default states. That is, Options that should be enabled by default should appear with their checkbox checked, and Choice Groups should show the default Choice. You can change these selections while in Test mode, to ensure that Choices and Options are correctly configured, including visibility settings and associated attributes, graphics, and scripts. Attribute and graphic settings that should not be changeable by consumers of the Wizard should not be visible.

- Check conditional visibility expressions. See Conditional visibility expressions for choice groups, choices, and options for additional information.
- Check that attributes and graphics are shown in their correct configuration when the corresponding Choice or Option is selected. See Enter Overrides and Set Visibility for Attribute and Graphic Values for additional information.
- Check that attributes and graphics are correctly trimmed. See Object wizard trimming for additional information.

Example 1: Verify behavior of associated attributes

Here, we examine how the selection of Choices and Options affects attributes that are associated with them. For an overview of how an Object Wizard behaves in Test mode, see About object wizard behavior in test mode.

Let's start by creating new attributes and associate them with Options.

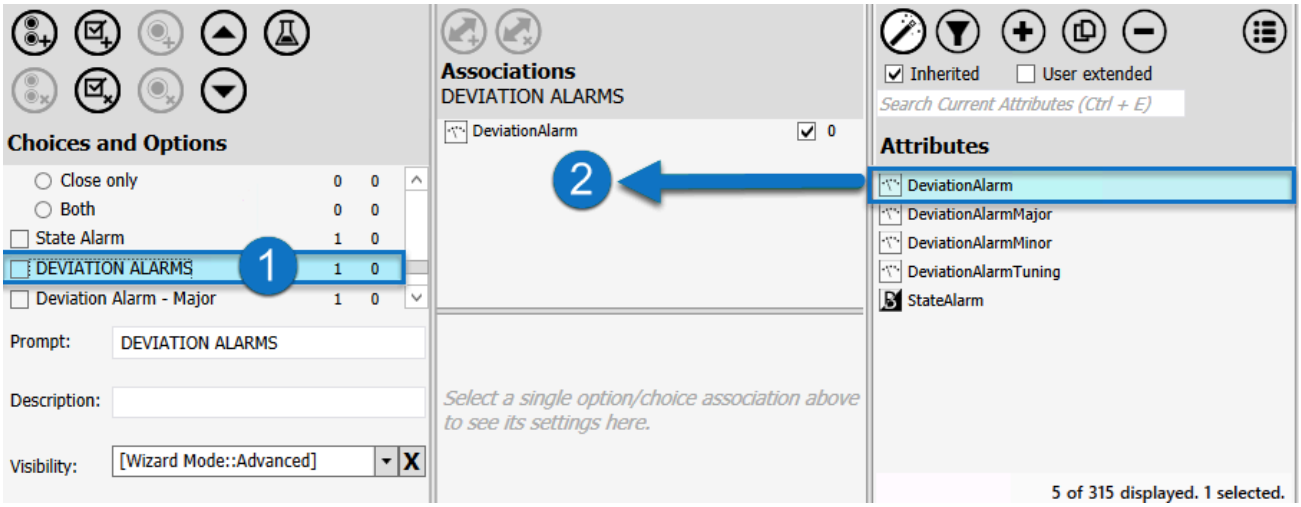
1. Create four new attributes.

   Name the attributes as follows:

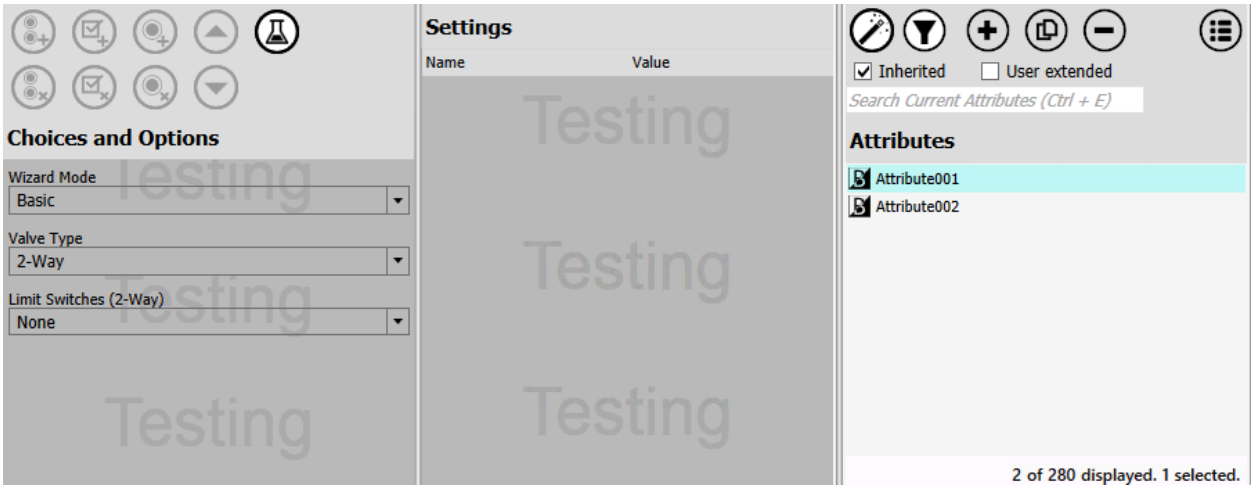
   - DeviationAlarms
   - DeviationAlarmMajor
   - DeviationAlarmMinor
   - DeviationAlarmTuning
2. Highlight one of the Deviation Alarm Options (1), then select the corresponding attribute and drag it to the **Associations** pane (2). For example, with the DEVIATION ALARMS option highlighted in the **Choices and Options** pane, drag and drop the DeviationAlarms attribute to the **Associations** pane.

   Make sure that the **Trimming** checkbox is selected in the **Associations** pane. See Object wizard trimming for additional information.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

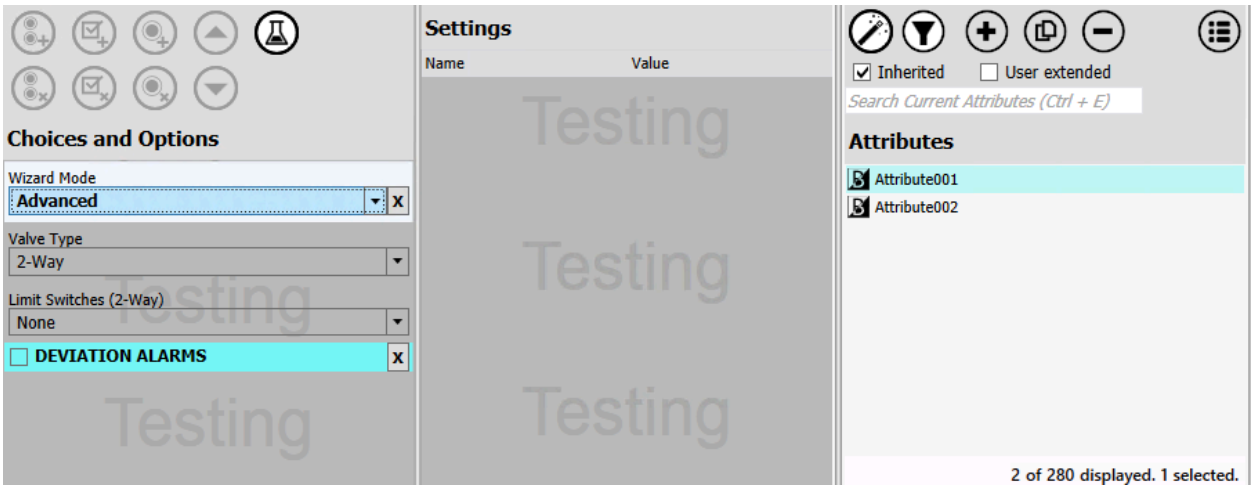


3. Repeat for the remaining Deviation Alarms with trimming enabled.
4. Create two more attributes: Attribute001 and Attribute002. We will leave these attributes unassociated with any particular Choice or Option. Since they are not a part of the Object Wizard, these attributes will be included in all objects that are derived from the template; these two attributes will not be trimmable.
5. To verify attribute behavior, select the **Test** button.
   - With no Object Wizard selections made, the attributes that are not associated with any Wizard Choices or Options (Attribute001 and Attribute002) are shown in the **Attributes** pane. Attributes associated with Choices or Options (DeviationAlarm attributes) are not shown.

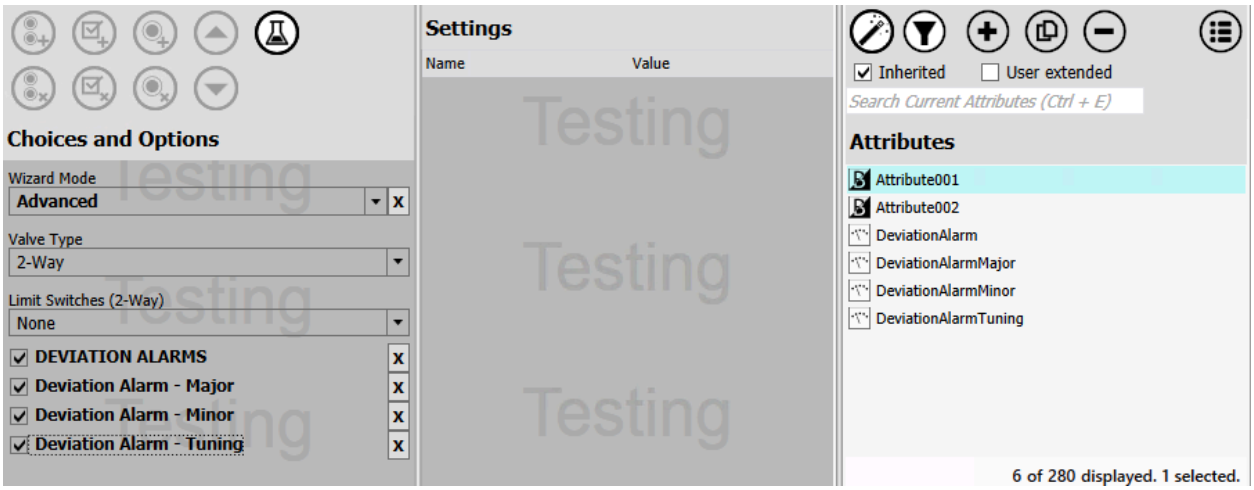


   - Change the Wizard Mode to **Advanced**. This displays the DEVIATION ALARMS option (unchecked). Note that the associated DeviationAlarms attribute is not shown.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.



- To display the DeviationAlarm attribute, set the DEVIATION ALARMS option to TRUE (checked). Continue setting the remaining Deviation Alarm options to TRUE. Note that as each Option is selected, its corresponding attribute is displayed.



- Clearing an Option (setting it to FALSE) turns off the attribute.

Note: When configuring an instance from an Object Wizard, unused attributes that have trimming enabled are not propagated to the instance. Only attributes that are not associated with the Object Wizard, and attributes with trimming enabled that evaluate as required, are present in the instance after the instance is saved. Trimming does not occur until after an object is saved. Any configured overrides for trimmable attributes remain in memory until the user saves the instance.

Example 2: Verify behavior of associated content

Here, we examine how the selection of Choices and Options affects graphics that are associated with them. For an overview of how an Object Wizard behaves in Test mode, see About object wizard behavior in test mode.

Start by creating new graphics and associate them with some Object Wizard Choices.

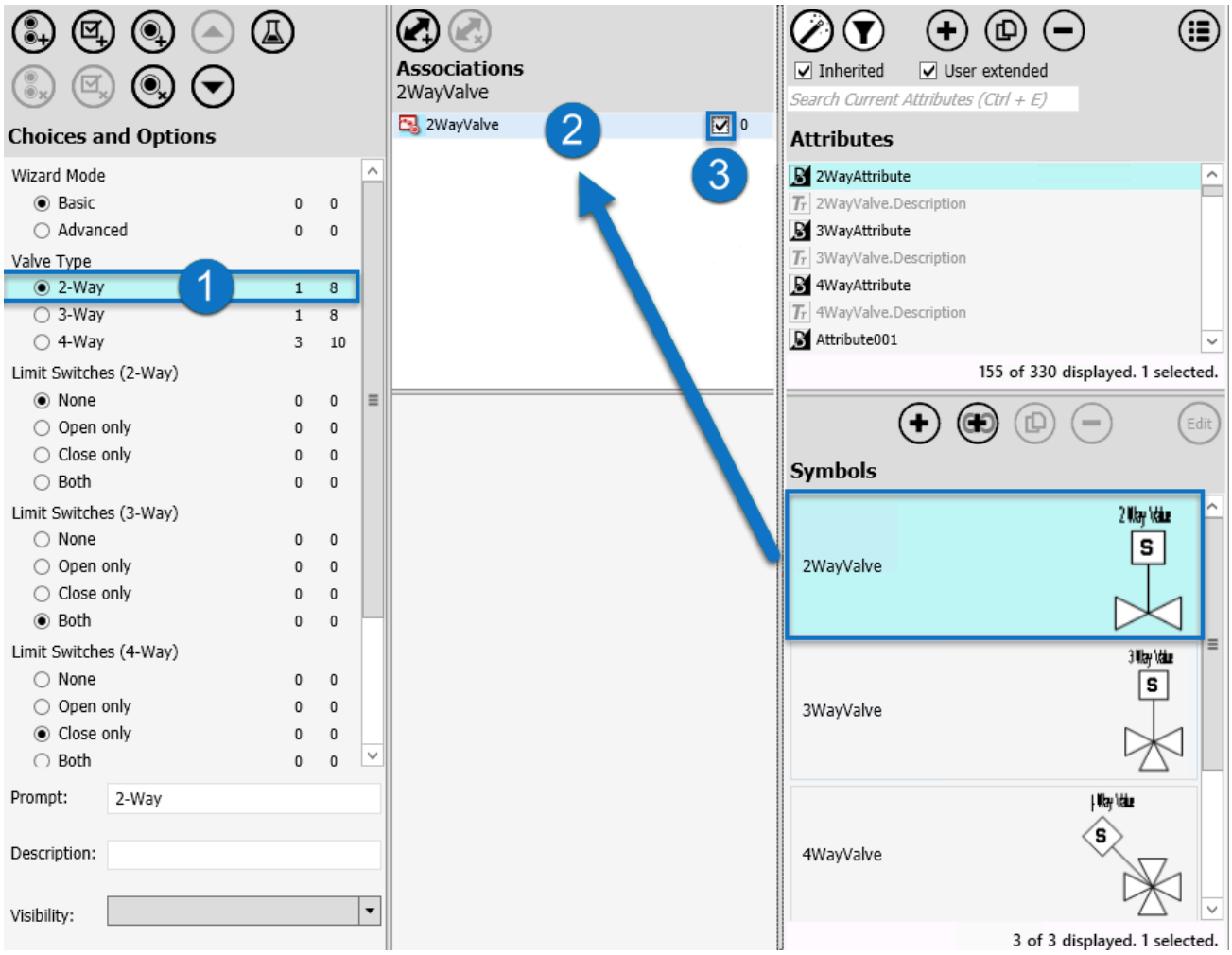
1. Create three new graphics. In this example, each graphic contains an embedded Situational Awareness Library graphic.
   - 2WayValve

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

- 3WayValve
- 4WayValve

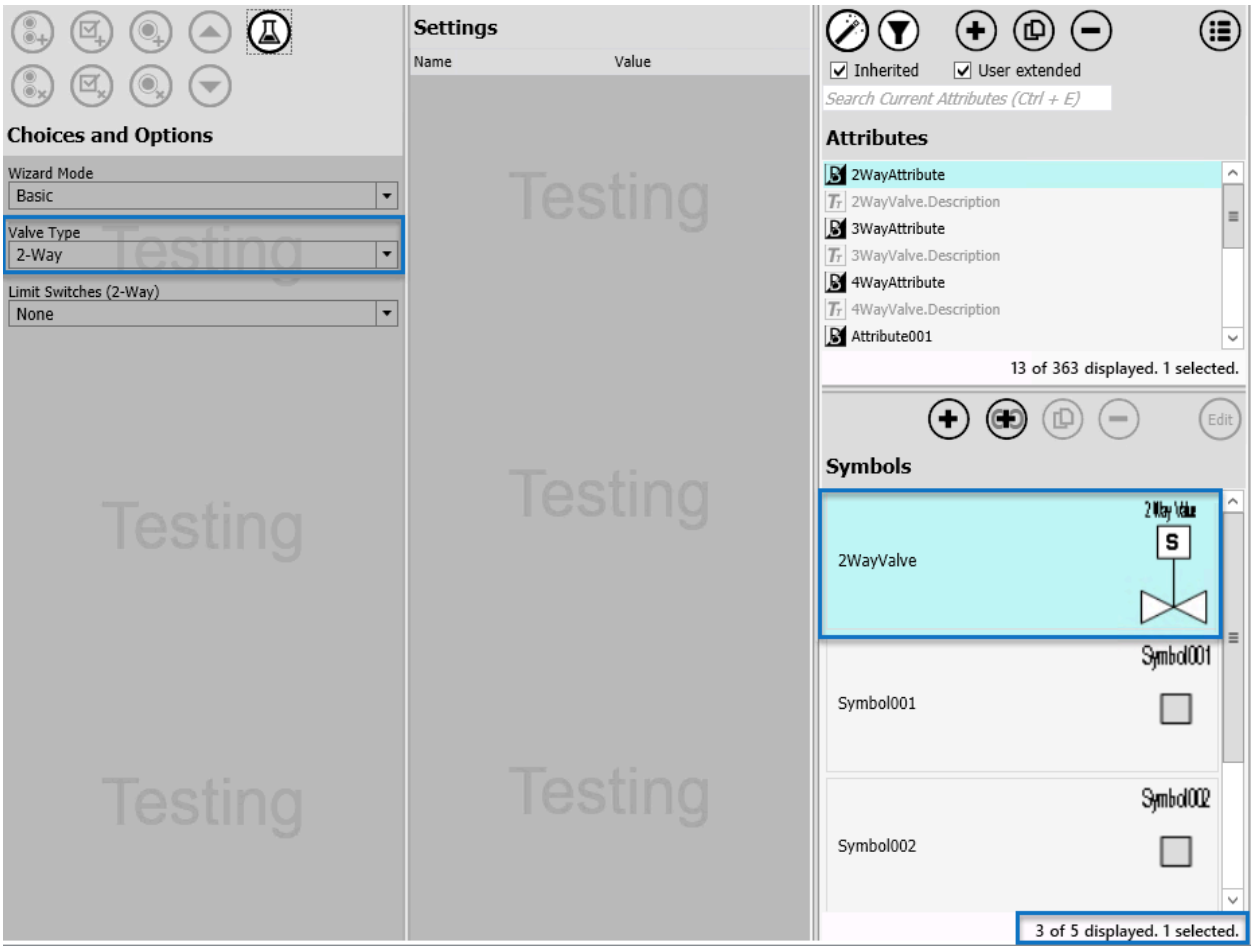
2. Highlight one of the Choices in the Valve Type Choice Group (1), then select the graphic with the corresponding name and drag it to the **Associations** pane (2). For example, with the 2-Way valve type Choice highlighted in the **Choices and Options** pane, drag and drop the 2WayValve graphic from the **Symbols** pane to the **Associations** pane.

   Make sure that the **Trimming** checkbox (3) is selected in the **Associations** pane; the default state for graphics is unselected. See Object wizard trimming for additional information.



3. Repeat for the remaining valve types (with trimming enabled).
4. Create two more graphics: Symbol001 and Symbol002. We will leave these graphics unassociated with any particular Choice or Option. Since they are not a part of the Object Wizard, these graphics will be included in all objects that are derived from our Object Wizard. It does not matter if trimming is enabled for these, since without any Object Wizard association, they are considered to be owned by the object.
5. To verify attribute behavior, select the **Test** button.
   - With the default Object Wizard selections made (in this case, 2-Way Valve), the 2WayValve graphic associated with this choice is shown, along with the two graphics not associated with any Wizard Choices or Options (Symbol001 and Symbol002). These can be seen in the **Symbols** pane. Note the message at the bottom of the **Symbols** pane that states "3 of 5 displayed." This is because the 2WayValve and 3WayValve graphics are hidden.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.



- Change the Valve Type to **3-Way.** This displays the 3WayValve graphic and hides the 2WayValve and 4WayValve graphics.
- Change the Valve Type to **4-Way.** This displays the 4WayValve graphic and hides the 2WayValve and 3WayValve graphics.

Note: When configuring an instance from an Object Wizard, unused attributes that have trimming enabled are not propagated to the instance. Only attributes not associated with the Object Wizard, and attributes with trimming enabled that evaluate as required, are present in the instance after the instance is saved. Trimming does not occur until after an object is saved, and any overrides a user enters for a trimmable attribute remain in memory until the user saves the object.

Propagation of object wizard changes to derived objects

Most changes made to an Object Wizard propagate through the object hierarchy to all objects derived from the Wizard. Examples of changes that are propagated include:

- Adding, editing, renaming, changing the order of, or deleting Choice Groups, Choices, or Options.
- Adding, removing, or modifying an association (attribute, graphic, or script) with a Choice or Option.
- Changing the default state (setting) of an Option.
- Changing the visibility setting of a Choice or Option.
- Changing the default value or visibility setting of an attribute or graphic associated with a Choice or Option.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

- Changing the visibility setting of a value in the **Settings** pane.

Important: With the exception of the first two items in the above list, changes DO NOT propagate to derived templates if the setting is marked as overridden in the derived template.

Object wizard behavior when a default choice is deleted

If you remove a default Choice from a Choice Group, the topmost Choice automatically becomes the new default, unless you explicitly select a different one. The new default propagates to child objects (templates and instances).

You cannot delete a Choice if there are only two Choices in the Choice Group; you must delete the entire Choice Group (a Choice Group, by definition, must contain at least two Choices). In this case, any associated attributes or graphics that were associated with the deleted Choice Group become a part of the object; they no longer have an association with the Object Wizard. Attributes and graphics that are not associated with the Object Wizard are not trimmed from derived instances.

Propagation limitations of object wizards

You cannot delete, rename, or change the order of Choice Groups, Choices, or Options in a derived Object Wizard. However, additional Choice Groups and Options can be added to a derived template with an existing Object Wizard. You can also selectively hide Wizard elements in the derived Object Wizard. Elements that are hidden will invoke a default behavior, or limit the Wizard consumer to choose between the remaining elements. For example, in a Choice Group with three Choices, you could hide Choice #2, leaving the Wizard consumer to choose between Choice #1 and Choice #3. See Object wizard visibility settings in a derived template for additional information.

See Object wizard derivation for additional information about derived objects.

Differences in propagation behavior of object wizard overrides between releases

This section describes several areas where differences in Object Wizard propagation exist between releases of Application Server, specifically, between Application Server 2023 and subsequent releases, as compared to prior releases. Changes were made to Application Server 2023 to provide a more predictable developer and user experience by modifying how Object Wizards configurations propagate to better align with the behavior that most users expect to see.

Deleting a Choice Group or Association in a Parent Template

The propagation of override values from a parent template with an Object Wizard to its child instances, as related to both dynamic attributes and extensions, can vary between releases. Prior to Application Server 2023, deleting a Choice Group, Option, or removing an attribute association in the Object Wizard would be propagated only partially to the child instances.

Note that overrides made within the instance, and not at the template level, are retained. This behavior remains constant between releases.

As an example of varying behavior, two different attributes are assigned to two different Choices within a Choice Group:

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

- Attribute A_1 is assigned to Choice 1
- Attribute A_2 is assigned to Choice 2.
- The values of both attributes are overriden in the template.
- The override values are propagated to a child instance.
- The Choice Group is deleted in the template.

As of Application Server 2023, both attributes in the child instance revert to their initial values, and the overrides are discarded. This is the behavior that most customers expect.

However, in prior releases, only attribute A_2 reverted to its original value, and attribute A_1 retained the override value. This behavior could cause some confusion, and has therefore been changed in Application Server 2023. Now, once the association or Choice Group containing the overriden value is removed from the parent template, all affected attributes in child instances revert to their values as defined within the template object in the Attribute Editor.

Resetting an attribute override in a parent template

A similar difference between releases exists when a dynamic attribute override is reset in the parent template. Continuing with the previous example:

- Attribute A_1 is assigned to Choice 1
- Attribute A_2 is assigned to Choice 2.
- Assign an override to the description for each Choice in the wizard override settings.
- After saving the parent template to propagate the overrides to the child instance, reset the descriptions in the wizard override settings (click the **X** in the Settings pane for both attributes).

As of Application Server 2023, both description attributes in the child instance are deleted (attribute A_1.description and A_2.description no longer exist in the child instance). Again, this is the behavior that most customers expect.

In prior releases, the description attributes remain in the child instances; only the value value was removed.

Changing IO mode in a parent template

All the settings in the derived templates are preserved, even if settings are changed in the parent template. This is existing behavior. However:

- If the child has IO Mode set as Read-Write, and
- If ‘Output every scan’ in the parent set to true

As of Application Server 2023, child instances have “reflect input to output” set to true. In previous version this was set to false.

Visibility changes in child objects

In some circumstances, the visibility of a Choice Group in derived objects could change unexpectedly if you override an attribute setting for one of the choices in the parent template. This change in visibility status can

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

occur in both child templates and instances. This unexpected behavior can occur if you derive a template from an Object Wizard, and then override a setting and change a default choice. While changing a default choice automatically hides the Choice Group, even making the Choice Group visible again by overriding the visibility setting may not correctly propagate to child objects (both templates and instances) if an attribute setting has also been overridden. As a result, the Choice Group in derived objects may not be visible, even though it is set to be visible in the parent template.

This behavior is corrected in Application Server 2023 to ensure that Choice Group visibility remains consistent between the parent and child objects.

Extend an inherited object wizard

You can extend the capabilities of a template derived from an Object Wizard by adding different types of content to it. Actions you can take include:

- Expand the existing Object Wizard by adding new Choice Groups or Options.

  Note: You cannot add Choices to an existing Choice Group.
- Associate attributes, graphics, and scripts with existing Choices and Options.
- Enter overrides for attribute and graphic settings.
- Change the default setting for Choice Groups.

  Note: When you change an inherited default Choice, the entire Choice Group is automatically hidden. However, you can turn visibility back on for the Choice Group. After that, you can selectively hide Choices within the Choice Group, if necessary.
- Selectively hide Choice Groups, Choices, and Options.
- Change the default default state of Options.

  Note: When you change an Option's default state, the Option is automatically hidden. However, you can turn visibility back on.

Add a choice group to a derived template

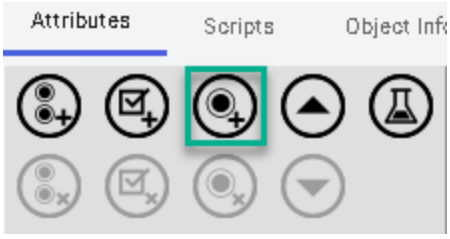
To add a new Choice Group to a derived template with an Object Wizard:

1. Select the **Add Choice Group** button to add a Choice Group.



2. Rename the Choice Group by entering the new name in the **Prompt** textbox.
3. You can change the relative position of the new Choice Group as needed by using the **Move Up** and **Move Down** buttons.
4. Select the **Add Choice** button to add additional Choices, if needed. Choice Groups include two Choices by default.

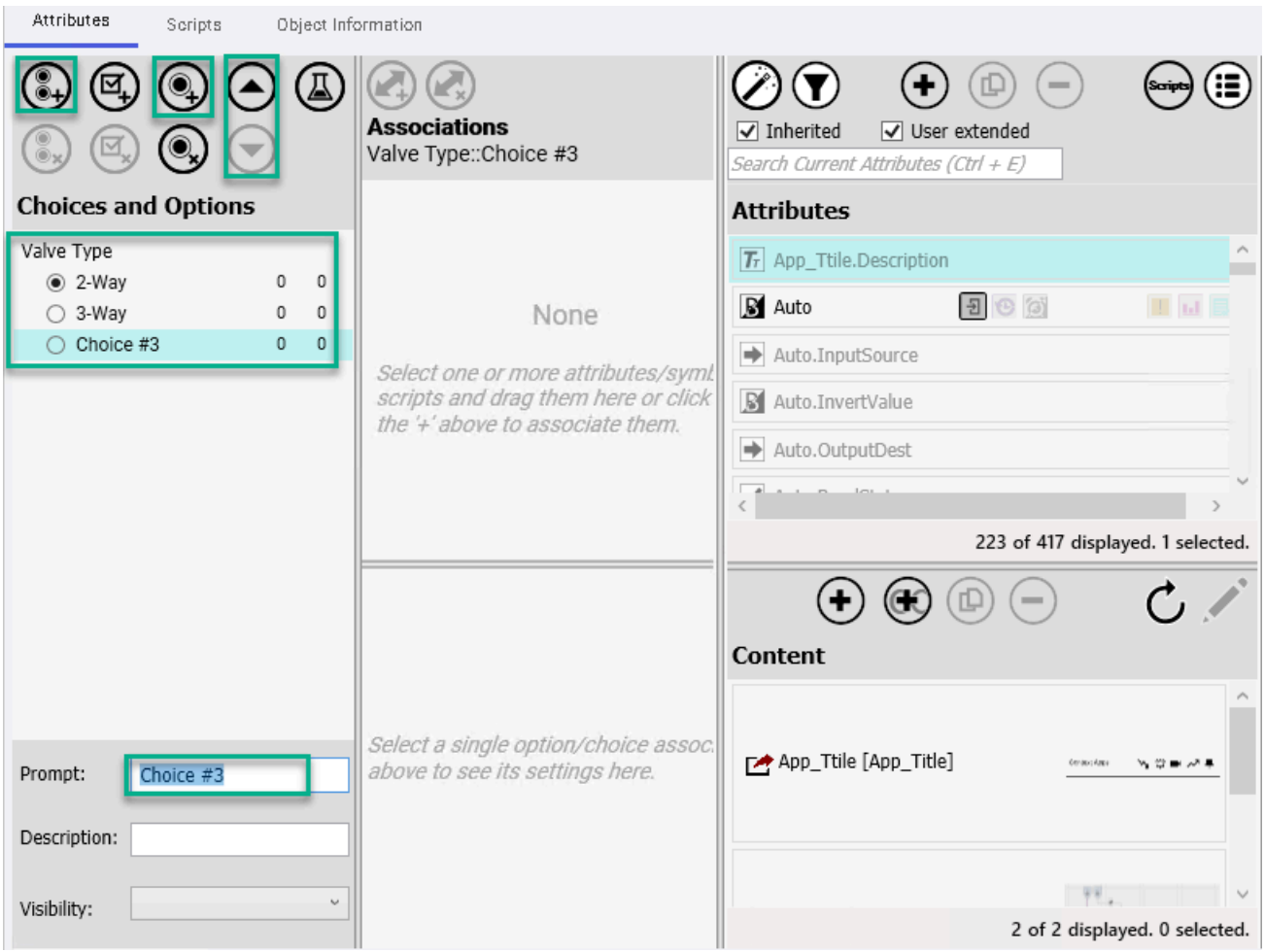
© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.



Choices within a Choice Group are mutually exclusive; one, and only one, Choice can be selected from a Choice Group. Choices are chosen by selecting the radio button next to the Choice. For example, an Object Wizard user must choose Choice #1 OR Choice #2 (but not both or none).

5. Rename the Choice by entering the new name in the **Prompt** textbox.

Note: You cannot add Choices to an existing Choice Group in a derived template.



Add an option to a derived template

To add a new Option to a derived template with an Object Wizard:

1. Select the **Add Option** button to add an Option.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

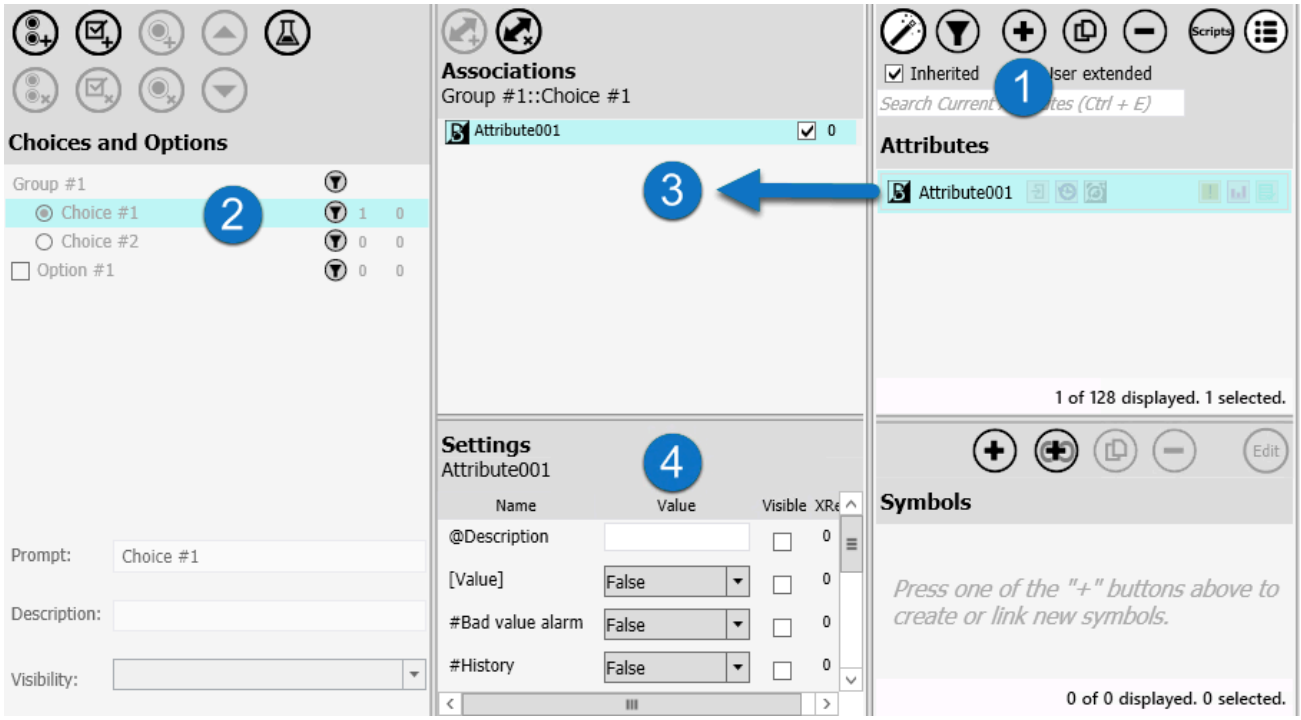


2. Rename the Option by entering the new name in the **Prompt** textbox.

Add an Association to an Inherited Object Wizard

To add an association (attribute, graphic, or script) to the inherited Object Wizard:

1. Create a new attribute, graphic or script (as needed) and configure any of its base settings, as needed.
2. Highlight a Choice or Option in the **Choices and Options** pane.
3. Add the new item in the **Associations** pane.
4. Enter overrides (for attribute and graphic settings) as needed for associated the Choice or Option.



5. Add and configure additional items and associate them with the Object Wizard, as needed.

Note: A Choice or Option does not have to have any associations. An individual attribute, graphic, or script can be associated with more than one Choice or Option. Graphic and attribute override settings can override the default values of an attribute; the overrides affect only the Choice or Option for which they are entered. For example, an attribute can have a value of 5 for one Choice, and a value of 10 for another.

Add a script to a derived template

As with attributes, you can also add new scripts to existing Choices or Options. The scripts can be pre-existing

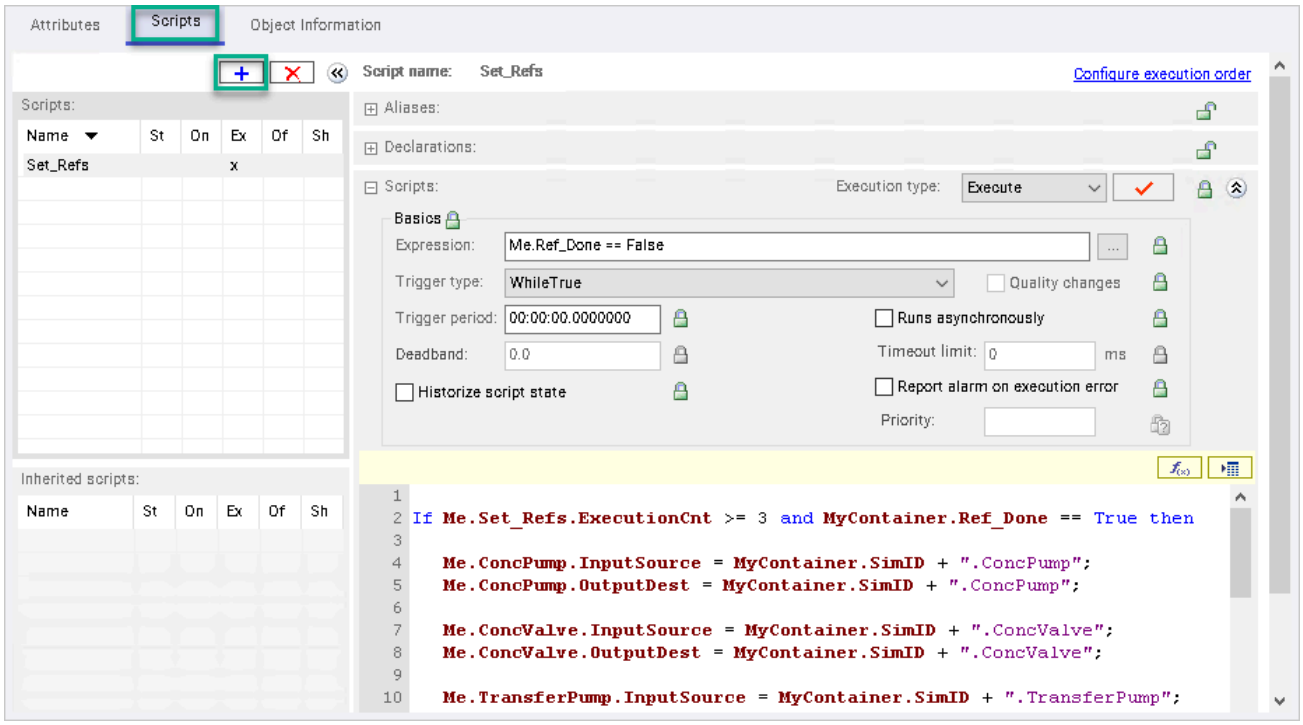
© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

within the object, or you can add new scripts to the object, and then associate them with the Object Wizard.

See the *Application Server Scripting Guide* for additional information about scripts.

To add an script association with the Object Wizard:

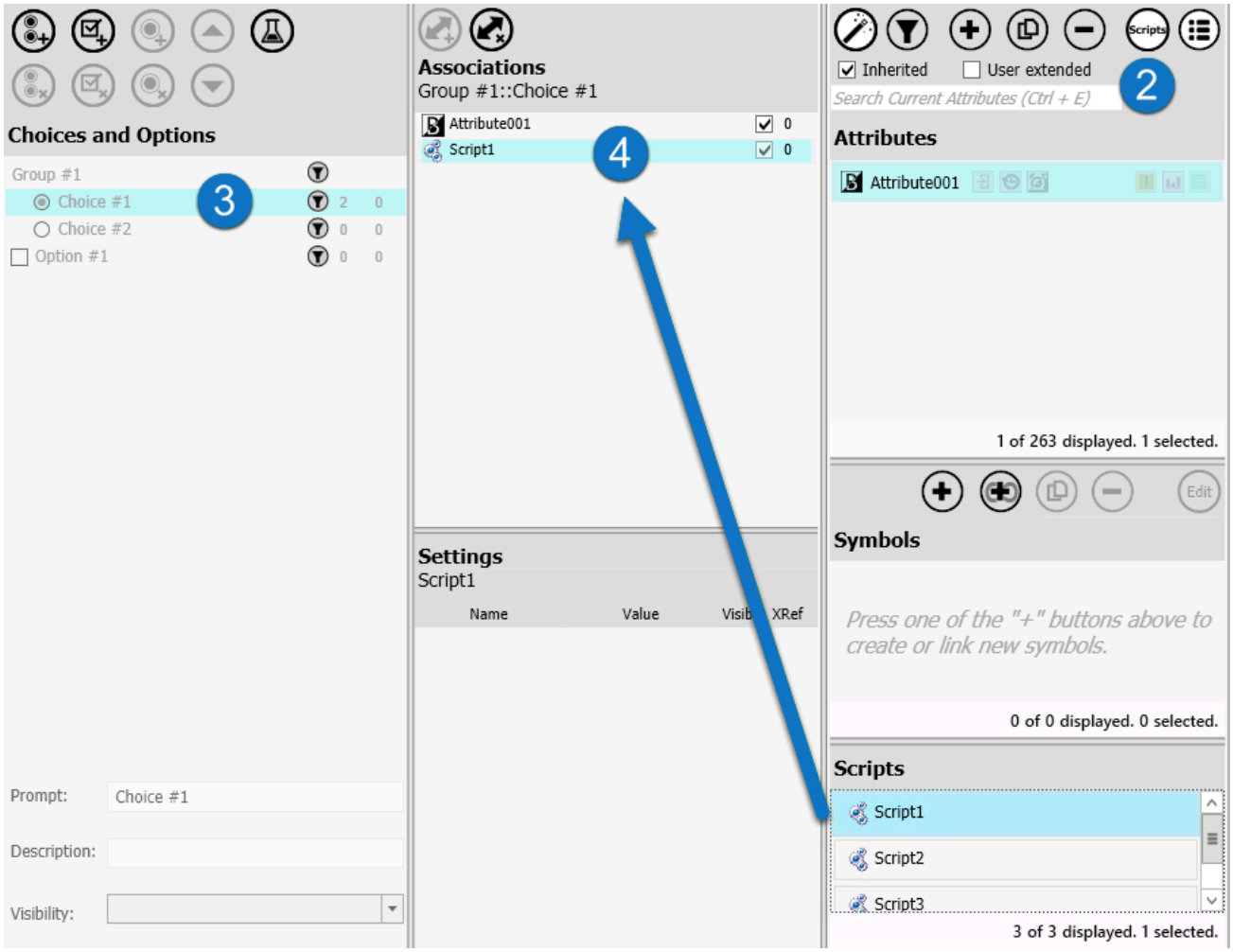
1. If you need to create a new script, select the **Scripts** tab and then select the **Add Script** button. See the Application Server User Guide, "Write and edit scripts," for additional information.



2. If necessary, select the Scripts button to show the **Scripts** pane.
3. Highlight a Choice or Option in the **Choices and Options** pane.
4. Drag and drop the script from the **Scripts** pane to the **Associations** pane.

   No additional configuration is needed for scripts. If you need the script to modify a script for another Choice or Option, create a copy of the script and change it.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.



5. Add additional scripts and associate them with the Object Wizard, or associate the same script with other Choices or Options.

Add content to a derived template

To extend or change the functionality of a derived template with an Object Wizard, you can add new graphics to existing Choices or Options.

You can create a new graphic for the object, and then associate the graphic with the Object Wizard. You can also link graphics from the Visualization folder to the object. See Link to shared content in the visualization folder for additional information.

See the Application Server User Guide, "Create and work with content," for additional information about graphics.

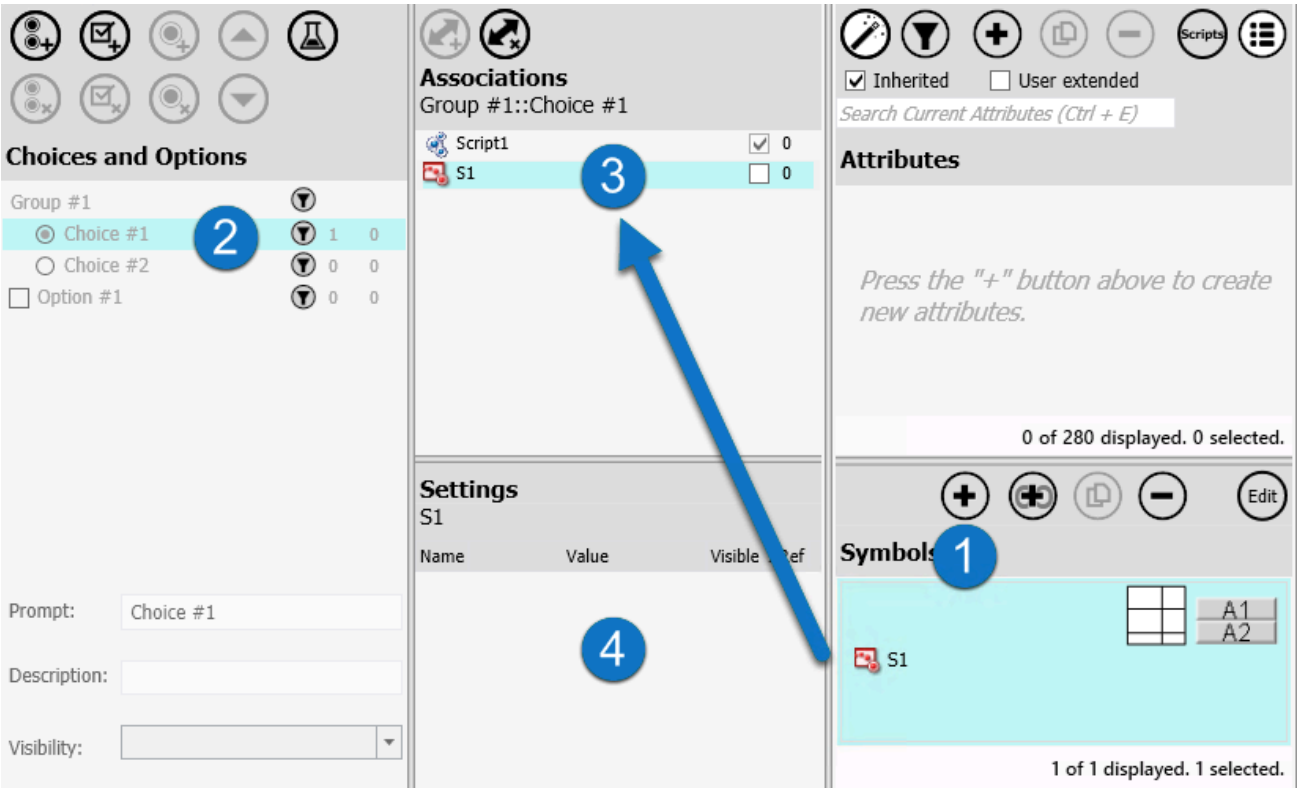
To associate new content with the Object Wizard:

1. Select the **Add** button to add new content to the object.

   Highlight a Choice or Option in the **Choices and Options** pane.
2. Drag and drop the graphic from the **Symbols** pane to the **Associations** pane.
3. If the graphic contains a Symbol Wizard or custom properties, modify the settings as needed for the Choice

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

or Option with which it is associated.



4. Add additional graphics and associate them with the Object Wizard, or associate the graphic with other Choices or Options.
5. If applicable, configure Symbol Wizard and custom property values for each graphic association in the **Settings** pane.

Note: A Choice or Option does not have to have a graphic associated with it. A graphic can be associated with more than one Choice or Option. You can override the default Symbol Wizard and custom property values; the overrides affect only the Choice or Option for which they are entered.

Link to a graphic in the visualization folder

To extend or change a derived Object Wizard's functionality, you can link a graphic from the Visualization folder with the Object Wizard. Once you have linked the graphic, you can associate it with the Object Wizard.

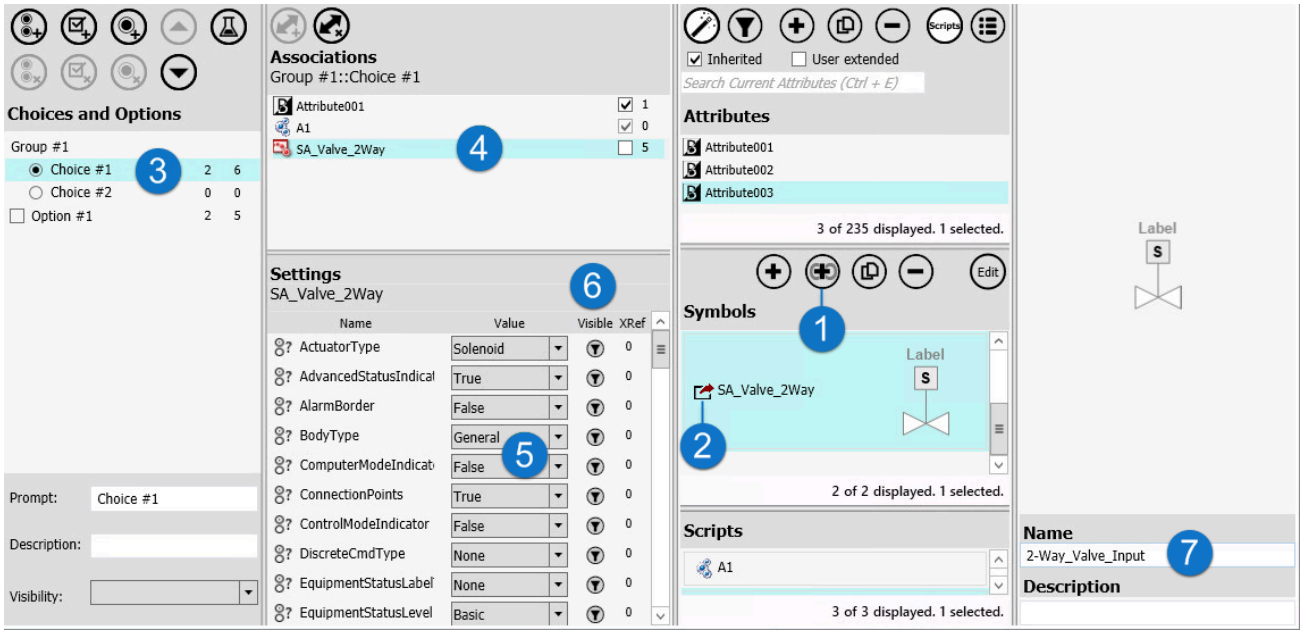
To link a graphic or other item from the Visualization folder to an Object Wizard:

1. In the **Content** pane, select the **Link Content** button **(1)** to link a graphic from the Visualization folder to the object.
2. The **Galaxy Browser** opens. Select a graphic or Symbol Wizard from the Visualization folder and select **OK**.
3. The graphic is added to the object. The icon **(2)** next to the graphic name indicates the graphic is linked to the object.
4. Highlight a Choice or Option in the **Choices and Options** pane **(3)**.
5. Drag the graphic from the **Content** pane to the **Associations** pane **(4)**.
6. If the graphic is a Symbol Wizard, modify the settings **(5)** as needed for the Choice or Option with which it is

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

associated.

7. Turn visibility of the Symbol Wizard settings **(6)** on or off, as needed. If the settings are visible, users can override your configuration settings.
8. In the **Details** pane **(7)**, edit the default name of the graphic as needed. You can also add a description (optional).



Note: A Choice or Option does not have to have a graphic associated with it. A graphic can be associated with more than one Choice or Option. You can override the default values of a Symbol Wizard; the overrides affect only the Choice or Option for which they are entered.

Graphics and Object Wizards

Two different methods are for associating graphics with Object Wizards.

- Add graphics into the template that you have created for that particular template. These graphics are owned by the object. While changes to the graphic in the object will propagate to child objects, non-related instances and templates that may use copies of the graphic will not be updated when the original graphic is edited.
- Link an external graphic to the template. Create a link from the template to an external graphic in the Visualization folder. Linked graphics can be shared between unrelated objects (objects that are not part of the same derivation hierarchy). Changes to the graphic will be reflected in each template and instance that link to the Visualization folder graphic.

Example 1: Add a graphic to the object

To add a custom graphic to a template:

1. In the **Content** pane, select the **Add** button.
2. Rename the graphic in the **Details** pane of the **Object Editor**. You can also add a description of the graphic

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

(optional).

3. Select the **Edit Content** button. The **Graphic Editor** will open in a new window.
4. Select the **Graphic Editor** window and create the graphic to add to the template.
5. Save the graphic and close the **Graphic Editor**.
6. To associate the graphic with an Object Wizard Choice or Option, see Associate content with a choice or option.

Example 2: Link to a graphic in the visualization folder

To link a graphic in the visualization folder to a template:

1. In the **Content** pane, select the **Link Content** button.
2. The **Galaxy Browser** opens and displays the Visualization folder.
3. Select the Visualization folder graphic from the navigation pane that you want to link to the template.
4. Rename the graphic in the **Details pane** of the **Object Editor**. This name applies only to the current object. You can also add a description of the graphic (optional).
5. Select the **Edit Content** button. The **Graphic Editor** will open in a new window.
6. Select the **Graphic Editor** window and edit the graphic as needed.
7. Save the graphic and close the **Graphic Editor**. When you save the Visualization folder graphic, changes will be reflected in every object to which the graphic is linked, including child objects.
8. To associate the graphic with an Object Wizard Choice or Option, see Associate content with a choice or option.

Configure instances

You can use the IDE to configure instances from a template with an Object Wizard, or you can graphically create instances using an editor that is started from the Graphic Editor. If you are using AVEVA OMI for your runtime environment, you can also access the editor through the ViewApp Editor or Layout Editor. See the AVEVA OMI online help for more information about these editors.

- From the IDE, the Object Wizard presents a series of prompts or questions, from which you select an answer to build a representation of a piece of field equipment, such as a pump, valve, PLC, etc.
- If you are using the Graphic Editor or one of the AVEVA OMI editors, select a graphic from the **Toolbox** tab and drag it onto an overview graphic. See Create a new instance graphically for more information.

Configure an instance with the IDE

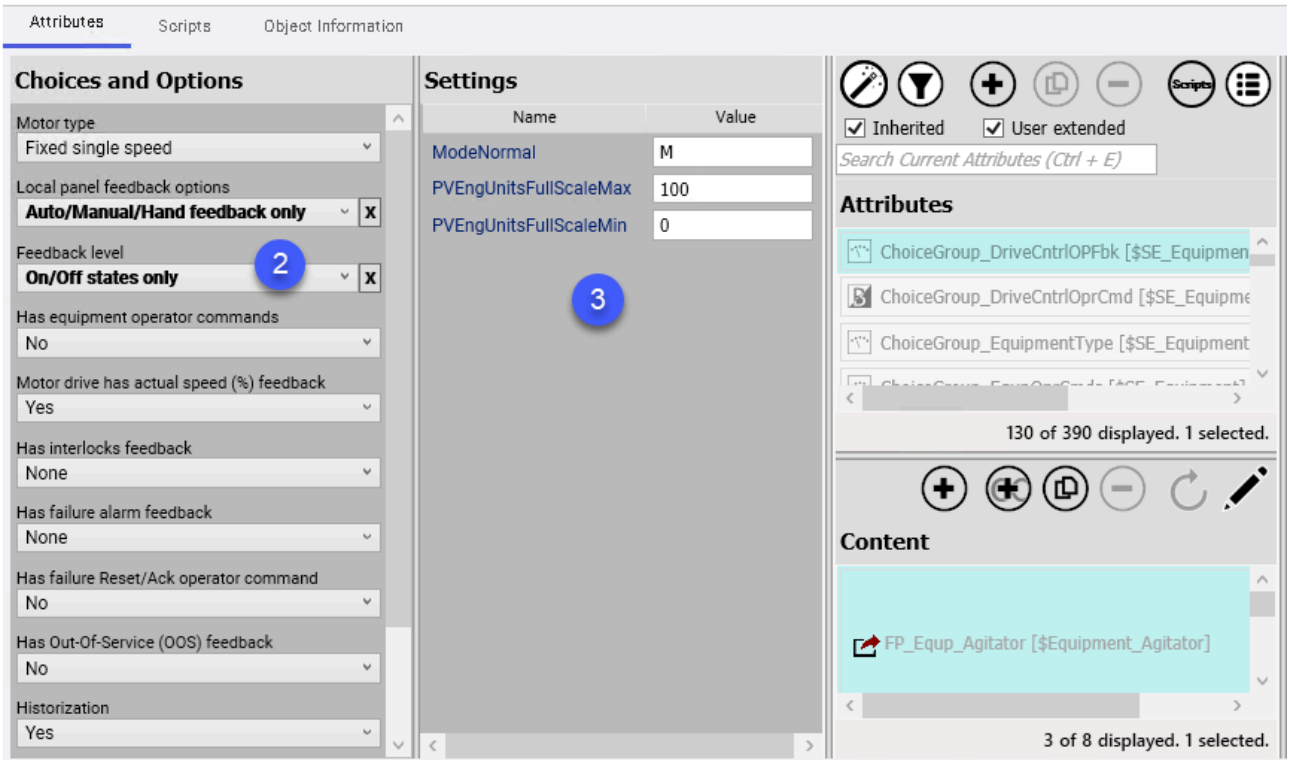
To configure an instance using an Object Wizard in the IDE:

1. In the IDE, open the instance you need to configure.
   - To create and configure a new instance:
     Select the template you want to use, and derive an instance from it. Then, open the instance. See Create an instance for additional information.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

- To change the configuration of an existing instance:
  Select the instance you want to use and open it. See Create an instance for additional information.

2. Start at the top of the **Choices and Options pane**. Either select a Choice from the first Choice Group, or select/deselect the first Option as needed to match the physical asset you are modeling. Depending on the selections you make, attributes associated with the Choice or Option may appear in the **Attributes pane**.
   - If you change from the default Choice, the override is indicated by an **X** and the override is shown in **bold** type.
   - If you change the default state of an Option (selected or not selected), the override is indicated by an **X**.
   - You can restore a default setting by clearing the **X**.
3. If an associated attribute contains configurable settings, these are listed in the **Settings pane**. Overrides are also indicated by an **X** next to the changed setting and are indicated by **bold** type.



4. Continue sequentially down the list of Choices and Options.
5. Save and close the instance, and then deploy it.

View and configure graphics

To view and configure graphics associated with an Object Wizard:

1. Select the graphic from the **Content** pane.
2. Open the graphic by selecting the **Edit Content** icon or by double clicking the graphic.
3. The **Graphic Editor** opens.
   - If the graphic is inherited, it opens as read-only.
   - If the graphic is a Visualization folder template, it opens as read-only.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

- You can only configure graphics that have been created for the object (object-owned).

4. Configure Symbol Wizard settings and custom property settings as needed.
5. Save and close the **Graphic Editor**.

See the Application Server User Guide, "Create and work with content," for additional information about working with graphics.

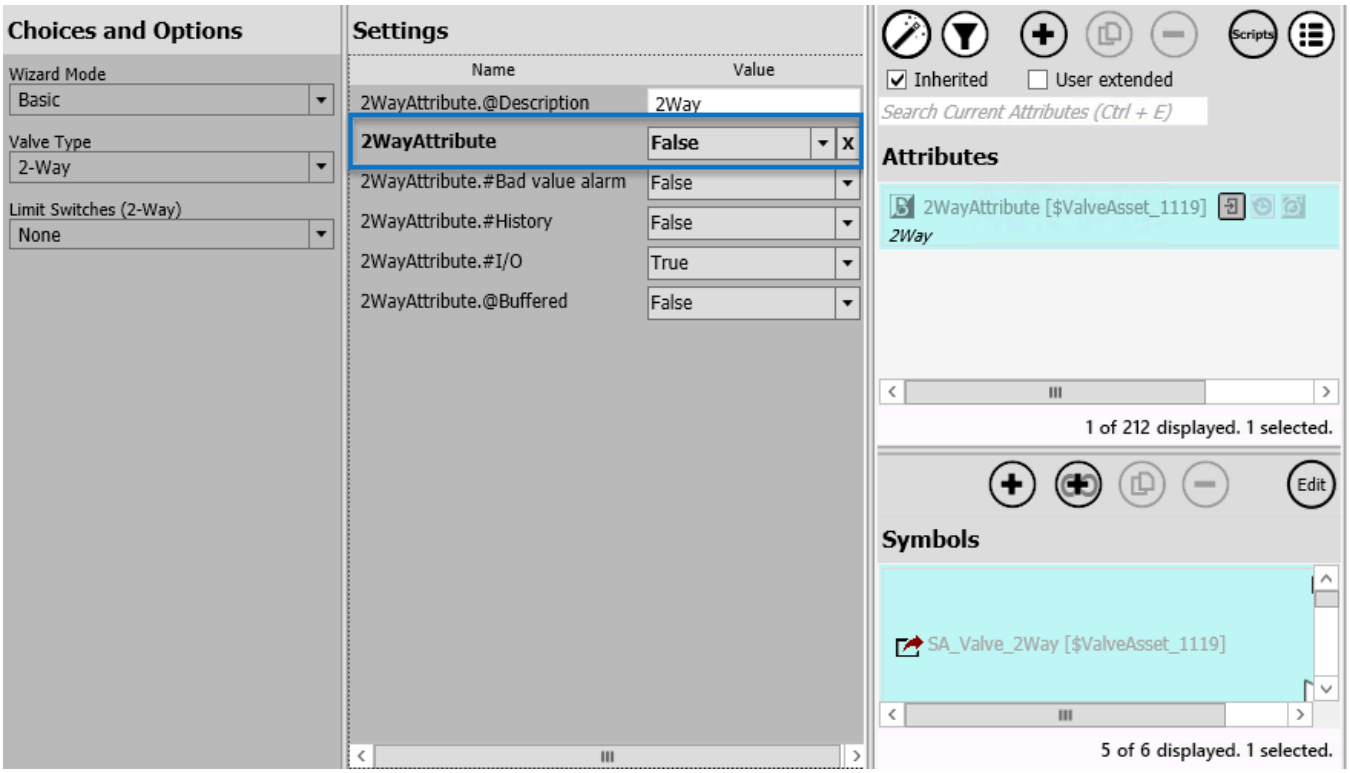
Guidelines for configuring instances with an object wizard

Keep in mind the following behaviors and guidelines when you use an Object Wizard to configure instances:

- Start at the top, with the first item of the Object Wizard.
- The answers you provide to each prompt may cause subsequent prompts to change. Therefore, if you skip selecting a choice or option, the subsequent selections you make could be changed or removed when you return to select a skipped step.
- Some prompts could be added or removed, based on the answers provided to previous prompts.
- If an attribute or graphic is associated with a selected Choice or Option, you may have the option to override certain settings.

Override default settings

When you override a default attribute or graphic setting in the Object Wizard, the override is marked by an X, and the setting and its description are displayed in bold.



Settings can be different types, including Boolean, integer, string, etc. Settings can be used to turn on or off a

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

feature for the object, such as I/O, Historization, or Alarms. Changing one setting can cause other settings to be shown or hidden.

Create a new instance graphically

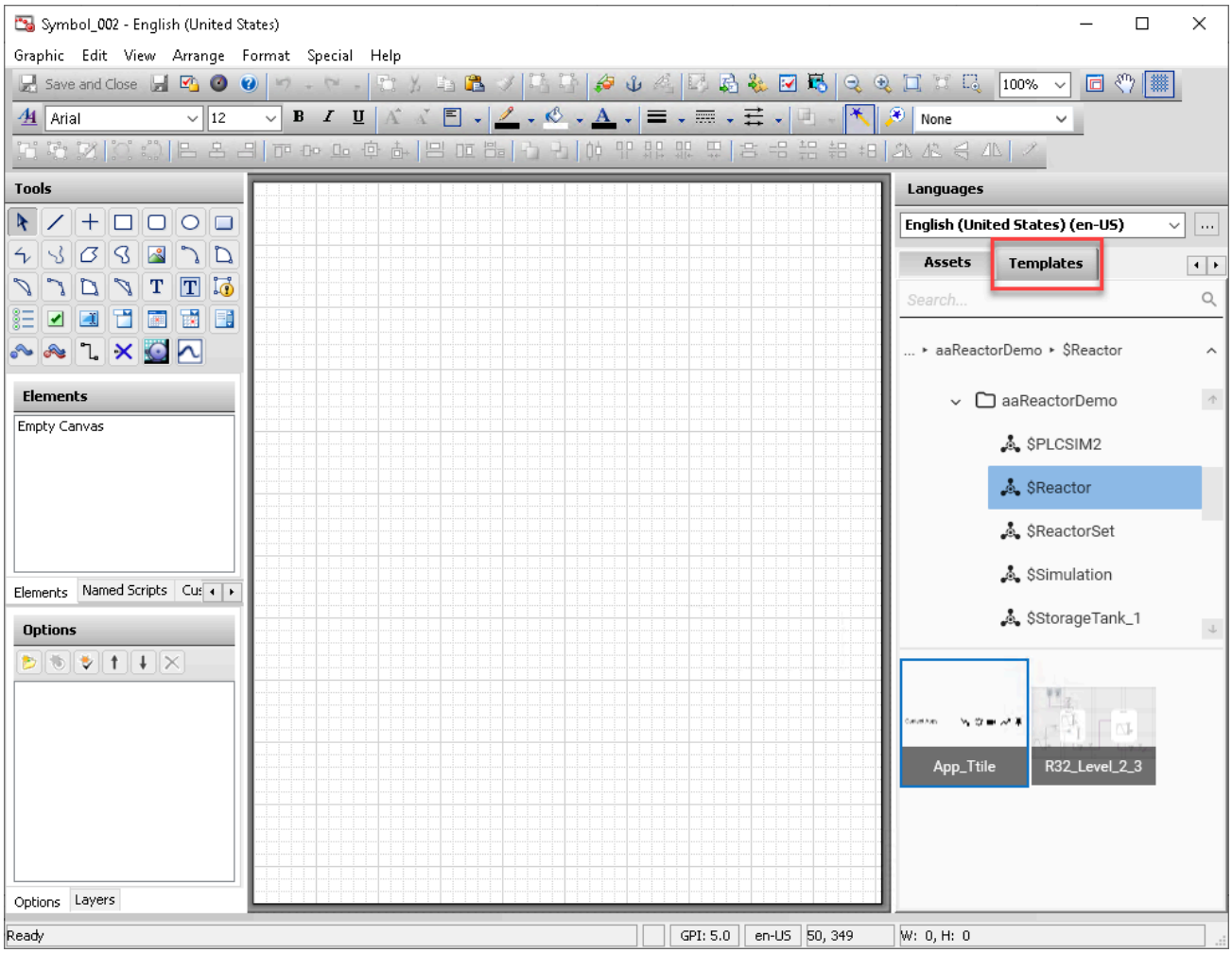
You can quickly and easily add and configure new instances from templates that contains an **Object Wizard** and *at least one graphic,* simply dragging a thumbnail image from the template onto the Graphic Editor canvas. Dropping the graphic thumbnail onto the canvas opens the **Configure New Asset** editor, where you can configure the Object Wizard settings. You can also configure Symbol Wizard and custom property settings, if applicable to the graphic you select. With this editor, you do not need to use the **System Platform IDE** to configure object properties, or spend a lot of time configuring graphic properties in the **Graphic Editor**.

Note: You can add an asset configured from an Object Wizard with a linked Symbol Wizard directly to an AVEVA OMI pane. However, InTouch HMI does not support adding and configuring an asset in an InTouch HMI window that has been derived from an Object Wizard with a linked Symbol Wizard. If you are using InTouch HMI, you must instead embed the symbol in an overview symbol. InTouch HMI does not support directly linking an Object Wizard to graphics in the Visualization folder that use a Symbol Wizard, such as those in the Situational Awareness Library.

To start the **Configure New Asset** editor:

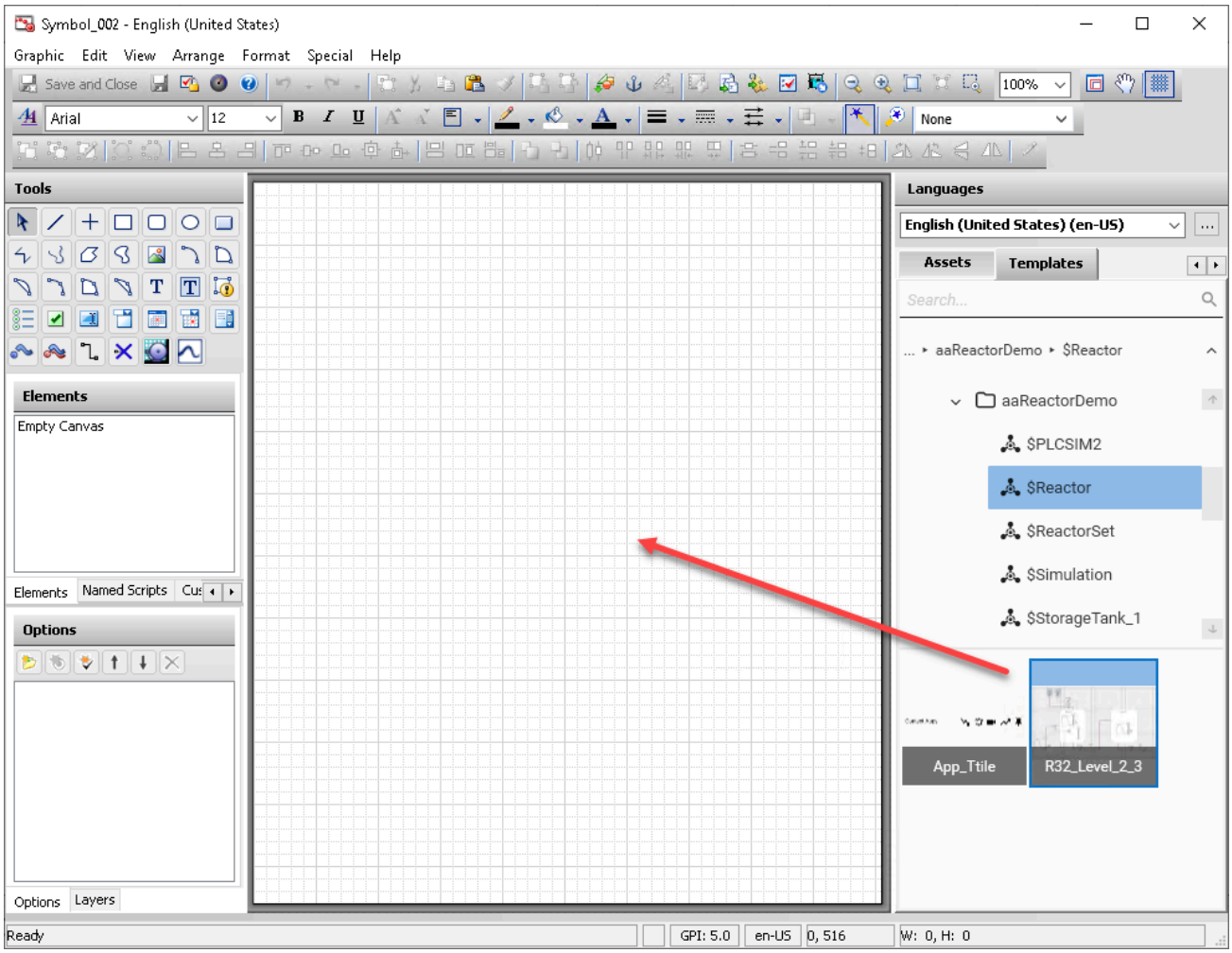
1. Select a new or existing symbol in the **Visualization** folder. The **Graphic Editor** opens.
2. If necessary, use the arrow keys until the **Templates** tab is shown, then select it.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

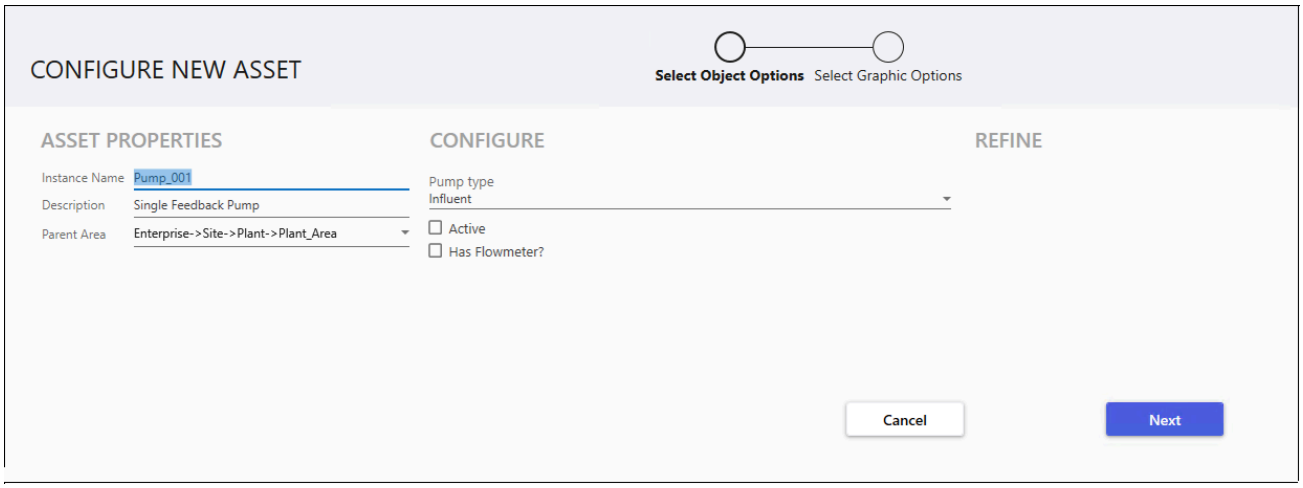


3. Navigate to the template you want to use. The template you select must contain at least one graphic.
4. Select the graphic to use and drag it onto the canvas.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.



5. As soon as you drop the image onto the canvas, the **Configure New Asset** editor opens to the **Select Object Options** page.



Under **Asset Properties** (left column) edit the following entries:

- **Instance Name** (name of the new asset). The Instance Name is the same as the object tag name. Each Instance Name must be unique within the selected **Parent Area**.
- **Description** (a description of the new instance).

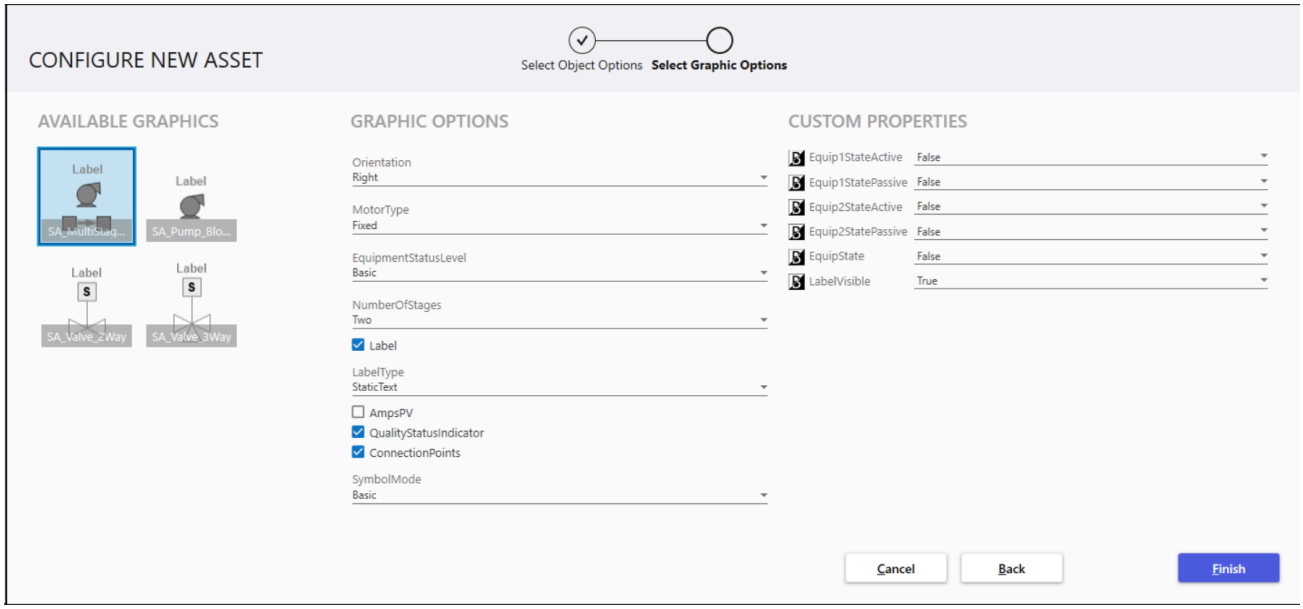
© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

- **Parent Area** (select the appropriate parent area for the new instance).
- Configure the **Object Options** (center column).
- Configure any properties listed under **Refine** (right column—there are none shown in the example above), then click **Next.**

  Object options determine which graphic options can be used. Only the applicable graphic (or graphics) will be available. For example, if an object includes options that allow it to be configured as a pump or as a conveyor belt, you will not be able to add a pump graphic if you have selected a conveyor belt asset on the **Object Options** page.

  See VisualBuild Object options for additional information.

6. The **Select Graphic Options** page opens. If more than one graphic is displayed, select the one to use, then configure its graphic options. Click **Finish** when you are done.



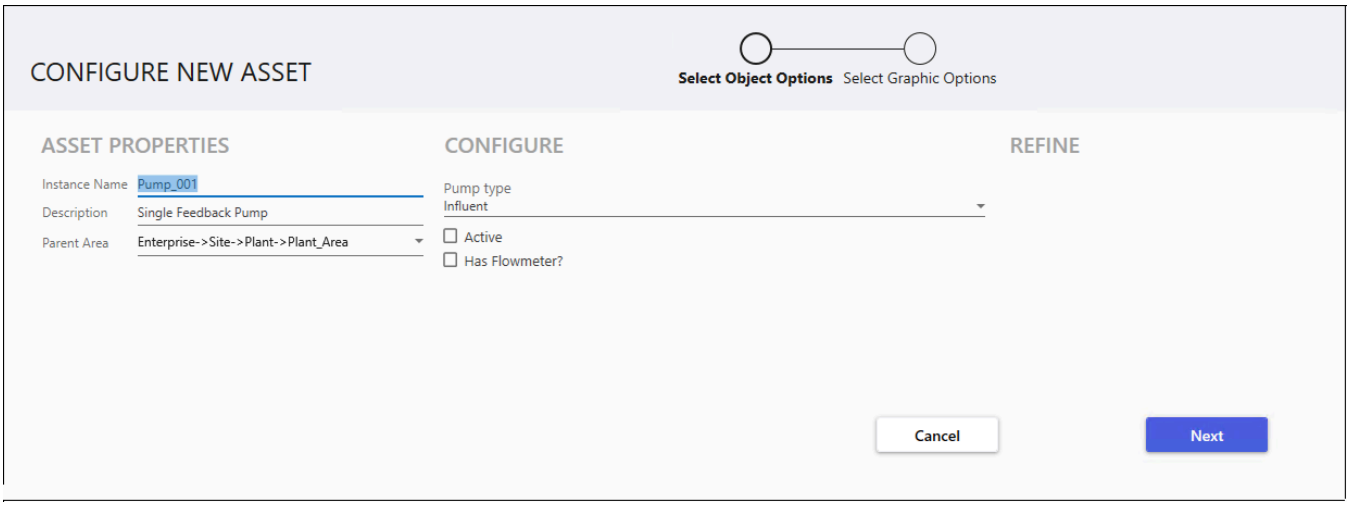
Graphics options are Symbol Wizard options. Select the Symbol Wizard options applicable to the asset you are adding to the overview symbol. Once you select the graphic options, you can configure any applicable custom properties.

7. After you click Finish, the **Configure New Asset** editor closes. Adjust the size and position of the graphic on the **Graphic Editor** canvas as needed. Add additional symbols and graphics as needed.
8. When you have finished adding symbols and other edits, select **Save and Close** to exit from the **Graphic Editor**.

VisualBuild Object options

The **Select Object Options** page consists of three columns for **Asset Properties**, configuration (**Configure**), and settings (**Refine**).

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.



The **Asset Properties** column contains settings for:

- **Instance name** (what the instance will be named within the Galaxy). The instance name is the object's tagname, and can be up to 32 alphanumeric characters, with at least one letter and no spaces. $ cannot be used as the first character.
- **Description** (of the instance). There are no restrictions on the length or characters in the description.
- **Parent area**: This is a drop-down selection that lets you move the instance to the appropriate area of the Galaxy.

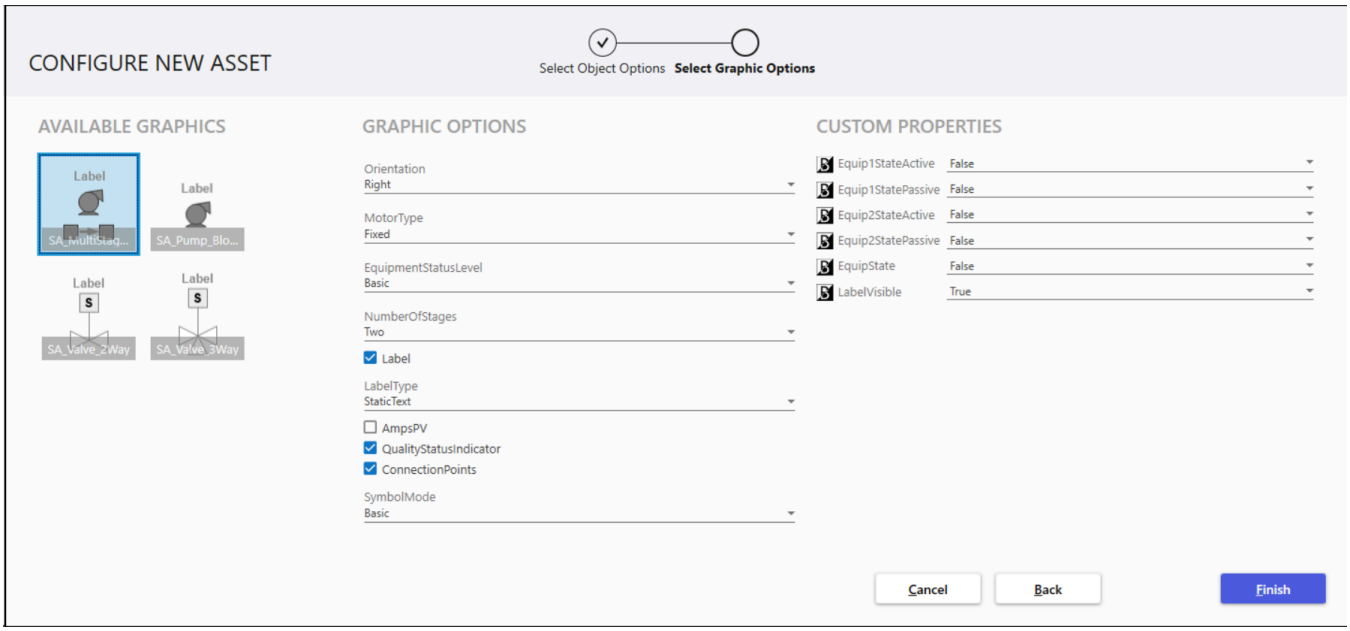
The **Configure** column contains object property selections, arranged as a series of prompts with drop-down answers. Start at the top and work down to the end. As you make selections, additional prompts may appear in the Configure area (always below the current prompt).

The **Refine** column contains additional settings that you can set. Typically, the settings will include things like upper and lower limits for scaling and alarms, on/off settings, or other items that further define settings that were selected in the **Configure** column. Some configurations do not contain any settings under the **Refine** column.

VisualBuild graphic options

The **Select Graphic Options** page consists of three columns for **Available Graphics**, **Graphic Options**, and **Custom Properties**.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.



The **Available Graphics** column displays the graphic or graphics that are associated with the object as configured on the **Select Object Options** page. If more than one graphic is displayed, select the one you want to use. Only one graphic can be selected for each asset you configure.

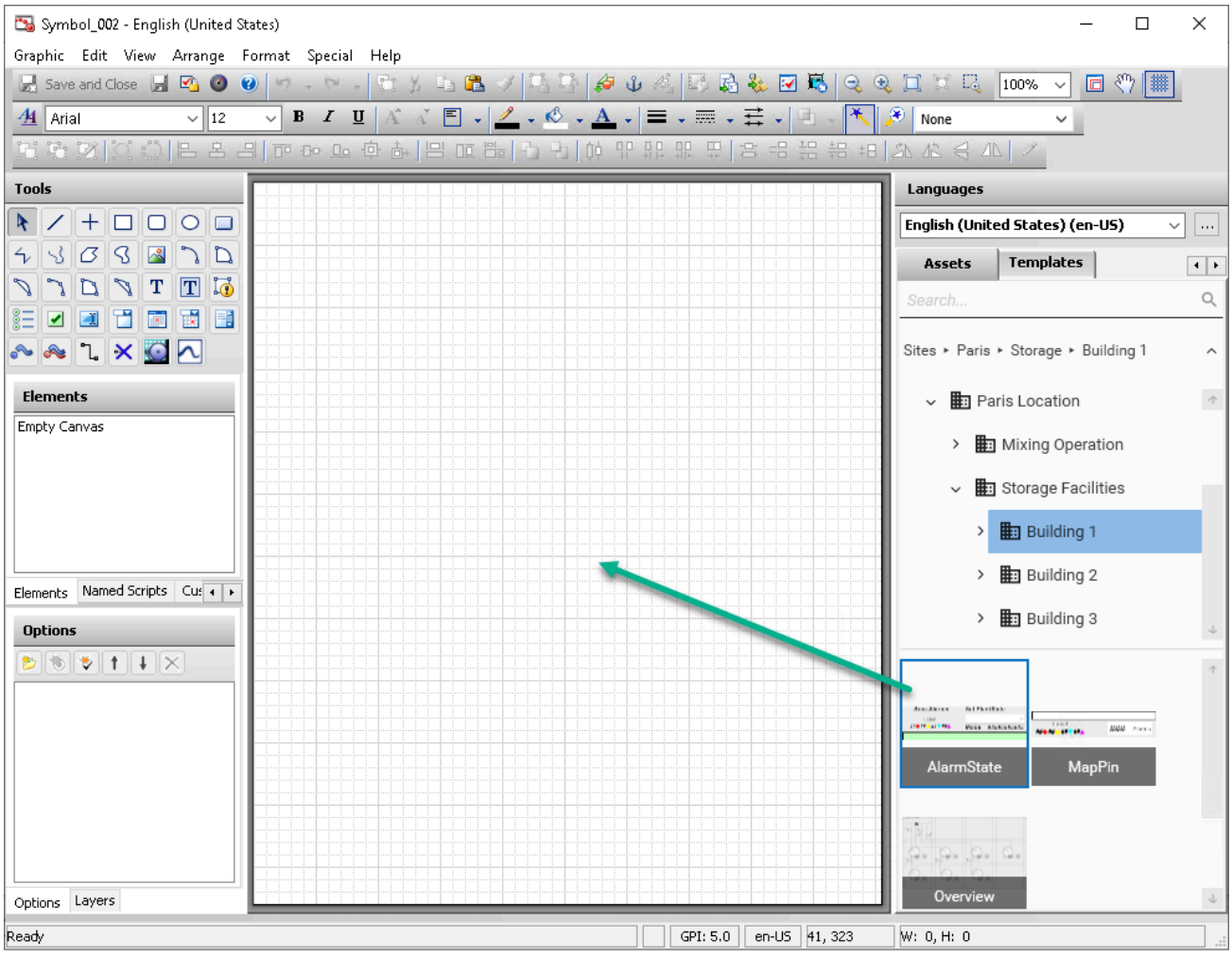
The **Graphic Options** column contains Symbol Wizard configuration selections that apply to the graphic. Typically, these are used to control things like orientation of the graphic, whether or not it has a visible label, to add alarm borders, etc.

The **Custom Properties** column contains additional refinements to set parameters for the graphic. This includes items like alarm limits and priority levels.

Add a configured asset to an overview graphic

1. To add a graphic from a configured instance, select the **Assets** tab in the **Graphic Editor.**

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.



2. Navigate to the asset you want to add to the open graphic and drag it onto the canvas.
3. Adjust the size and position of the graphic on the **Graphic Editor** canvas as needed. Add additional graphics as needed.
4. When you have finished adding graphics and other edits, select **Save and Close** to exit from the **Graphic Editor**.

Note: InTouch HMI does not support adding a graphic derived from a Symbol Wizard linked to an object wizard directly onto an InTouch window. The graphic must be contained in an overview graphic.

Configure the OPCClient object for data integration

The OPCClient Object should be used instead of legacy Data Integration (DI) objects, such as ABCIP, SIDirect, MBTCP and similar DI objects. Two changes were made to the OPCClient object for the System Platform 2023 release to support the deprecation of DINetwork Objects. A new attribute, **HierarchyPath**, and a default scan group have been added to the OPCClient object. Together, these changes let you easily replace the complex hierarchies sometimes required when using DINetwork Objects with a single entry in the OPCClient object configuration, without the need to reconfigure IO references.

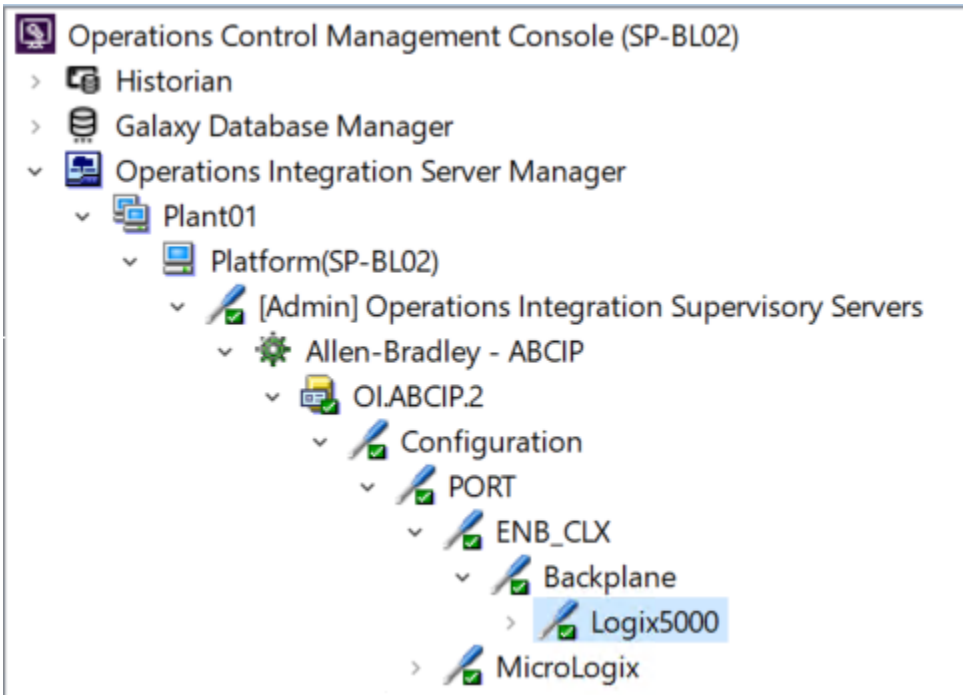
© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Convert existing DINetwork object references

The following procedure allows you to transition from DINetwork Objects to OPCClient objects, without the need to modify existing IO references.

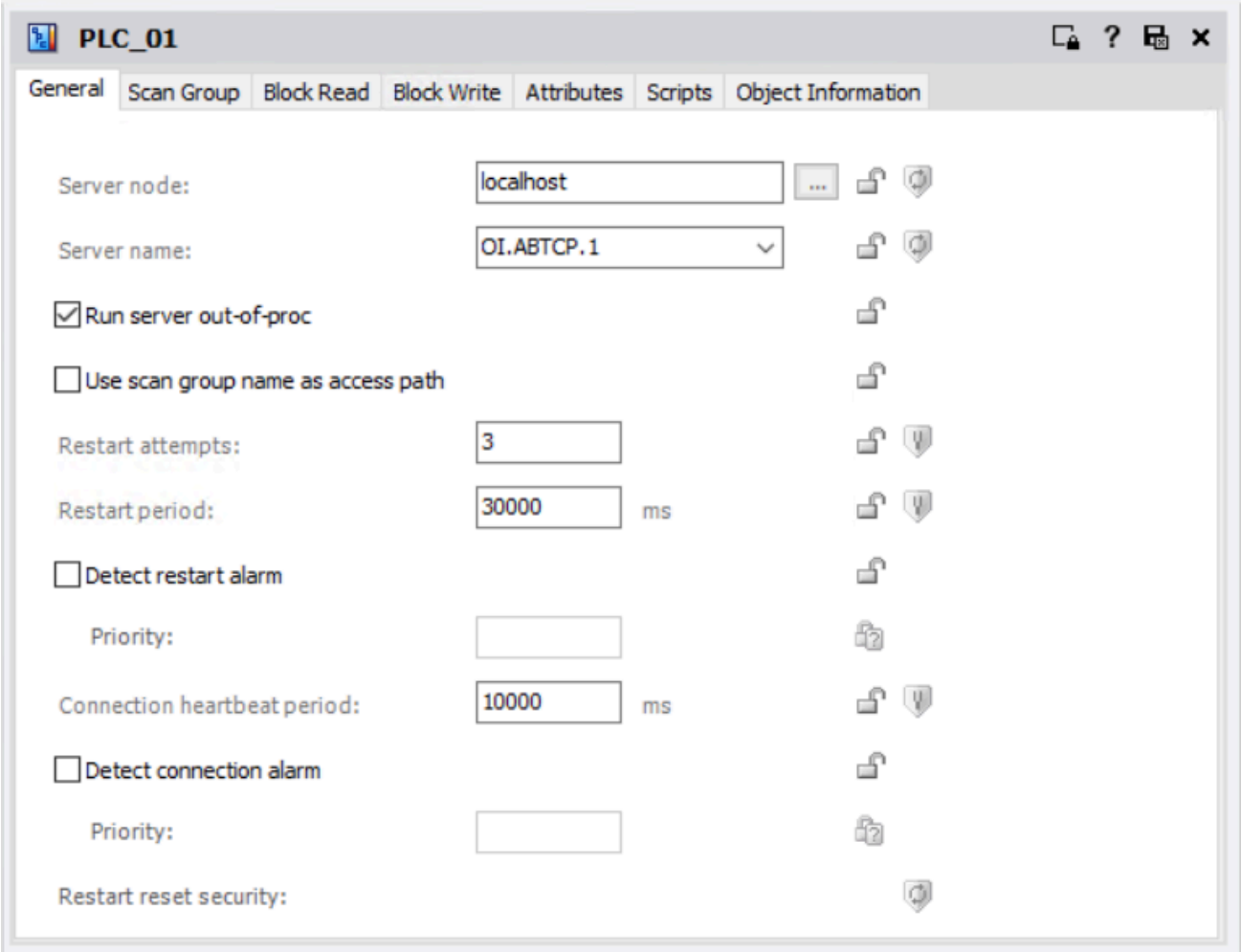
Note: This procedure assumes that the appropriate communication drivers are already installed in Application Server.

Before starting the procedure, you can view the hierarchy path by expanding the Operations Integration Server Manager in the **Operations Control Management Console**, and use it a reference to reconstruct the hierarchy path. Note that the hierarchy path can only be viewed when the Galaxy is deployed.



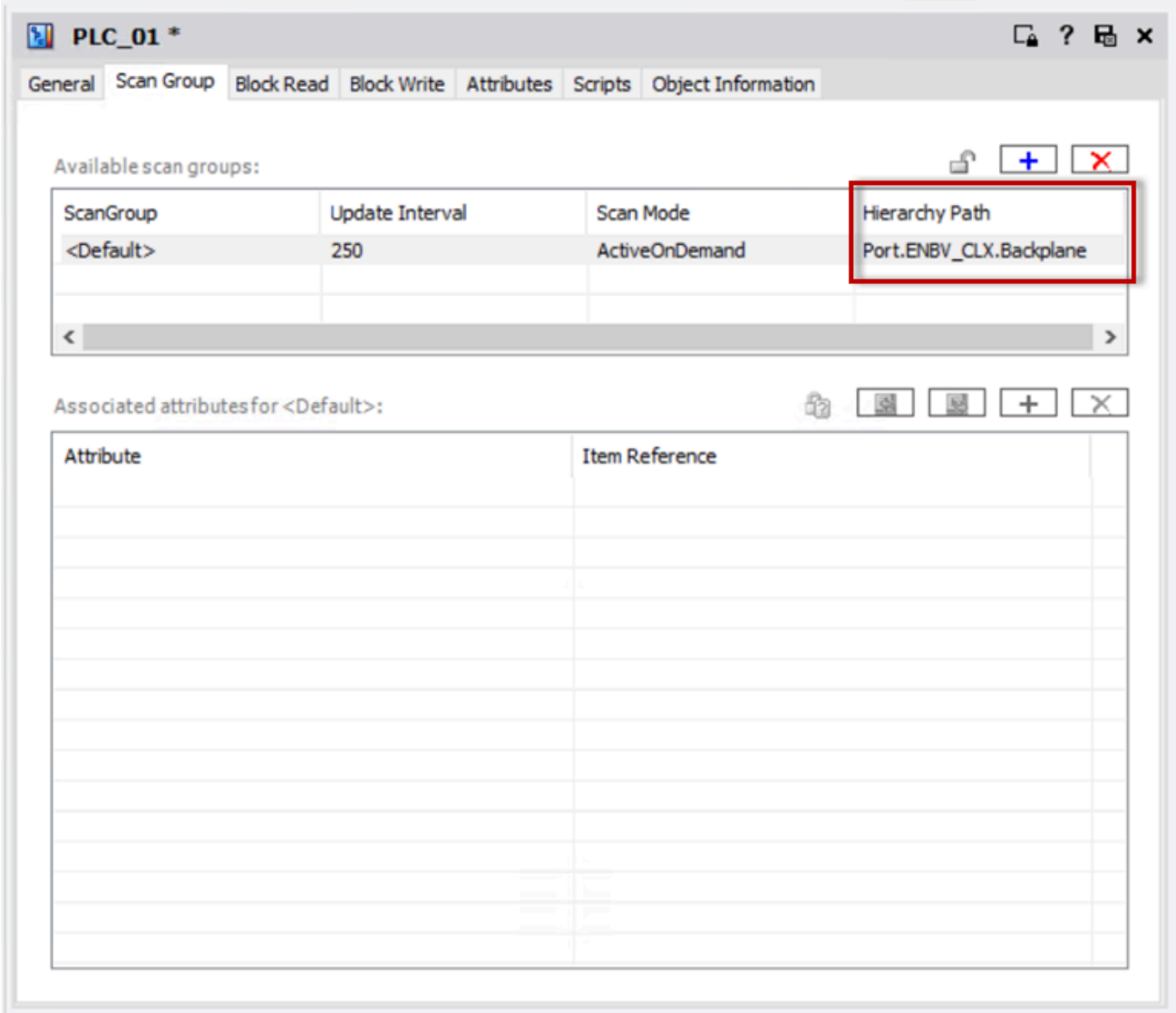
1. In the IDE, undeploy the DINetwork object. In the examples below, the DINetwork object name is PLC_01.
2. Rename the existing DINetwork object to <*ObjectName*>_OLD, for example PLC_01_OLD.
3. Create a new OPCClient object and give it the same name as the existing DINetwork object. In this example, the new object name is PLC_01.
4. Open the OPCClient object and configure the **General** tab with the same **Server Node** and **Server Name** as the existing DINetwork object. Edit other parameters as needed.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.



5. Move to the **Scan Group** tab.
   - A <Default> scan group appears automatically.
   - If additional scan groups are defined in the DINetwork object, add them now.
6. Set the **Update Interval** and **Scan Mode** for the <Default> scan group to match the configuration of the DINetwork object.
7. If necessary, create new Scan Groups.
8. Enter the **Hierarchy Path** for the PLC item. **Hierarchy Path** is used internally to recreate the full path to IO items so you do not have to alter existing IO references. The Hierarchy Path is required for each scan group. There is no need to include the scan group name in the Hierarchy Path. References will be built automatically to include the scan group.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.



9. When you are done editing, save and deploy the OPCClient object. Your existing IO references should begin working immediately. You can verify this in the Object Viewer.

Why you should use OPCClient objects instead of legacy DINetwork objects

As newer versions of PLCs have been released, many of the deprecated DINetwork Objects have not received the necessary software enhancements to allow them to function correctly with the latest firmware versions. To simplify the migration from DINetwork Objects in existing galaxies, the OPCClient object, based on industry standards, has been enhanced to function as the single DIObject capable and sufficiently full-featured to replace the deprecated objects and maintain connectivity with current and future PLCs.

This enhancement to the OPCClient object is part of the ongoing evolution of System Platform away from legacy DAServers and OI Servers, and towards the use of communication drivers. Many users continue to use legacy DINetwork Objects because of the ease of use provided by the default syntax that these objects allow, and to avoid having to reconfigure their IO references.

This evolution of the OPCClient object is designed to simplify the migration of existing System Platform galaxies. In prior releases of System Platform, the OPCClient object required that you provide the entire PLC hierarchy for

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

IO references, and thus converting from a DINetwork object to an OPCClient object required changing all existing IO references. In System Platform 2023 two additions were made to the OPCClient object:

- The new **HierarchyPath** attribute that was added to the OPCClient object allows an easy migration path from legacy DINetwork objects. HierarchyPath is only configurable in the IDE and cannot be changed at run time.
- A <Default> scan group was added to the **Scan Group** tab of the OPCClient to further simplify migration. This default group allows any scan group items that were in the DINetwork object's default scan group to be switched over seamlessly.

Note: References to items in custom scan groups can also be easily migrated.

Configure the OPC UA server

Application Server supports the OPC UA (Unified Architecture) protocol for machine-to-machine communications.

This OPC UA Server functionality allows third party clients to interact with Application Server and leverage industry standards, such as OPC UA. When OPC UA Server functionality is enabled on Application Server, a client can utilize the built-in OI Gateway or a third party client to connect to Application Server, and use OI Gateway or the client to securely browse the OPC UA namespace and interact with Application Server.

OPC UA configuration checklist

Required tasks for end-to-end configuration of the OPC UA server and OPC UA client

The configuration tasks are shown in the order in which they must be completed.

1. **Configure the System Management Server:** The System Management Server is used for establishing a trust relationship between machines, and must be configured to ensure secure communications between nodes. The System Management Server is normally configured during initial System Platform installation but can be changed as needed. See Common Platform Services for details.
2. **Set Galaxy Security:** Enabling Galaxy security is not mandatory but is strongly recommended. The OPC UA Server supports the Galaxy security configuration, and where username and password are required, it will leverage the Galaxy security configuration. See Configure security for details.
3. **Configure and deploy the OPC UA server:** Set the configuration options and deploy the OPC UA server to a run-time node. For details, see Configure and deploy the OPC UA service.
4. **IT compliance/firewall validation:** Firewall configuration and verification must be completed at this point of the configuration. The node to which the OPC UA Server has been deployed must have Inbound Rules for the firewall configured and verified.

   IMPORTANT! A firewall test must be successfully performed before proceeding with the remaining configuration tasks.
5. **Configure the OPC UA Client:** Client configuration may include the following:
   - Define the OPC UA server address (in the format `opc.tcp://<ServerName>:<PortNumber>`).
   - Select the correct OPC UA server security policy (`Basic256Sha256`).
   - Enter the configured OPC UA User Credentials (username and password).
   - Recommended: Disable Anonymous User. Anonymous User is enabled by default.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

6. **Security Certificate:** Download and configure the OPC UA security certificate on the run-time node.
7. **Validate connectivity:** Open the OPC UA client and verify that you can connect to the OPC UA Server, and can view items in the namespace.

Use an OPC UA client to read and write run-time data

You can leverage the OPC UA protocol to view and change attribute values and quality in a deployed, running galaxy from an external OPC UA node. This node can be an OPC UA connection configured on the Gateway Communication Driver, or a third party OPC UA client. From the OPC Client, you can browse the available namespaces on the run-time node. Note that your permissions in the OPC UA namespace are dependent on how you log in and how the Client Access Rules have been configured in the **Configure Services** dialog.

Prior to using OPC UA for browsing the galaxy namespace, you must configure and deploy the **AVEVA.OPCUAService**. See Configure and deploy the OPC UA service for details.

Depending on the requirements of your galaxy, you can create additional OPC UA server services. However, each service must have a unique port number.

Once the OPC UA service is configured and deployed on the run-time node, you must:

- Configure the firewall on the run-time node to allow the OPC UA client to access it. For details, see Configure the firewall for the OPC UA service.
- Install a security certificate on the OPC UA client node to ensure secure communications with the run-time node. For details, see Configure an OPC UA data source object

To use OPC UA client/server for reading and writing run-time data

1. Configure certificates for the Gateway Communication Driver OPC UA connection or third-party OPC UA client you will use for browsing the system. Certificates must be installed in the **Trusted People** certificate store.
   - If you log in as anonymous, access to the galaxy data is read-only. See Client access rules and galaxy security for additional information about how Galaxy security affects OPC UA client access.
   - If you log in with a valid username and password, access to the galaxy data is read/write.
   - OPC UA LogIn By Certificate is not supported.
2. Once you are connected to the OPC UA namespace and with the galaxy deployed, you can browse galaxy objects and attributes in both the **Model** and **Deployment** views. If the galaxy is not deployed, you can browse objects but not attributes.
3. If you have logged in with a valid username and password, you can change attribute values. If you are logged in anonymously, you can only view values. However, your access to data is also affected by the Galaxy security configuration and your permissions. For more information about how Galaxy security affects user access through OPC UA, see Client access rules and galaxy security.

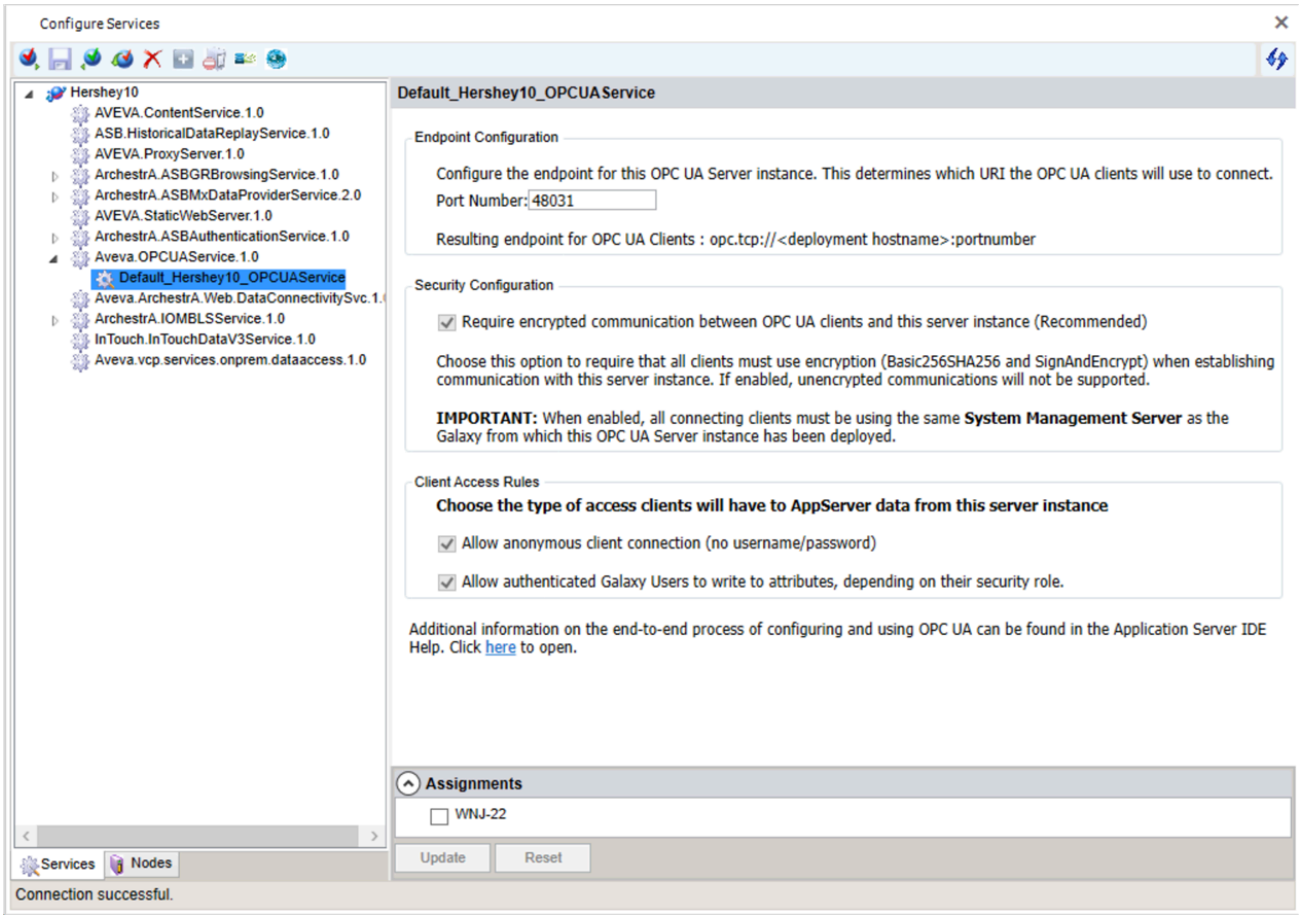
Configure and deploy the OPC UA service

The AVEVA OPC UA Service provides access from an OPC UA client to Application Server data, without the need for the Galaxy Browser, a gateway, or other protocol translation mechanism.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

To configure and deploy the OPC UA Server Service

1. On the IDE ribbon, select **Galaxy** then select **Configure**
2. Select **System**, then select **Services**.
3. The **Configure Services** utility opens.



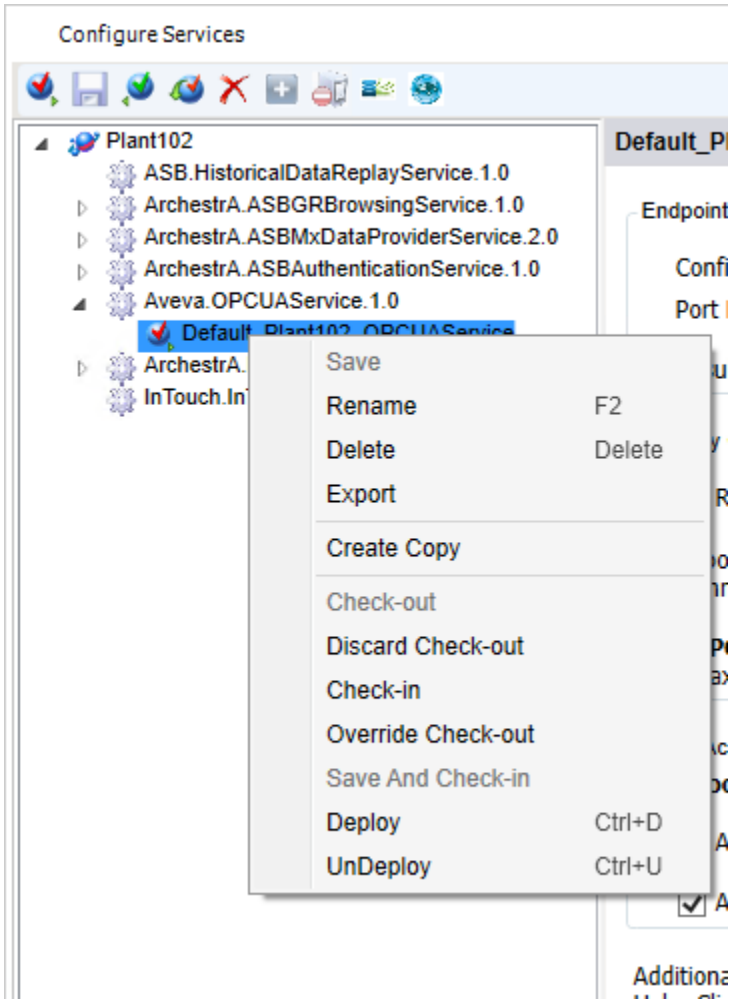
4. Expand the items in the left-hand pane as needed. Right-click the OPC UA service name and select **Check-out** from the context menu.
5. Edit the **port number** for the OPC UA instance if necessary. The default port is **48031**.
6. **Security configuration**
   - **Require encrypted communication:** It is strongly recommended that you enable this option, as this will encrypt the payloads across the connection. Note that the client must match this configuration.

     Important! An OPC UA connection cannot be established if you do not enable this option while OS security is enabled, even if the option "Allow authenticated Galaxy Users to write attributes" (under Step 7) is enabled.
7. **Client access rules**
   - **Anonymous client connection:** Allowing anonymous client connections is recommended ONLY for initial setup configurations and testing. Anonymous client connections should be disabled for production environments. Refer to Client access rules and galaxy security to see the effect that this option has on data access.
   - **Allow authenticated Galaxy Users to write attributes:** Enabling this setting will only take effect when

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

encrypted communication is also enabled (see Step 6, above). The OPC UA client must match this configuration.

8. The **Assignments** section (below the right pane) shows both the local node and deployed remote runtime nodes to which the OPC UA service can be assigned. Select the checkmark next to the runtime node where you want to deploy the OPC Server service, and then click **Update**.
9. In the left pane, right-click the OPC UA service instance name and select **Check-in** from the context menu.



10. Right-click the instance again to open the context menu and select **Deploy**, or press CTRL+D. A message appears indicating whether the service has been successfully deployed to the OPC UA client node. If deployment is successful, the icon next to the instance name changes to indicate that the instance has deployed.
11. If you have deployed the service to a remote node, you can verify functionality through the Operations Control Management Console.
    a. Under Operations Integration Server Manager, select the remote node name.
    b. Expand Operations Integration Supervisory Servers and select the OPCUA Client (under OI.GATEWAY.3). Check the following settings:
        a. Server Node: localhost
        b. OPCUA Server: opc.tcp://localhost:48031

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

To add additional OPC UA services

Each OPC UA service is dedicated to a single OPC UA client node. To add additional OPC UA services:

1. Right-click **AVEVA.OPCService**, and then select **Create** from the context menu, or press CTRL+N. The new instance appears in the tree structure.

   Note: Each instance must have a unique port number. Enter the port number in the **Base Address** field. The default port number is **48031**. See Configure service TCP ports for a list of port numbers used by ASB services.
2. Rename the OPC UA service as needed. Right-click on the service name and select **Rename** from the context menu, or press F2. Then, enter the new name.
3. Repeat the steps above for configuring and deploying each additional OPC UA service.

To change a deployed OPC UA service

1. Check out the service instance.
2. Make any needed changes.
   - **Port Number:** If you are creating multiple services, each service instance should have a unique port number. If more than one service has the same port number, an error is generated in the logger. Multiple instances of the service can be deployed, as long as each service has a unique port number. A new URI (uniform resource identifier) is automatically generated when a port number is changed.

     Note: You may need to open the inbound port in the firewall to allow communication with the remote node.
   - **Security Configuration:** When enabled (default), communication between OPC UA clients and the OPC UA server is encrypted. This is the recommended setting. If this setting is unchecked (disabled), communication is not encrypted.
   - **Client access rules**

     Allow anonymous client connection (enabled): When enabled (default), an anonymous OPC UA client is allowed to connect to the OPC UA server. This is recommended only for testing and initial set up configurations. Once you have completed configuration and/or testing, be sure to disable this setting to provide protection against possible unwanted intrusions and to ensure that only authenticated users have access. Anonymous client connection should not be enabled in a production environment.

     Galaxy Security settings do not have any affect on these behaviors. See Configure security for more information.

     Allow authenticated Galaxy user to write to attributes (enabled): When enabled (default), an authenticated Galaxy user can change attribute values in runtime, if their security role allows them to do so. See About roles for more information.

     Allow authenticated Galaxy user to write to attributes (disabled): When disabled (unchecked), an authenticated Galaxy user is not permitted to change attribute values in runtime, even if their security role allows them to do so.

     See Client access rules and galaxy security for more information about user permissions for each setting combination.
3. Check in the service or services.
4. Undeploy and then redeploy the service or services.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Client access rules and galaxy security

Client Access Rules configured for the OPC UA Service interact with the Galaxy security authentication mode to allow or deny different levels of access for authorized users.

There are two configurable Client Access Rules in the OPC UA Service dialog. By default, both rules are enabled:

- Allow anonymous client connection (no username/password)
- Allow authenticated Galaxy Users to write to attributes, depending on their security role

The following table defines the level of data access users are allowed under different combinations of Client Access Rule configurations, Galaxy security authentication mode, and the type of OPC UA credentials (anonymous or authenticated user with username/password).

If security for the Galaxy is enabled (Galaxy security = **Secured**, column **1**), encrypted communication between the OPC UA clients and OPC UA service must also be enabled. See Configure and deploy the OPC UA service.

Galaxy security	OPC UA	Client access rules		Level of data access		
Authentication mode	**Client credentials**	**Allow anonymous connection**	**Allow authenticated Galaxy users**	**Connect**	**Read**	**Write (see below)**
Secured	Authenticated	Enabled	Enabled	YES	YES	YES
Secured	Authenticated	Enabled	Disabled	YES	YES	NO
Secured	Authenticated	Disabled	Enabled	YES	YES	YES
Secured	Authenticated	Disabled	Disabled	YES	YES	NO
Secured	Anonymous	Enabled	Enabled	YES	YES	NO
Secured	Anonymous	Enabled	Disabled	YES	YES	NO
Secured	Anonymous	Disabled	Enabled	NO	N/A	N/A
Secured	Anonymous	Disabled	Disabled	NO	N/A	N/A
None	Authenticated	Enabled	Enabled	NO	N/A	N/A
None	Authenticated	Enabled	Disabled	NO	N/A	N/A
None	Authenticated	Disabled	Enabled	NO	N/A	N/A
None	Authenticated	Disabled	Disabled	NO	N/A	N/A
None	Anonymous	Enabled	Enabled	YES	YES	NO
None	Anonymous	Enabled	Disabled	YES	YES	NO
None	Anonymous	Disabled	Enabled	NO	N/A	N/A

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

None	Anonymous	Disabled	Disabled	NO	N/A	N/A

Important: Whenever Client Access Rules and Galaxy Security allow a user to write data, this permission is always conditioned by whether or not the user's configured security role also allows them to write data to a specific attribute. This means that when Galaxy security is enabled, the user's security role must explicitly allow them to write to attributes, regardless of the OPC UA client access rule setting. If their security role does not allow them to write to attributes, they cannot, even if the level of data access in the above table shows that they can.

Configure the firewall for the OPC UA service

OPC UA communications in Application Server require that the run-time node firewall allows a connection with an OPC UA client node. Before changing firewall settings, however, it is recommended that you perform a Firewall test.

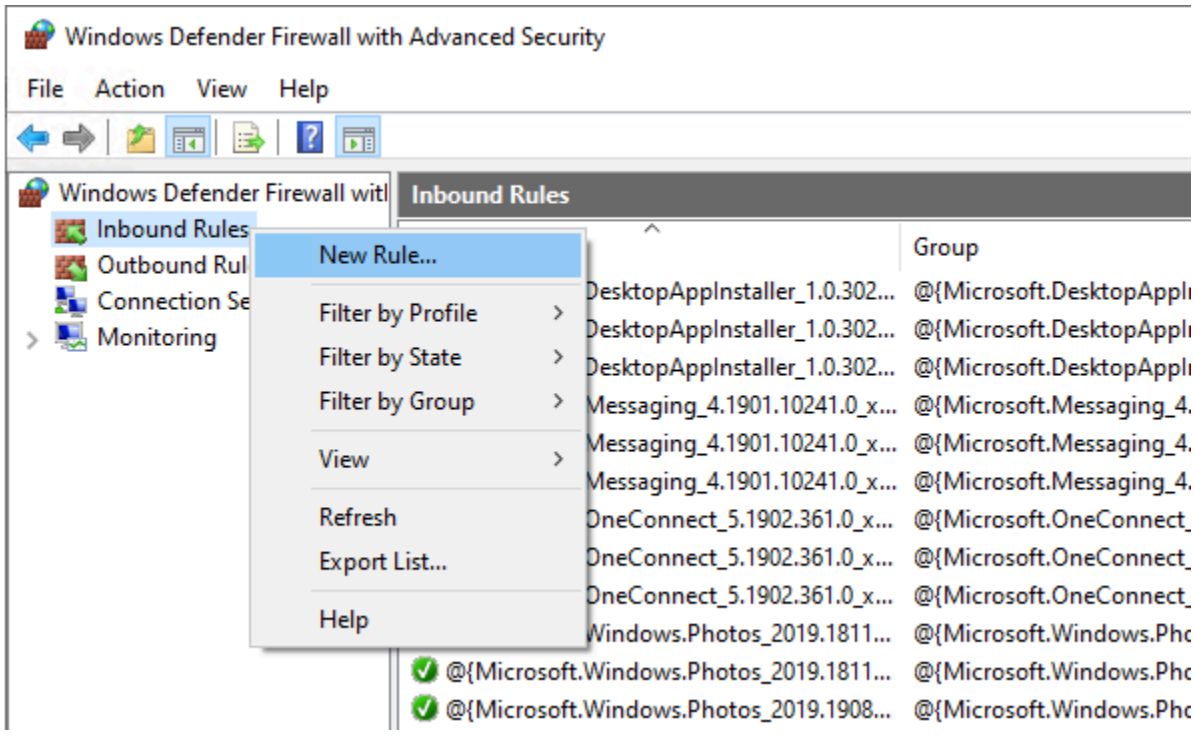
Configure the run-time node firewall

Important: The firewall rules must be added to the node to which the OPC UA Server Service is deployed.

To configure the run-time node firewall

On the run-time node(s) where the OPC UA Server Service is deployed, open the Windows firewall and configure it as follows:

1. In the Windows Search bar, open **Windows Firewall**.
2. Select **Advanced Settings** and create an **Inbound Rule**.

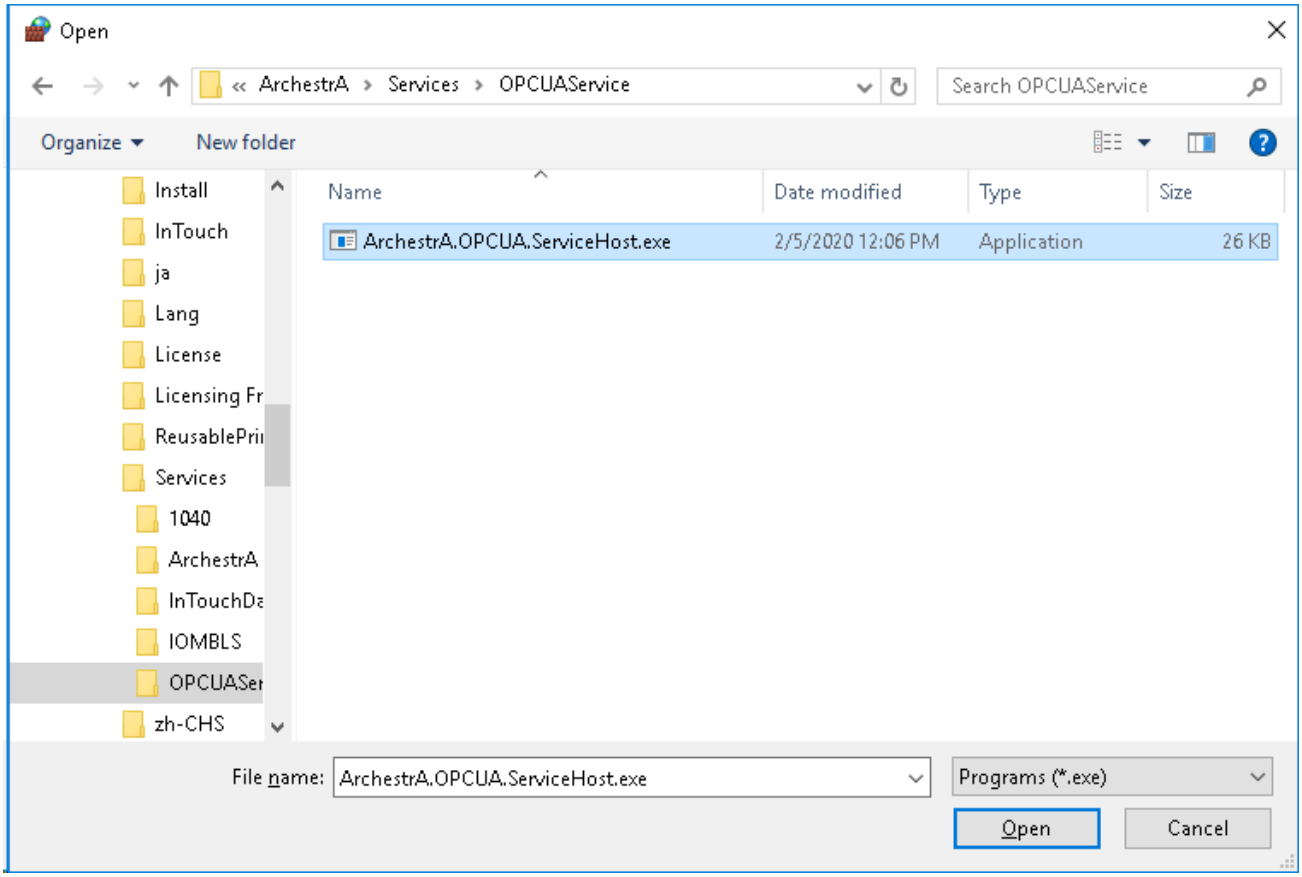


3. Click "**New Rule**." The **Rule Wizard** opens. .

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

4. Select **Program** for the Rule Type and click **Next**.
5. Browse to the OPC UA Server location. If System Platform was installed in the default location, the path should be:

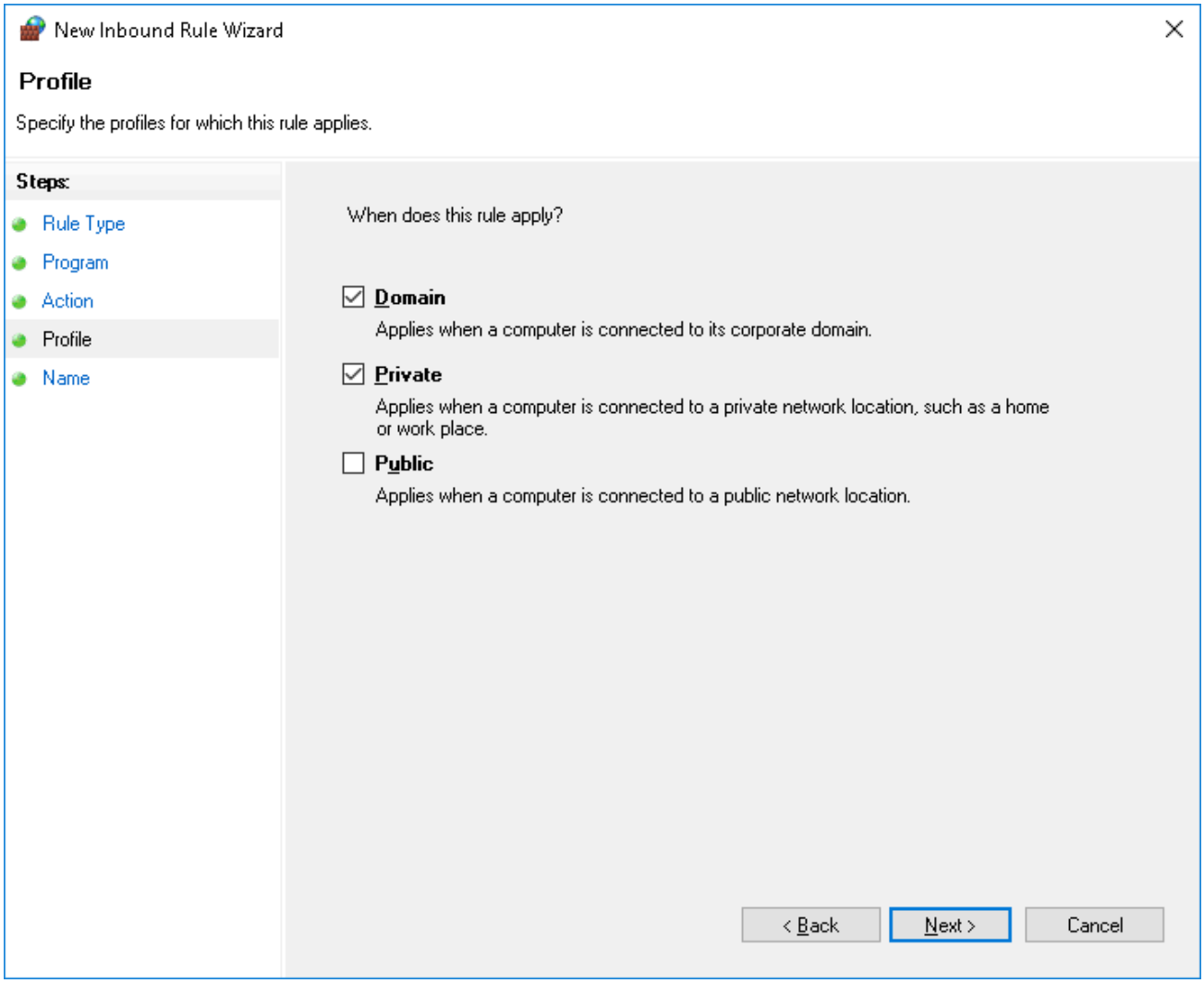
   `C:\Program Files (x86)\Common Files\ArchestrA\Services\OPCUAService`



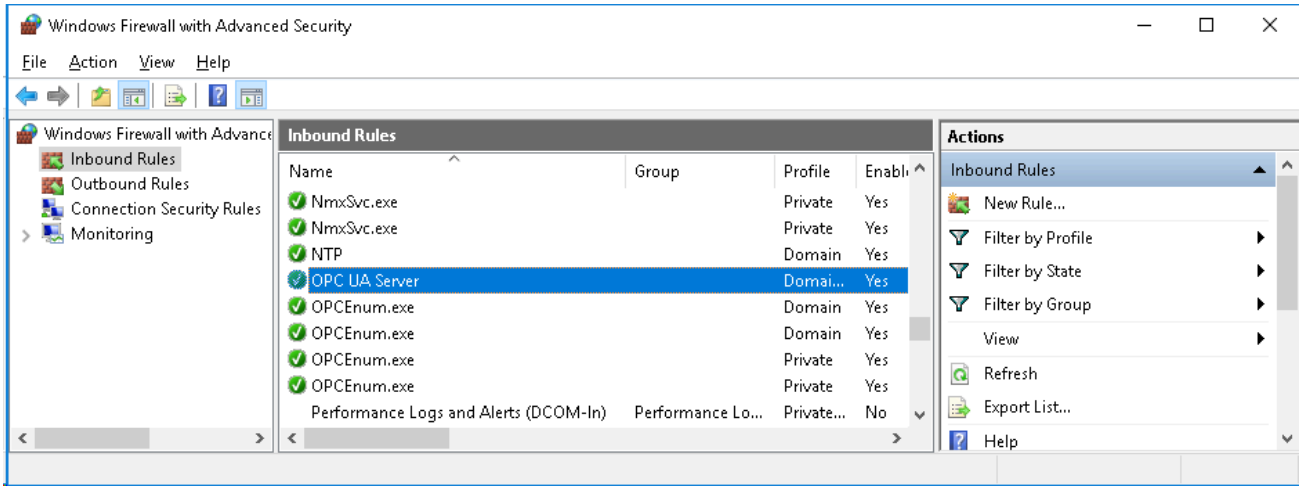
6. Select "**ArchestrA.OPCUA.ServiceHost.exe**" and then click **Open**.

   This adds the file path to the wizard. Click **Next**.
7. On the **Action** dialog box, select the option "**Allow the connection**." Click **Next**.
8. The wizard will ask when the rule applies.
   - For **Domain** environments: Select **Domain** and **Private**. We recommend that you deselect **Public**.
   - For **Workgroup** environments: Select **Public**. The Domain and Private settings have no affect in a Workgroup environment.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.



9. Finally, provide a name for this rule (for example, "OPC UA Server"). If you will be configuring multiple OPC UA services from Application Server, be sure to use names that differentiate each service from the others.
10. Now, check that the new rule has been added to the list of InBound Rules in the Windows Firewall and that it is enabled.



11. Verify that you can connect to the run-time node from the OPC UA client node by repeating the Firewall test.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Firewall test

To perform a firewall test with Telnet

1. From a separate node from where the OPC UA Server service is running, enable **Telnet** by turning on the **Telnet Client** Windows Feature.

   Note: This test is ideally run from the OPC UA client node.

   a. Open the Windows **Control Panel**.
   b. Go to **Programs and Features**, then select **Turn Windows features on or off.**
   c. Scroll down the list of features to **Telnet Client** and enable it. Telnet is disabled by default.
2. Prior to running this test, verify that the Application Server OPC UA Service has been deployed from the run-time (GR) node to the OPC UA client node. See Configure and deploy the OPC UA service for details.
3. Run Telnet in a command window on the OPC UA client node by entering the following:

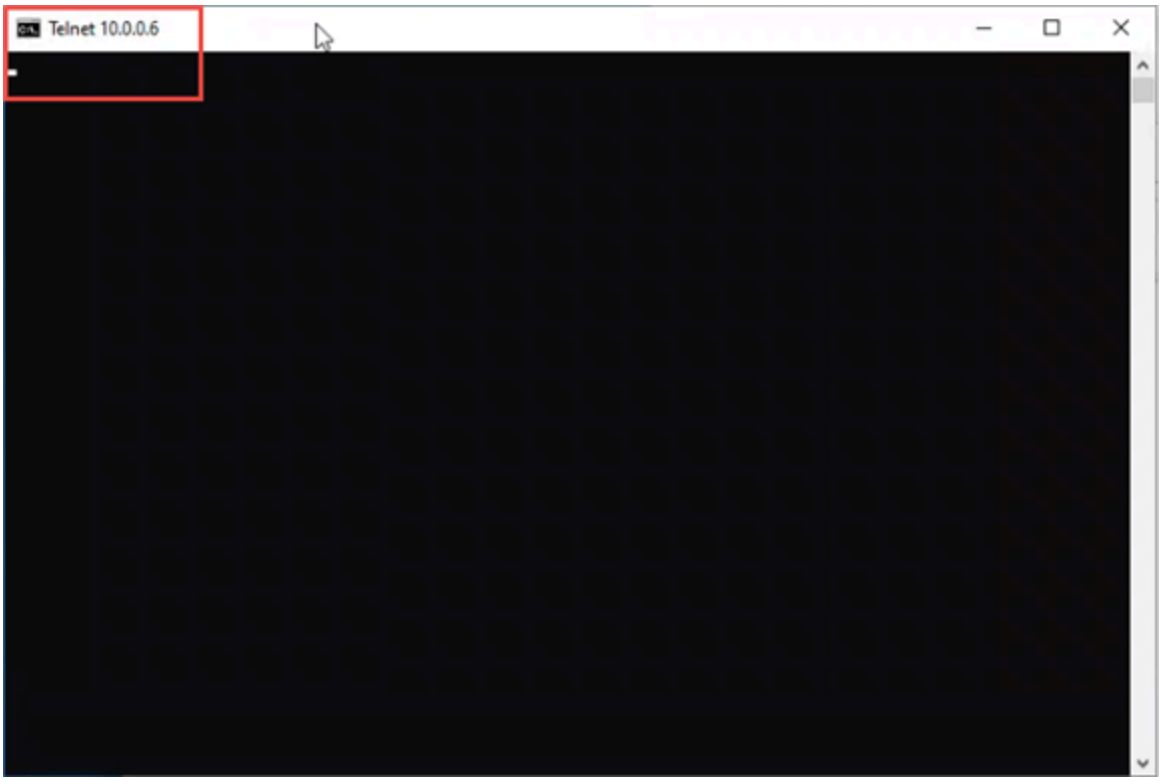
   telnet <nodeName or ipAddress> <portNumber>

   where

   - **nodeName** is the machine name of the Application Server run-time node. Use nodeName or ipAddress, not both.
   - **ipAddress** is the IP address of the Application Server run-time node. Use nodeName or ipAddress, not both.
   - **portNumber** is the port number you configured in Application Server for the OPC UA Service. The default port number is 48031.

     Example: `telnet 10.10.10.06 48031`
   - If the command is not successful, it will time out with a message stating that the connection failed. In this case, go to Configure the run-time node firewall.
   - If the telnet command is successful, the command prompt changes to a Telnet prompt.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.



4. If the Firewall Test is successful, configure the OPC UA client and OPC UA server certificates. The next step in setting up your OPC UA connection depends on if you are using a third-party OPC UA client application, or the OPC UA connection available with OI Gateway. See either:
   - Configure server and client certificates for third-party OPC UA client applications, or
   - Configure an OPC UA data source object

Configure server and client certificates for third-party OPC UA client applications

IMPORTANT! These procedures apply ONLY if you are using a third party OPC UA client. If you are using OI Gateway as the OPC UA client, skip to Configure an OPC UA data source object.

To configure encrypted communications between the System Platform OPC UA server and a third-party OPC UA client, both computers will need access the following certificates:

- <Computer Name> ASB OPC UA Server
- The client certificate from your OPC UA client.

To complete this setup, you will need to perform three steps:

1. Copy the certificate from the OPC UA server node to the OPC UA client node. This step includes the following:
   - Exporting the OPC UA server certificate.
   - Installing the certificate on the client node.
2. Copy the certificate from the OPC UA client node to the OPC UA server node.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

3. Make sure that the firewall has been configured to allow the ArchestrA.OPCUA.ServiceHost.exe application.

Export the OPC UA server certificate to the OPC UA client node

Export the OPC UA server certificate

1. Open the Windows Certificate Manager on the OPC UA server node.

   To open the Certificate Manager, either type "`Manage Computer Certificates`" in the Windows search box and select, or open the command prompt and run "`certlm.msc`."
2. In the tree view, expand the **Personal** node, then select on **Certificates**.
3. Locate the certificate "**<computer name> ASB OPC UA Server**".
4. Right-click on the certificate and select 'All Tasks\Export…"

   The **Certificate Export Wizard** opens.
5. Depending on type of certificates that your client application uses, you will need to export the certificate as one of two certificate file types:
   - **.cer file**, if the client uses the Windows Certificate Store
   - **.der file**, if the client uses file-based certificates.

   Even though the file extensions are different, the file formats are the same.
6. In the Certificate Export Wizard, select the following options.
   - **Export Private Key:** select "No, do not export the private key" (default), then select **Next**.
   - **Export File Format:** select either "DER encoded binary X.509 (.CER)" or "Base-64 encoded X.509 (.CER)," depending on the requirements of your client application. Make the selection, then select **Next**.
   - **File to Export:** enter a file name to export the Root CA, for example "`c:\temp\<machine name> OPC UA Server.cer`," then select **Next**.
7. When the **Export Wizard** finishes, you may need to change the file extension of the certificate from ".cer" to ".der," depending on what your client application is expecting. You will need to do this if your OPC UA Client application stores certificates in a specific folder, rather than in the Windows Certificate Store.

Import the OPC UA server certificate on the client computer

To complete the process of copying and installing the OPC UA server certificate to the OPC UA client node, you need to import the OPC UA Server certificate to the OPC UA client computer.

Each OPC UA client application has its own mechanism to manage certificates. Generally, an OPC UA client will use one of two mechanisms to manage certificates:

- Utilizing the Windows Certificate Store
- Storing certificates in a specified folder, defined by the OPC UA Client application.

Refer to the documentation for your OPC UA client applications for more details on how to import a server certificate.

Import a certificate into the OPC UA client windows certificate store

1. Copy the certificate file (<machine name> OPC UA Server.cer) to the OPC UA client node. Where you copy

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

the file is not important.

2. Right click on the certificate file and select **Install Certificate** from the context menu.

   This opens the Certificate Import Wizard.

   Note: Administrator privileges are required to import the certificate.

3. In the Certificate Import Wizard, select the following options.
   - **Store location:** Select "Local Machine," then select **Next**.
   - **Certificate store:** Browse to "Personal," then select **Next**.
   - **Completing the Certificate Import Wizard:** Review the settings, then select **Finish**.

Copy a certificate to specified location on the OPC UA client

1. Copy the certificate file (<machine name> OPC UA Server.cer) to the folder that is specified by your OPC UA client applications. The following table shows the location of the certificate folder for a number of common OPC UA clients.

OPC UA client	Manufacturer	Folder
Datafeed OPC UA Client	Softing	C:\ProgramData\Softing\OpcClient\pki\trusted\certs
UaExpert	UnifiedAutomation	C:\Users\Admin\AppData\Roaming\unifiedautomation\uaexpert\PKI\trusted\certs
UA Client Getting Started	UnifiedAutomation	C:\ProgramData\unifiedautomation\UaSdkNetBundleEval\pkiclient\trusted\certs
Matrikon	Matrikon	C:\Users\Admin\Appdata\Local\Matrikon\OPCUAExplorer\pki\DefaultApplicationGroup\trusted\certs
KEPServer	Kepware	C:\ProgramData\Kepware\KEPServerEX\V6\UA\Client Driver\cert
Top Server	Software Toolbox	C:\ProgramData\Software Toolbox\TOP Server\V6\UA\Client Driver\cert

   Refer to the documentation for the OPC UA client application for additional information about managing certificates.

2. Configure the OPC UA certificate, as described below.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Configure OPC UA client certificates on the OPC UA server

The next step in the of configuring OPC UA server and client certificates is to trust the OPC UA client certificate that has been installed on the OPC UA server node.

The easiest way is to attempt to connect an OPC UA client to the server. Since trust of the client certificate has not yet been established, the connection is expected to fail.

Once the connection fails, any OPC UA client certificates that are not installed on the OPC UA server are placed in the “Rejected Certificate” folder on the OPC UA Server.

By default, the folder location is:

C:\ProgramData\AVEVA\PCS\OPC UA Rejected Client Certificates\certs

Note: Access to this folder requires administrator rights, and the folder is hidden by default.

Import certificates placed in the Rejected Certificate folder

1. To import the OPC UA Client certificates, browse to the rejected certificate folder.
2. Right click on the certificate for the OPC UA Client that you want to trust and select “Install Certificate”. This opens the **Import certificate Wizard**.
3. Select the following options in the wizard:
   - **Store location:** Select "Local Machine," then select **Next**.
   - **Certificate store:** Select “Trusted People,” then select **Next**.
   - Completing the Certificate Import Wizard: Review the settings, the select **Finish**.

OPC UA ports and port configuration

OPC UA communicates via a single TCP port. This is specified in the **Endpoint Connection** setting in the configuration of the OPC UA Server, and defaults to port 48031. For details, see Configure and deploy the OPC UA service

If you will run multiple OPC UA Servers on the same computer, you must specify a different port number for each subsequent server.

Confirm that this port is not blocked by any firewall software installed on your computer. For information about testing and configuring communications through firewall, see Configure the firewall for the OPC UA service

Configure an OPC UA data source object

OI Gateway provides a convenient method for configuring security certification on the server and client OPC UA nodes.

To access the OI Gateway

1. From the Start Menu on the run-time node, open the Operations Control Management Console (OCMC) (**Start > AVEVA > Operations Control Management Console).**
2. In the console tree, navigate to the **OI.GATEWAY.3** node under Operations Integration Supervisory Servers.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

To add an OPC UA data source object to your Gateway Communication Driver Hierarchy

1. Right-click **Configuration** in the hierarchy, and select **Add OPCUA Connection** from the shortcut menu.

   A new object is created in the hierarchy tree.
2. The OPCUA Connection object is named "New_OPCUA_000" by default. Rename it, if desired.

Step 1: To Configure the OPCUA Server Details

Use the **OPCUA Server Details** section of the OPC UA editor to configure the OPC UA Server. Ensure that the configuration on the OPCUA faceplate matches the configuration settings in the OPCUA Server that you are trying to connect.

Prerequisites:

Before configuring the OPC UA Server Details, ensure that you have the following details:

- If the OPC UA server can be directly connected:
  - **OPC UA Server Endpoint URL:** Required for the OPC UA client to connect to the OPC UA server. Note down the endpoint from your OPC UA server. At the least, you should know the IP address of the OPC UA server.

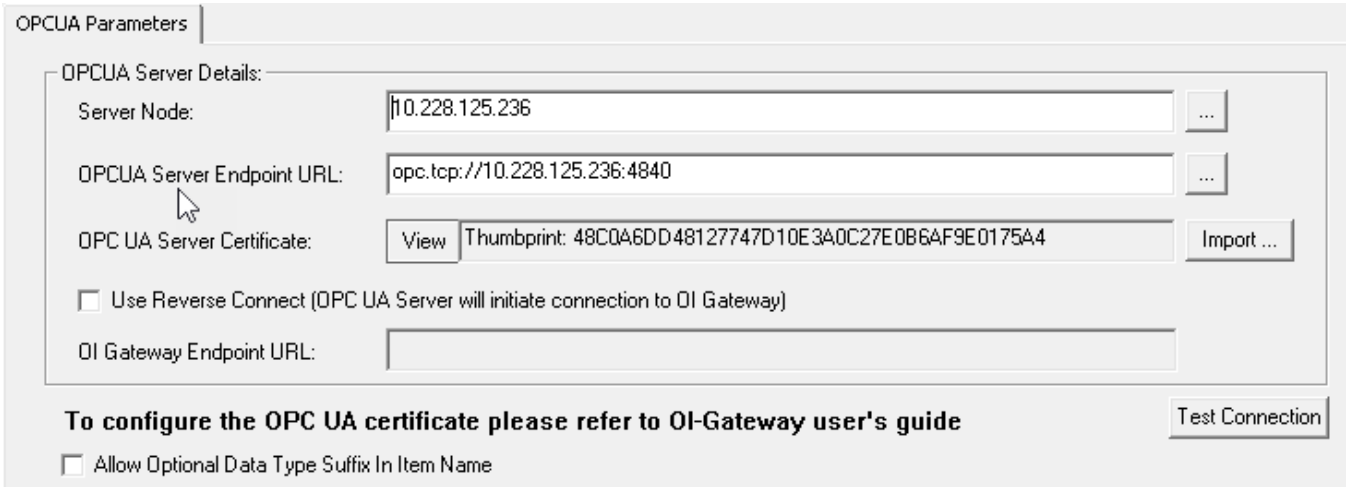
    The Gateway Communication Driver can also connect to an OPC UA server in an IPv6 network by entering the link-local IPv6 address of the machine on which the OPC UA server runs.

    The general syntax of the OPC UA Server Endpoint URL is:

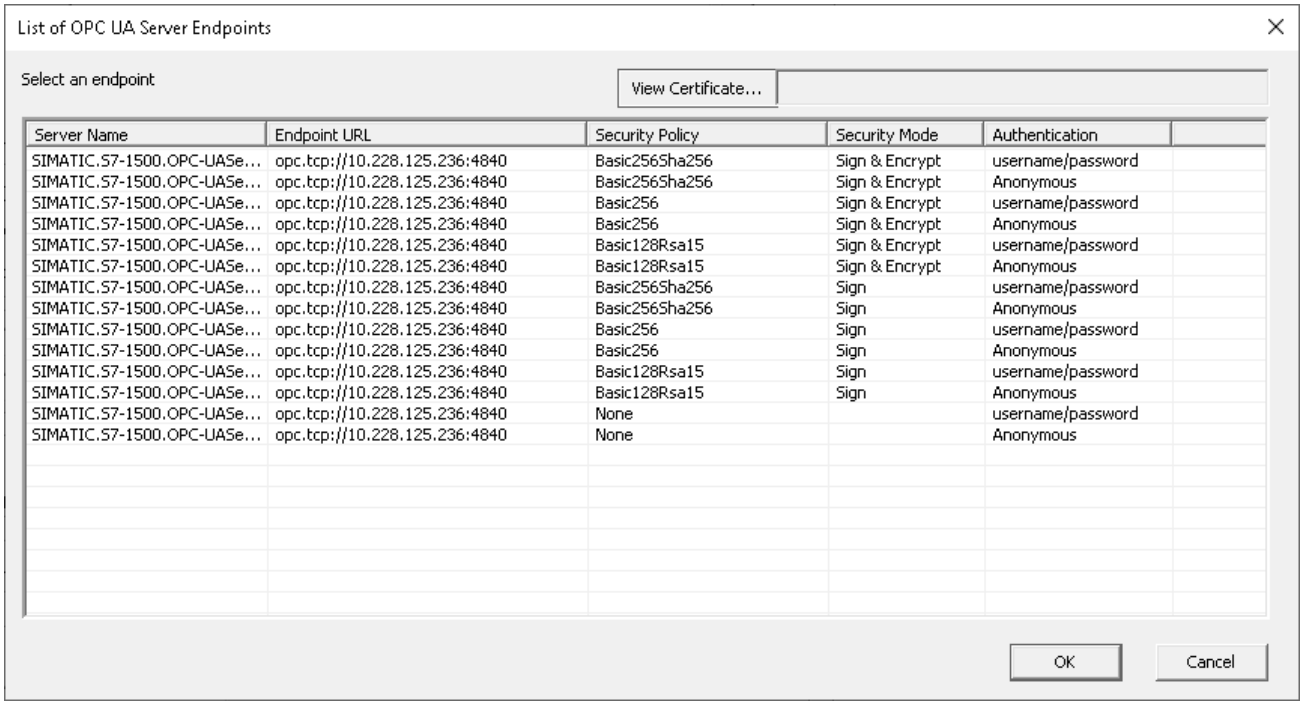
    `opc.tcp://[<IP address of the OPC UA Server>]:<Port number>/<Server Name>`
  - **OPC UA Server Certificate:** Required to establish a trust relation between the OPC UA server and OPC UA client. You can import a digital certificate of the OPC UA directly. Alternatively the Gateway Communication Driver can download a certificate from the OPC UA. Review the certificate to ensure that it is genuine before trusting it.
  - **Security Policy:** Indicates the encryption policy used for the connection. Note down the security policy used by the OPC UA server.
  - **User Authentication:** If your OPC UA server is configured with the User name and Password, then you need to enter the same user credentials for the OPC UA connection. Note down the User name and Password that is used in your OPC UA server.
- If the OPC UA server is protected by a firewall and cannot be directly connected, you can use the OPC UA Reverse Connect feature where the OPC UA server can be configured to initiate the connection to the Gateway Communication Driver:
  - **Gateway Communication Driver Endpoint URL:** Required while using Reverse Connect to allow the OPC UA Server to initiate a connection with the Gateway Communication Driver. Note down the complete endpoint URL as mentioned in the OPC UA server for the reverse connect client URL.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

To configure the OPC UA Server details



1. In the **Server Node** field, browse the **Server Node** using the browse button (...), or enter the server node name or the IP address.
2. OPC UA **Server Endpoint URL field:** If you know of the endpoint exposed by the OPC UA server, you can enter it manually. If the OPC UA server supports local discovery through OPC UA defined port 4830, you can just leave this field blank. When you click on the ellipse (...) button, the end point selection dialog will be posted to allow you to select the appropriate endpoints supported by the OPC UA server. Be sure to click on the **View Certificate** button in the **Endpoint Selection** dialog to ensure the digital certificate reflects appropriately the corresponding OPC UA server that you are connecting to. You can select any endpoint in the list by selecting any entry in the list followed by the **OK** button or by simply double-clicking on any entry.



3. In the **OPC UA Server Certificate** field, click the **Import** button to add the OPC UA server certificate, or enter the details manually. Select the certificate that you have downloaded from your OPC UA server and click **OK**. In the **Certificate Import** window that appears, click **View Certificate** to verify if you have selected the

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

correct certificate, and then click **Accept** to trust the certificate. You can also click **View** if you wish to see the certificate details after trusting the certificate.

Important: Importing a certificate from an untrusted source may incur a security risk. It is recommended to review the certificate to ensure it is issued by a trusted authority before accepting it.

4. If the OPC UA Server is protected by a network firewall and that it also supports OPC UA Reverse Connect, select the **Use Reverse Connect (OPC UA Server will initiate connection to OI Gateway)** checkbox. The OPC UA Reverse Connect feature enables the OPC UA server to initiate a connection with the Gateway Communication Driver. In a secure environment, inbound connections to the OPC UA server are restricted as the server operates behind a firewall. By using Reverse Connect, the firewall restriction is alleviated as the connection is being made from the OPC UA server to the client. This field is optional. If you do not select this option, you can skip to step 6.
5. If you select the **Use Reverse Connect (OPC UA Server will initiate connection to OI Gateway)** checkbox in the above step, the **OI Gateway Endpoint URL** field gets enabled. This is endpoint that the Gateway Communication Driver will listen to incoming connection from the OPC UA server. Make sure the port specified in this URL is not blocked by the local firewall of the Gateway Communication Driver computer. An inbound rule may need to be created in the firewall setting to open the specific local port for TCP connection. This must have the same value of the end point URL configured in the OPC UA server for Reverse Connect.

   The general convention for the URL is: `opc.tcp://<hostname>:<port>/<reference>`

   where,

   <hostname>: machine host name

   <port>: tcp port number that is available and opened at the firewall

   <reference>: a unique string that identifies this connection. Ensure that this string is unique across reverse connection strings in any instances and hierarchies of Gateway Communication Driver on the same machine.
6. Click **Test Connection** to establish the connection between the OPC UA Server and Gateway Communication Driver. When Reverse Connect is used to listen to any connection from the OPC UA server, a progress dialog will be displayed. The OPC UA server will need to connect to OI Gateway within 120 seconds to avoid any connection time-out in the progress dialog.

   Once the connection is established, the **Advanced Configuration** section is minimized and the updated **OPC UA Namespace** section is displayed.

   Note: When using Reverse Connect, the Gateway Communication Driver has an internal time-out of 120 seconds while awaiting connection initiated from the OPC UA server. You can also cancel the test anytime by clicking on the **Cancel** button.
7. Select the **Allow Optional Data Type Suffix in Item Name** check-box to add data type as a suffix to an item name. This parameter is hot configurable. For more information about OPC UA data item names, access the OPC UA Tag Browser or see OPC UA Item Names and Syntax.

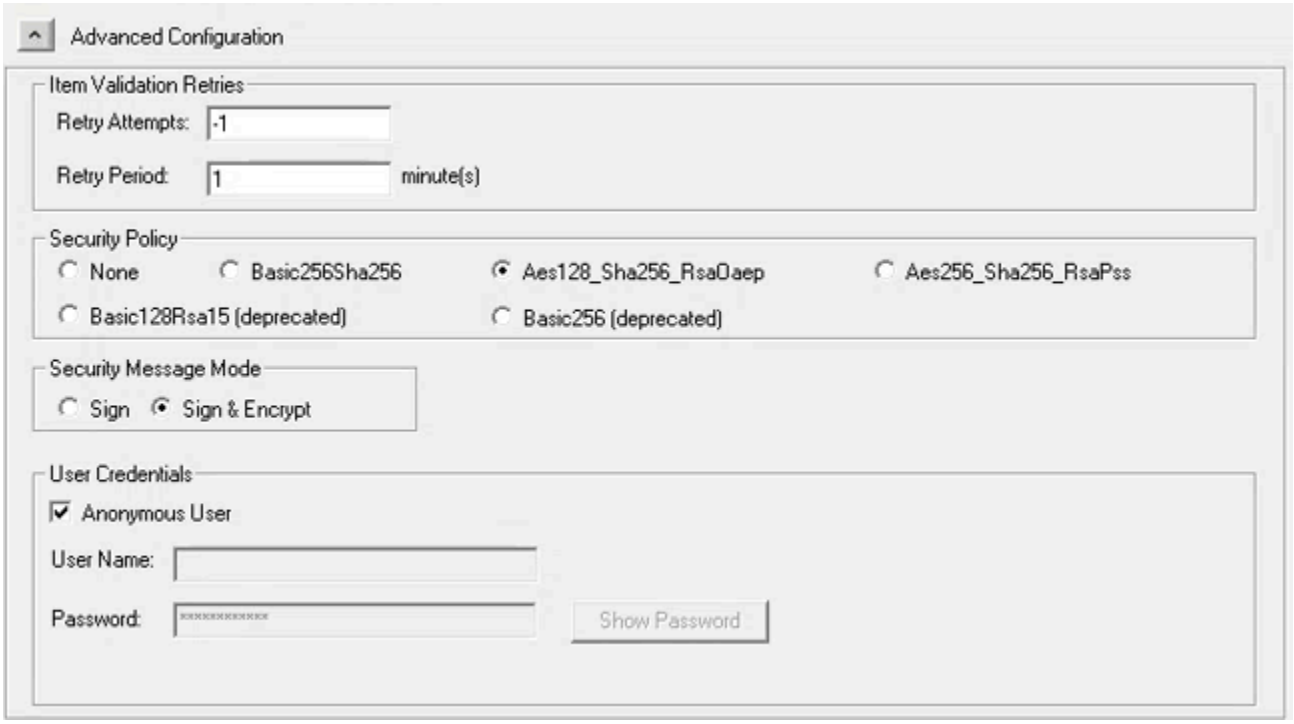
For errors while connecting to the OPC UA Server, see Connectivity with the OPC UA Server in the Troubleshooting chapter.

Step 2A: Advanced Configuration (If using secured OPC UA connection)

Use the **Advanced Configuration** section of the OPC UA editor to set the OPC UA connection security parameters: Security Policy, Security Message Mode, and User Credentials. To expand/collapse the **Advanced Configuration** section, click the arrow next to **Advanced Configuration**. You can click the arrow button to

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

maximize or minimize the Advanced Configuration section.



To configure the OPCUA Security

Before configuring the OPC UA security parameters, we recommend that you reference the OPCUA Server to match the security configuration.

- **Item Validation Retries**

  The Item Validation Retries allows to configure the retry details for item validation.

  - **Retry Attempts**: Indicates the number of Retry attempts for the validation. The valid range is between -1 to 10000000. The default value is -1.
  - **Retry Period**: Indicates the retry period (in minutes) for each retry of the item validation. The valid range is 1 to 10000000.The default value is 1 minute.

- **Security Policy**

  The Security Policy indicates the encryption policy used for the connection.

  - **None**: No encryption is applied.
  - **Basic256Sha256:** Indicates that Basic256Sha256 security is applied.
  - **Aes128_Sha256_RsaOaep:** Indicates that Aes128_Sha256_RsaOaep is applied.
  - **Aes256_Sha256_RsaPss**: Indicates that Aes256_Sha256_RsaPss is applied.
  - **Basic128Rsa15 (deprecated):** Indicates that Basic128Rsa15 security is applied.
  - **Basic256 (deprecated):** Indicates that Basic256 security is applied.

  Note: It is strongly recommended to use the security policy (Basic256Sha256, Aes128_Sha256_RsaOaep or Aes256_Sha256_RsaPss) to encrypt communication with the server. Encryption is critically important when passing username and password to an OPC UA server. Note that legacy policies Basic128Rsa15 and Basic256 are no longer considered secure and are classified as deprecated. While those deprecated policies may be preferable to no encryption at all, it is recommended that a more secure policy is used when supported by targeted OPC UA servers.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

- **Security Message Mode**

  The **Security Message Mode** indicates the message mode of the connection. If the **Security Policy** is set to **None**, the **Security Message Mode** is **Not Applicable**. If the **Security Policy** is set to any options other than **None** (that is, Basic128Rsa15, Basic256, or Basic256Sha256), the **Security Message Mode** options - **Sign** and **Sign and Encrypt** populates for selection.

  - **Sign:** All messages are signed but not encrypted.
  - **Sign and Encrypt:** All message are signed and encrypted. This is the most secured option and is recommended to use this option.

- **User Credentials**

  This section allows you to configure the user credentials for the OPC UA connection. Select the **Anonymous User** checkbox to allow the OPC UA client to connect to the OPC UA server without credentials. If your OPC UA client wants to connect to an OPC UA server that does not support anonymous connections, the OPC UA client must provide a valid user name and password. Clear the checkbox to provide a user name and password in their respective fields.

  Note: If User Name and password is configured for connection with OPCUA Server, it is recommended to configure the appropriate security policy Basis256Sha256, Aes128_Sha256_RsaOaep or Aes256_Sha256_RsaPss) to ensure username and password values are fully encrypted.

- Once you configure all the fields, click the **Save** icon to save your configuration.

Step 2B: Trusting the gateway communication driver OPC UA certificates with the OPC UA server

When using a secured connection to the OPC UA Server, the Gateway Communication Driver node must be trusted by the OPC UA Server. Apart from the procedure mentioned in **Step 1: To Configure the OPCUA Server Details** for trusting OPC UA server, you can also trust the certificate manually.

Follow the steps below to trust the Gateway Communication Driver node.

1. Open the OPC UA configuration.
2. Ensure that the configuration on the OPC UA faceplate matches the configuration settings in the OPC UA Server that you are trying to connect. Click **Test**.

   A new Gateway Communication Driver OPC UA certificate is automatically copied to the OPC UA Server.
3. Follow the OPC UA Server workflow to trust the new certificate that was published to it.
4. Return to the OPC UA Configuration screen and click **Test** to ensure that the operation is successful.

Note: If you are upgrading from a version older than 5.2, you must trust the certificate again as explained above. It is recommended to download the OPC UA Server Certificate directly, and import it while configuring the OPC UA Server details. This ensures that the certificate is genuine.

Step 2C (Optional): Specify your own certificate for the Gateway Communication Driver

If you want to use your own certificates (self-signed or CA-signed), as a certificate for the Gateway Communication Driver, follow the steps below:

1. Gateway Communication Driver uses a digital certificate named "OIGateway OPCUA". To use your own certificate, replace the existing Gateway Communication Driver certificates with your own certificates, using the existing "oiGateway OPCUA" name.
2. You must replace the public certificate and the private key in the respective folders:
   - Public certificate (.der file): **C:\ProgramData\Wonderware\OI-Server\$Operations Integration**

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Supervisory Servers$\OI.GATEWAY\CertificateStores\certs\OIGateway OPCUA.der

- Private key (.pem file): **C:\ProgramData\Wonderware\OI-Server\$Operations Integration Supervisory Servers$\OI.GATEWAY\CertificateStores\private\OIGateway OPCUA.pem**

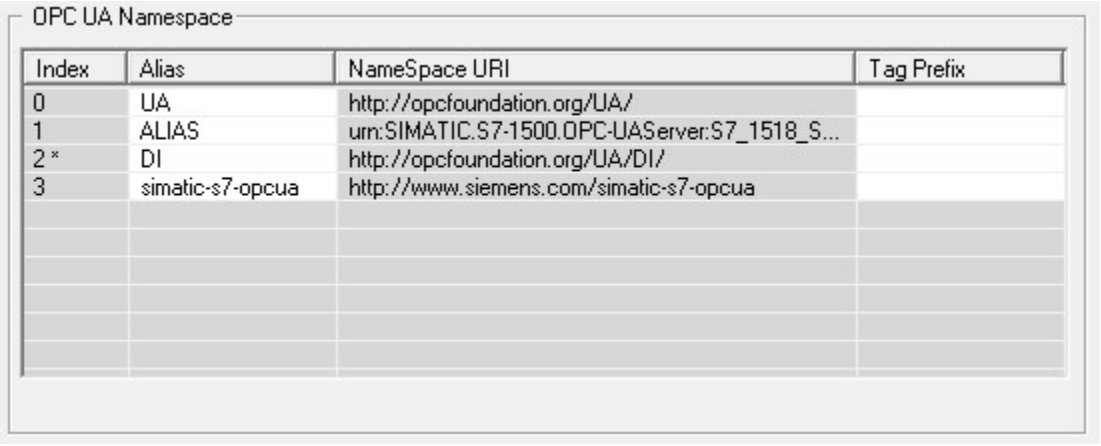
The self-signed certificate has a validity of ten years from the date it is generated and is encoded with the machine name according to the digital certificate requirement in the OPC UA specification.

3. Follow the steps in section 2B.

OPC UA Namespace

The **OPC UA Namespace** section of the OPC UA editor displays the Namespace alias table for the Namespace URIs present in the OPC UA Server to which the OPC UA Client is connected. The OPC UA Namespace is defined and exposed by the OPC UA server. Each namespace is identified by an index ID. You can define an alias name for a namespace according to the following rules:

- The alias name must be defined for the namespace URI exposed by the Gateway Communication Driver.
- The alias name must be in string format.
- There can be no more than one alias per namespace.
- The same alias cannot be given to two OPC UA namespaces.
- The alias must not conflict with any other namespace display name.



The Namespace grid displays the following information:

Column name	Description
Index	Displays the Namespace index for the Namespace URI present in the OPC UA Server. The default Namespace is appended with * after the Index. The items configured in the default Namespace need not be subscribed in the syntax using the alias or the Namespace name.
Alias	Displays the alias name for the Namespace URI available in the OPC UA Server. You can change this name during configuration. The

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

	Alias box cannot be blank. You can use the "_" and "#" special characters in the alias name. You cannot create duplicate alias names.
Namespace URI	Displays the Namespace URI imported from the OPC UA Server. This information cannot be edited.
Tag Prefix	Here you can add the common prefix for an item reference. To add a prefix: Double-click anywhere in the cell under the Tag Prefix column and enter the desired prefix. A maximum of 250 characters is allowed.

To edit the OPC UA namespace

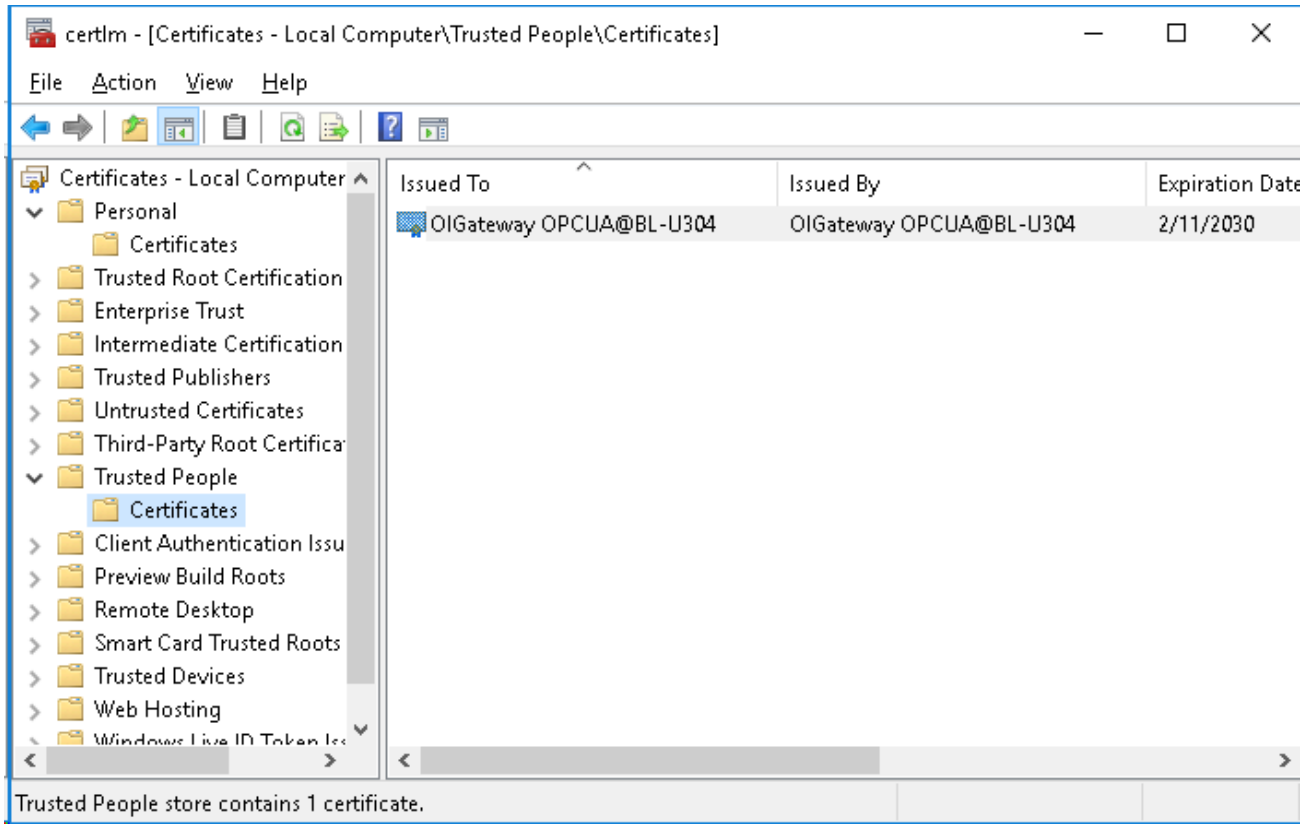
1. Right-click the Namespace URI you want to set as the default, and then select **Set as Default Namespace**. The Index will then be appended with an asterisk (*) to indicate the default Namespace URI.
2. In the **Alias** text box of the selected Namespace, type an alias name for the Namespace URI.
3. **Save** the OPC UA configuration.

Verify OPC UA certificate installation

Verify certificate installation

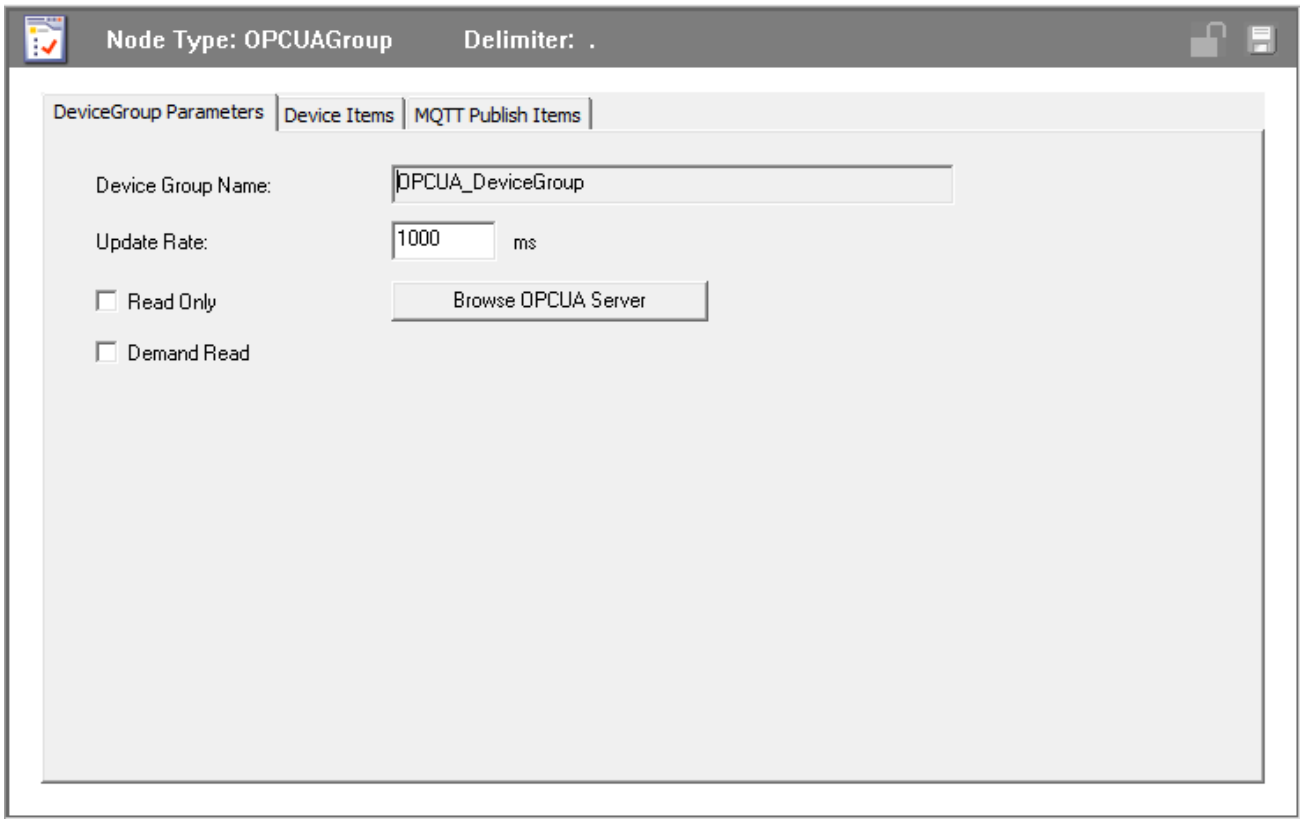
1. Open the certificate manager by typing **certificate** in the Windows search window, then select **Manage Computer Certificates** from the search results. The **Certificate Manager** opens.
2. Navigate the **Trusted People** folder and check that the OPC UA certificate has been installed.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.



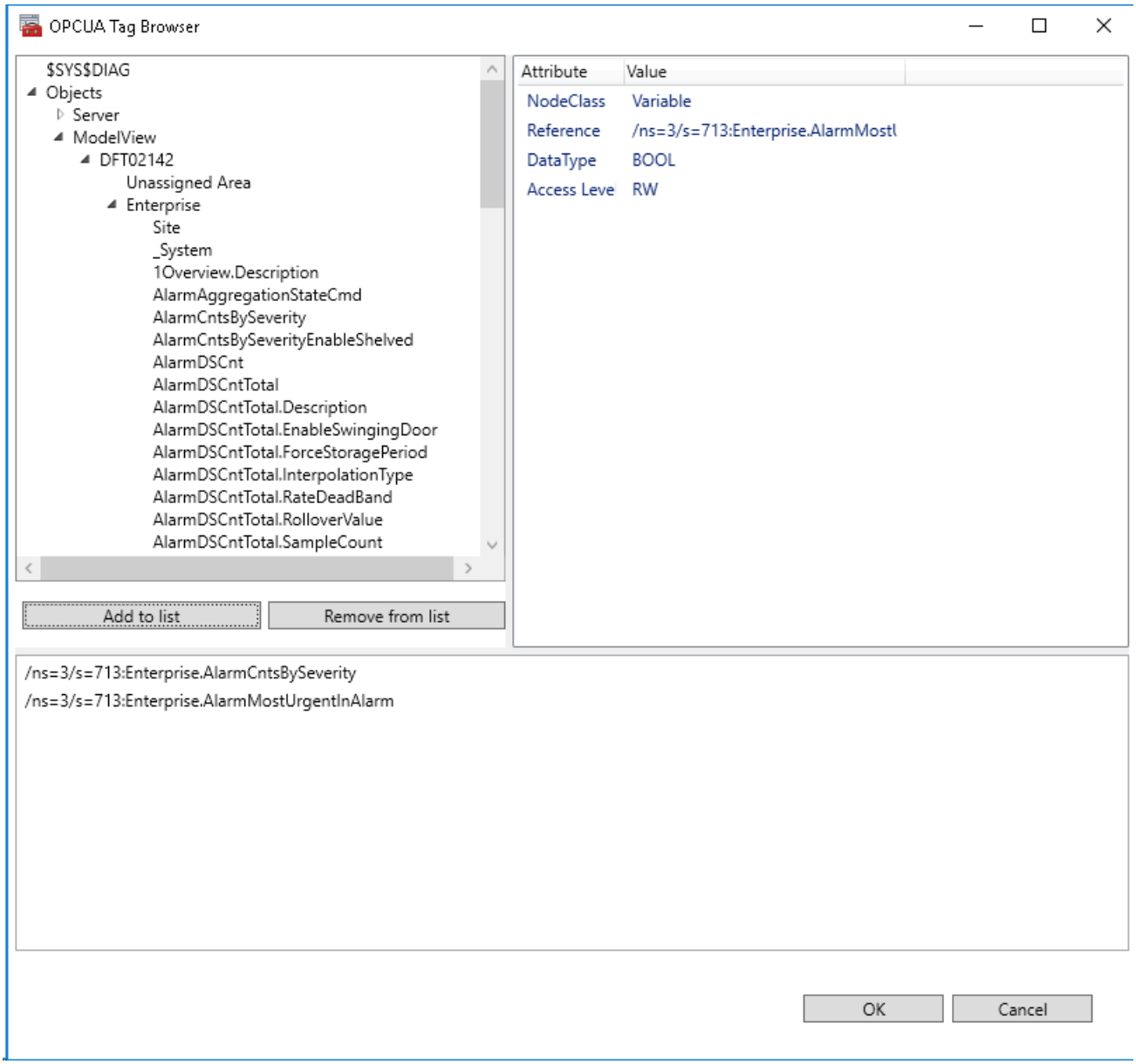
3. From the OPC UA client node, open the **Operations Control Management Console** again, and select the **Test Connection** button in OPC UA parameters window. At the bottom of the window, the **OPC UA Namespace** alias list will be automatically populated, indicating the connection was successful.
4. Add a group under the OPC UA Connection. The device group name prefixes **OPCUA_** to the group name you enter.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.



5. Select the **Browse OPC UA Server** button to browse the OPC UA hierarchy.
6. Optional: Add OPC UA tags to monitor in the alias list to help ensure that you can browse the namespace.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.



7. Switch back to OI Gateway configuration in the **Operations Control Management Console**, and notice that the OPC UA tags are now listed in the **Device Items** window.
8. By default, item names in the list duplicate the complete item reference path. Rename the items as needed to be more user-friendly.

Use an OPC UA client to read and write run-time data

You can leverage the OPC UA protocol to view and change attribute values and quality in a deployed, running galaxy from an external OPC UA node. This node can be an OPC UA connection configured on the Gateway Communication Driver, or a third party OPC UA client. From the OPC Client, you can browse the available namespaces on the run-time node. Note that your permissions in the OPC UA namespace are dependent on how you log in and how the Client Access Rules have been configured in the **Configure Services** dialog.

Prior to using OPC UA for browsing the galaxy namespace, you must configure and deploy the

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

AVEVA.OPCUAService. See Configure and deploy the OPC UA service for details.

Depending on the requirements of your galaxy, you can create additional OPC UA server services. However, each service must have a unique port number.

Once the OPC UA service is configured and deployed on the run-time node, you must:

- Configure the firewall on the run-time node to allow the OPC UA client to access it. For details, see Configure the firewall for the OPC UA service.
- Install a security certificate on the OPC UA client node to ensure secure communications with the run-time node. For details, see Configure an OPC UA data source object

To use OPC UA client/server for reading and writing run-time data

1. Configure certificates for the Gateway Communication Driver OPC UA connection or third-party OPC UA client you will use for browsing the system. Certificates must be installed in the **Trusted People** certificate store.
   - If you log in as anonymous, access to the galaxy data is read-only. See Client access rules and galaxy security for additional information about how Galaxy security affects OPC UA client access.
   - If you log in with a valid username and password, access to the galaxy data is read/write.
   - OPC UA LogIn By Certificate is not supported.
2. Once you are connected to the OPC UA namespace and with the galaxy deployed, you can browse galaxy objects and attributes in both the **Model** and **Deployment** views. If the galaxy is not deployed, you can browse objects but not attributes.
3. If you have logged in with a valid username and password, you can change attribute values. If you are logged in anonymously, you can only view values. However, your access to data is also affected by the Galaxy security configuration and your permissions. For more information about how Galaxy security affects user access through OPC UA, see Client access rules and galaxy security.

Use a script to call an OPC UA server

Starting with System Platform 2023 R2, you can use Application Server object scripts and/or Industrial Graphics scripts to call OPC UA servers.

Method calls are executed in the OPC UA server through the OPC UA client, are are supported through the OPC client in the OI Gateway Communication Driver.

See Introduction to OPC UA scripts in the *Application Server Scripting Guide* for details.

Work with the Industrial Graphic Editor

This section describes how to use Industrial Graphics and the Industrial Graphic Editor to build a graphical interface for monitoring and interacting with Application Objects at run time. It also describes some of the basic steps for using Industrial Graphics with the AVEVA Operations Management Interface (OMI). Topics covered in this section include:

- Industrial Graphic Editor workflows for use with HMI/SCADA applications
- Create a graphic in the IDE

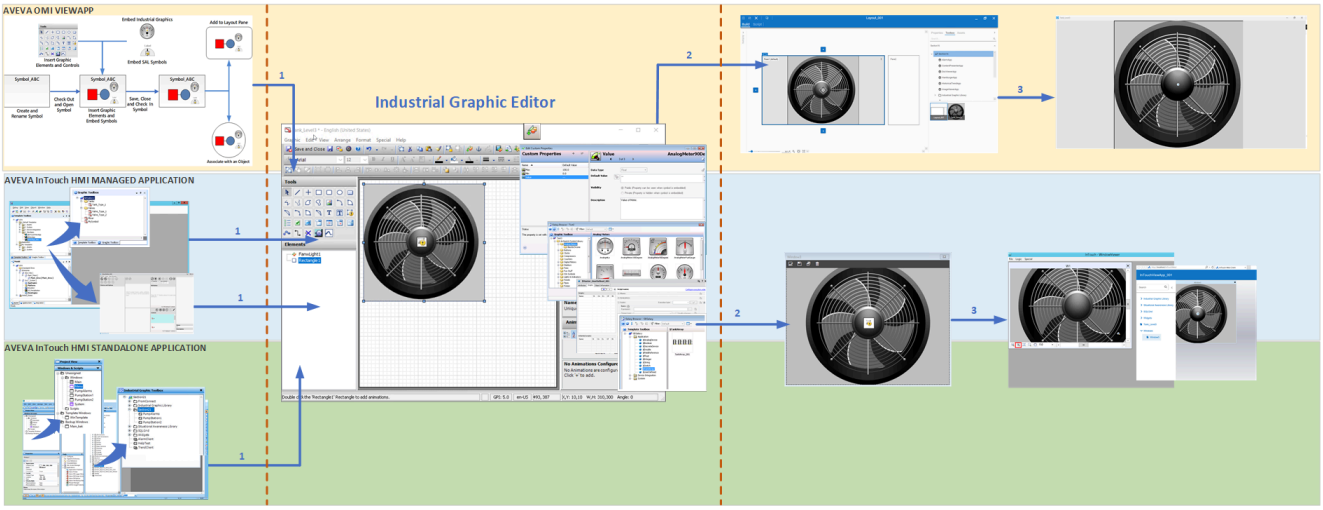
© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

- Add an Industrial Graphic to an AVEVA OMI pane
- Element styles
- Working with the SignedWrite() function for secured and verified writes
- Show/Hide content script functions

For detailed information about Industrial Graphics and the Industrial Graphic Editor, see the *AVEVA Industrial Graphic User Guide* or the Industrial Graphic Editor help.

Industrial Graphic Editor workflows for use with HMI/SCADA applications

The Industrial Graphic Editor is the standard graphic editor used across various HMI/SCADA applications. It can be used to create graphics that can be embedded into HMI/SCADA applications to represent assets on the plant floor. The overall workflow is described in the following sections.



InTouch managed applications

The workflow for adding a graphic to an InTouch HMI managed application starts with the IDE.

1. Create or edit a graphic, either
   a. From the Visualization folder:
      a. Create a new graphic or select an existing graphic.
      b. Open the graphic to launch the Industrial Graphic Editor.

   - or -

   a. From the Object Editor for the object you want to associate the graphic with:
      a. In the Content pane, select the **+** button to add a graphic to the object.
      b. Select the **Edit Content** button to launch the Industrial Graphic Editor.
2. Edit the graphic in the Industrial Graphic Editor. Modify graphical elements as needed:
   - Insert graphic elements and embed graphics.
   - Add/modify scripts, references, custom properties, and animations to enable the graphic to represent

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

the asset, and to respond to commands and data changes at run time.

3. Embed the graphic into a window.
4. To view the graphic in runtime, launch Window Viewer or the InTouch web client.

AVEVA OMI ViewApps

The workflow for adding a graphic to an OMI ViewApp starts with the IDE.

1. To create or edit a graphic, either:
   a. From the Visualization folder:
      a. Create a new graphic or select an existing graphic.
      b. Open the graphic to launch the Industrial Graphic Editor.

   - or -

   a. From the Object Editor for the object you want to associate the graphic with:
      a. In the Content pane, select the **+** button to add a graphic to the object.
      b. Select the **Edit Content** button to launch the Industrial Graphic Editor.
2. Edit the graphic in the Industrial Graphic Editor. Modify graphical elements as needed:
   - Insert graphic elements and embed graphics.
   - Add/modify scripts, references, custom properties, and animations to enable the graphic to represent the asset, and to respond to commands and data changes at run time.
3. Use the Layout Editor to place the graphic in a Layout pane.
4. Deploy the OMI ViewApp. You can then launch the ViewApp using the Application Manager of the OMI web client.

Create a graphic in the IDE

There are three ways to create a graphic in the IDE. Which one to use depends on what objects the you want the graphic to be available to:

- To have the graphic available to all objects, create it in the Visualization Folder
- To have the graphic available to any object created from a template, create the object in the template.
- To create a graphic for use with only one specific object, create it in the object.

The following topics describe each of these methods.

Create graphics in the visualization folder

You can create a new graphic in the Visualization folder. It is then listed in the Visualization folder with a proposed default name. You can:

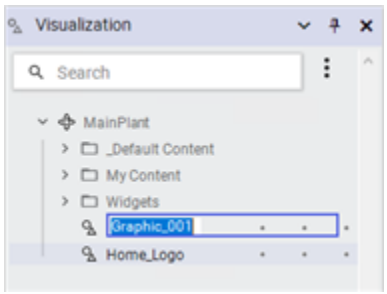
- Rename the graphic.
- Move the graphic.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

- Edit the graphic with the Industrial Graphic Editor.

To create a new graphic from the IDE

1. On the **Home** ribbon, in the **Create** area, select **Graphics**. The **Visualization folder** appears and a new graphic is listed.



Note: You can also press **CTRL + SHIFT + H** to create a new Industrial graphic or right-click in the **Visualization folder** and select **New** and **Graphic** from the shortcut menu.

2. Rename the graphic.

   Names must be unique within the entire Visualization folder hierarchy. Valid characters for graphic names include alphanumeric characters, #, and _ (underscore). Graphic names cannot contain spaces and the graphic name cannot begin with the $character.
3. Double-click the graphic name. The Industrial Graphic Editor appears.
4. Draw the graphic.

Create graphics in AutomationObject templates

You can create a graphic from the Object Editor **Attributes** tab in an AutomationObject template. Creating a graphic this way automatically associates the new graphic with the AutomationObject.

To create a new graphic for an AutomationObject template

1. Open the AutomationObject template. Click the **Attributes** tab. Any local and inherited graphics are listed.
2. In the **Content** area, click the **Add** (+) icon.
3. Give the new graphic a name. Names must be unique. Valid characters for graphic names include alphanumeric characters, $, #, and _ (underscore). Graphic names cannot include spaces and cannot begin with the $ character.
4. If needed, type the description of the graphic in the **Description** box.
5. Make sure the graphic is selected and click the **Edit** icon. The Industrial Graphic Editor opens.
6. Draw the graphic.

Create graphics in AutomationObject instances

You can create a graphic from the **Attributes** tab in an AutomationObject instance. Creating a graphic this way automatically associates it with the AutomationObject instance.

Note: AutomationObjects can also inherit graphics from their parent template. You can only view an inherited graphic in read-only mode. Inherited graphics can only be removed or edited in the template where they are

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

defined.

To create a new graphic for an AutomationObject instance

1. Open the AutomationObject instance. Click the **Attributes** tab. Any local and inherited graphics are listed.
2. In the **Content** area, click the **Add** (+) icon.
3. Give the new graphic a name. Names must be unique. Valid characters for graphic names include alphanumeric characters, $, #, and _ (underscore). Names cannot include spaces and cannot begin with the $ character.
4. If needed, type the description of the graphic in the **Description** box.
5. Make sure the graphic is selected and click the **Edit** icon. The Industrial Graphic Editor opens.
6. Draw the graphic.

Graphic Change Propagation

If the graphic is hosted by the Visualization folder and edited, all graphics hosted by AutomationObject templates and instances that contain embedded instances of this graphic are also updated.

If the graphic is hosted by an AutomationObject and edited, all graphics hosted by derived AutomationObjects are also updated.

Add an existing graphic to your graphic

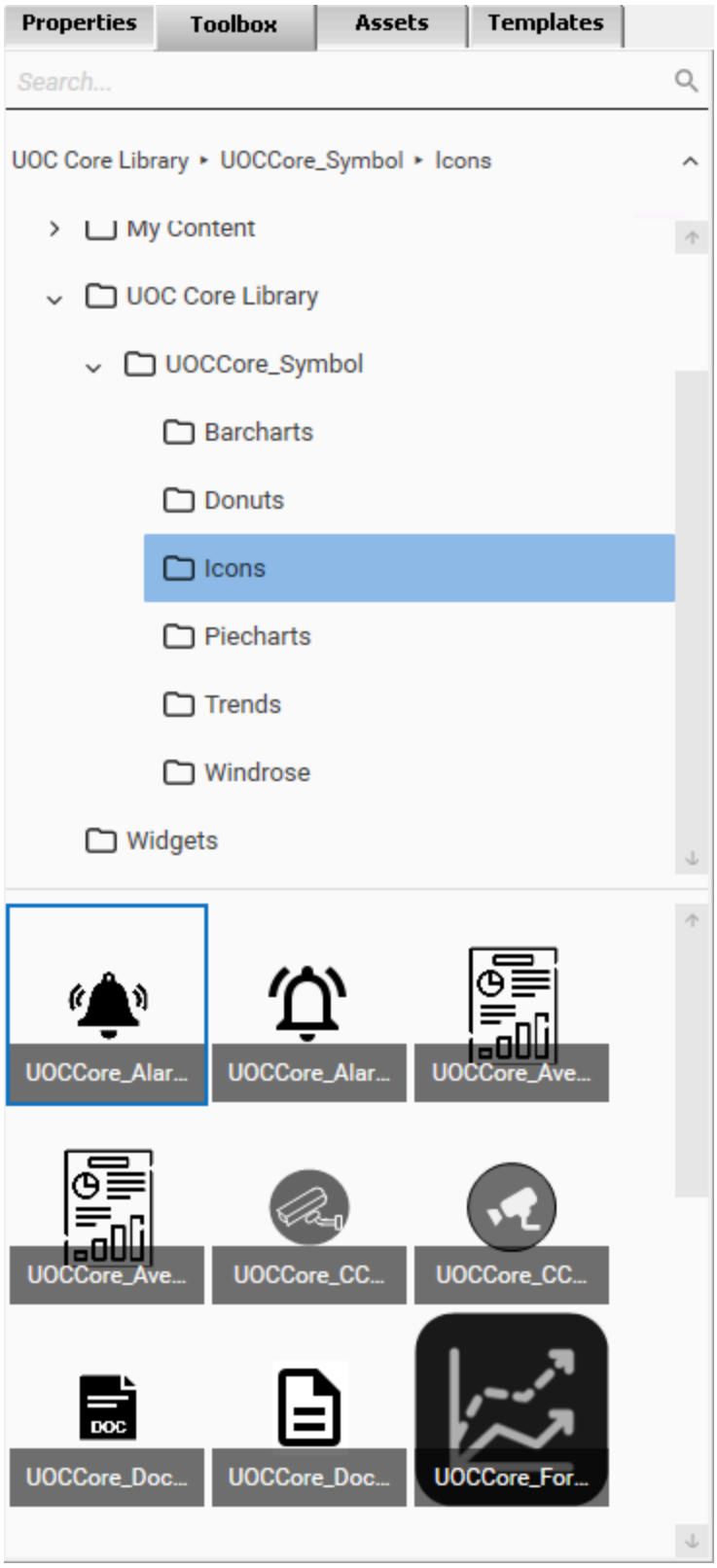
You can add an existing graphic to the graphic you are editing. The graphic may come from the Visualization Toolbox or an instance (asset). Once you have placed the graphic in your graphic, you can then edit it like any other component of the graphic.

Note: If you place a graphic from a template onto the graphic you are editing, the **Configure New Asset** Editor opens so you can create a new instance based on the template. You will generally only do this if the template contains an Object Wizard, See Create a new instance graphically for more information.

To add an existing graphic to the graphic you are editing

1. In the **Properties** pane, select the tab for the location of the graphic to add: the Visualization Toolbox or an asset (instance).
2. Navigate through the toolbox folders or your Galaxy structure to find the location of the graphic. The graphics in the folder or object you select appear below the navigation area.
3. Select the graphic you want to add and drag it to the desired location on the canvas.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

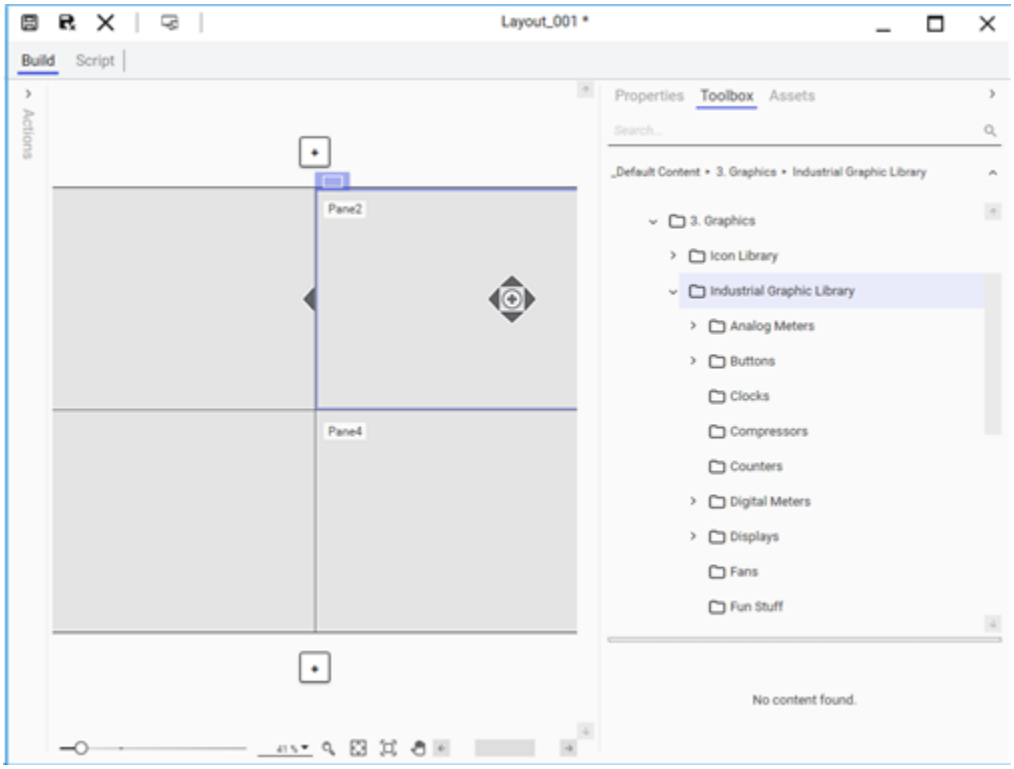


© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Add an Industrial Graphic to an AVEVA OMI pane

You can add an Industrial Graphic to an AVEVA OMI pane simply by dragging and dropping a thumbnail image onto a pane.

1. Select the **Graphics** folder from the **Toolbox** tab of the Layout editor.
2. Navigate to the **_Default Content** folder, then select **Graphics**.
3. Expand the **Industrial Graphics Library**.



4. Select the subfolder in the Industrial Graphics Library that contains the family of industrial graphic symbols you wish to use.
5. Symbol thumbnails appear below the list of symbols in the **Toolbox**.
6. Drag and drop the thumbnail image you wish to use onto the appropriate pane in the layout.

About graphic scaling

By default, a graphic attempts to scale to the size of its hosting pane and maintain the graphic's horizontal and vertical aspect ratio. The size of a graphic created using the Industrial Graphic Editor is determined by a bounding rectangle around all graphic elements and embedded graphics that compose the graphic.

When a graphic is placed in a layout pane, if the graphic's bounding rectangle exceeds the size of its hosting pane, the bounding rectangle becomes smaller by reducing the original size of its contents. If the graphic's bounding rectangle is smaller than the pane size, the bounding rectangle increases and enlarges the original size of its contents.

Graphics that have a defined fixed size render in runtime identically to graphics without a fixed size. A graphic's size adjusts to the size of the pane while maintaining its aspect ratio as appropriate. Most graphics that have a

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

configured fixed size are typically placed in a specific target pane of a known size that enables them to be the rendered exactly as designed in the Industrial Graphic Editor without any required resizing during configuration or runtime.

Additional information about Industrial Graphics

For more information about working with Industrial Graphics in the layout editor, see Graphics in the AVEVA OMI Help.

Element styles

An element style defines a set of visual properties that determine the appearance of text, lines, graphic outlines, and interior fill shown in Industrial Graphics. An element style that is applied to a graphic sets pre-configured visual property values that take precedence over a graphic's native visual properties.

Element styles provide the means for developers to establish consistent visual standards in their applications. An element style can define the same visual properties of text, lines, fill, and outlines for all graphics that belong to an application.

Likewise, element styles can show the current status of an object represented by a graphic. For example, an element style animation can be applied to a graphic when an object transitions to an alarm state.



Style libraries

A typical Style library includes a set of predefined element styles.

The predefined values of the element styles in this library can be changed. However, existing element styles cannot be renamed or deleted. Also, new element styles cannot be added to the library.

Visual properties defined by element styles

The following table lists the visual properties of graphic elements defined in an element styles.

Graphic element	Element properties
Text	• Font family • Font size

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

	• Font style • Font color • Blink On/Off
Fill	• Fill color • Fill gradient • Fill pattern • Fill texture • Blink On/Off
Line	• Line pattern • Line weight • Line color • Blink On/Off
Outline	• Outline Show/Hide • Outline Pattern • Outline Weight • Outline Color • Blink On/Off

An element styles may not define every visual property. If a property value is not defined in an applied element styles, the element's native style is used and can be changed. However, if an element's property value is defined in an applied element styles, the element's native properties are disabled and cannot be changed.

Element styles in animations

You can configure an element or a group of elements with Boolean or truth table animations that determine whether element styles are applied based on evaluated conditions or expressions. See Use element styles to configure an animation .

Property style order of precedence

To understand the behavior of an element's properties when an element style is applied, we recommend that you understand the order of precedence for the levels at which property styles are applied. See Order of precedence for property styles for the list.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Update element styles at application run time

You can update the element styles applied to graphics included in a running application.

- Updating element styles from your HMI/SCADA development environment

  When an application is deployed and updates were made to the applied element styles from the development environment, those updates will be propagated to the graphic elements in a running application without requiring your run-time viewer to be closed and re-opened.
- Importing an updated Graphic Style Library

  Importing an updated Graphic Style Library that includes different applied element styles will propagate those changes to graphic elements in a running application without requiring your run-time viewer to be closed and re-opened.

Manage element styles

You can import and export Style Libraries containing custom element styles from one application to another remote application. This section describes the tasks to create a set of custom element styles that can be used in other HMI/SCADA applications.

Import and export element style libraries

You can import a application's Style Library to load its element styles into another application. You can also modify element styles, and then export the element style Library as custom user-defined element styles.

Change the visual properties of an element style

You can modify the visual properties of any element style in the currently loaded Styles Library. You modify properties by setting overrides on the **Element style overrides** dialog box for the style you want to change.

In the **Element style overrides** dialog box, you can:

- Modify the appearance of text by setting overrides for the text font, text size, text style, text color, and blinking.
- Modify the appearance of graphic fill by setting overrides for fill color and blinking.
- Override the appearance of the line pattern, weight, color, and blinking.
- Override the appearance of the outline line pattern, weight, color, and blinking.
- Preview the appearance of an element style.
- Reset element style visual properties to their default values.

To show the Element style overrides dialog box

1. Navigate to the Style Library configuration dialog.

   The navigation path will differ according to your HMI/SCADA software. Typically, the path will be through menu items to configure your project or application components.
2. Click the tab for the type of element style you want to modify.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

3. Click the edit icon for the desired style.

   The **Element style overrides** dialog box includes the following:

   - The style name.
   - The preview area at the top right showing what the element style looks like with its current settings.
   - The **Reset to default** button returns the element style to its default values.
   - Property tabs where you can set values for each property defined in the selected element style.

Override element style text properties

You can modify an element style's text visual properties by setting alternative values for text font, text color, text style, and blink rate.

To change the appearance of text in an *element style*

1. Navigate to the Style Library configuration dialog.

   The navigation path will differ according to your HMI/SCADA software. Typically, the path will be through menu items to configure your project or application components.
2. Select the edit icon for an element style to show the **Element Style Override** window.
3. Click the **Text** tab.
4. To change the font, select **Font Override**, click the browse button, and select a font from the **Font** dialog box.
5. To override the font color:
   a. Select **Font Color Override**.
   b. Click the color box.
   c. Select a color from the **Select Font Color** dialog box.
6. To override the text blink behavior:
   a. Select **Blink**.
   b. Select a blinking speed from the **Speed** list.
   c. Click the color box to show the **Select Blink Color** dialog box.
   d. Select the color, gradient, pattern, and texture for the blink style.
7. Click **OK**.

Override element style fill properties

You can modify an element style's fill visual properties by setting alternative values for fill color and blink rate.

To override the fill appearance of an element style

1. Navigate to the Style Library configuration dialog.

   The navigation path will differ according to your HMI/SCADA software. Typically, the path will be through menu items to configure your project or application components.
2. Select the edit icon for an element style to show the **Element Style Override** window.
3. Click the **Fill** tab.
4. To override the fill style:

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

a. Select **Fill Color Override**.
b. Click the color box.
c. Select a style from the **Select Fill Color** dialog box.

5. To override the fill blink behavior:
   a. Select **Blink**.
   b. Select a blink speed from the **Speed** list.
   c. Click the color box.
   d. Select a style from the **Select Fill Color** dialog box.
6. Click **OK**.

Override element style line properties

You can modify an element style's line visual properties by setting alternative values for line color, line pattern, and line weight

To override the line appearance of an element style

1. Navigate to the Style Library configuration dialog.

   The navigation path will differ according to your HMI/SCADA software. Typically, the path will be through menu items to configure your project or application components.
2. Select the edit icon for an element style to show the **Element Style Override** window.
3. Click the **Line** tab.
4. To override the line pattern, select **Line Pattern Override** and select a line pattern from the adjacent list.
5. To override line weight, select **Line Weight Override** and enter a new line weight in the adjacent box.
6. To override line color properties:
   a. Select **Line Color Override**.
   b. Click the color box.
   c. Select a color style from the **Select Line Color** dialog box.
7. To override the line blink behavior:
   a. Select **Blink**.
   b. Select a blinking speed from the **Speed** list.
   c. Click the color box.
   d. Select a style from the **Select Blink Color** dialog box.
8. Click **OK**.

Override element style outline properties

You can modify an element style's outline visual properties by setting alternative values for text font, text color, text style, and blink rate.

To override the outline appearance of an element style

1. Navigate to the Style Library configuration dialog.

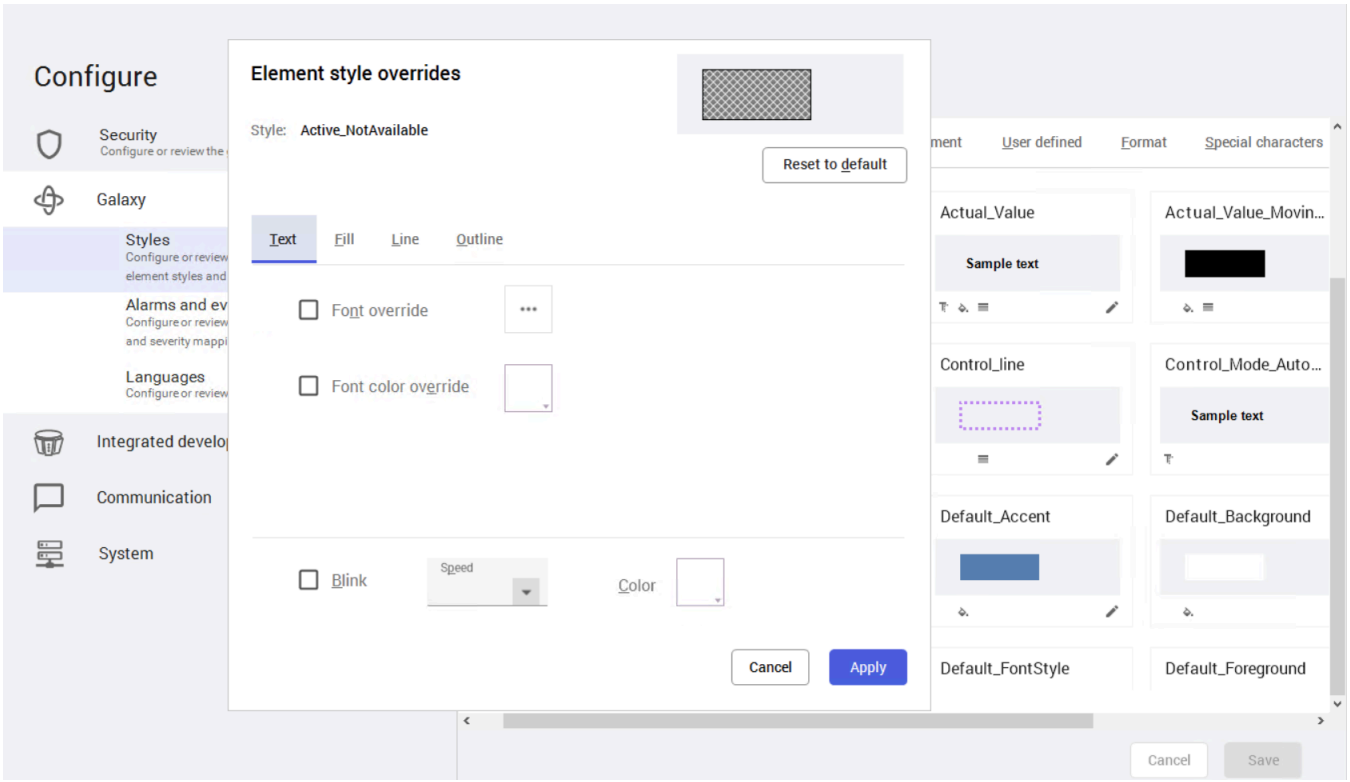
© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

The navigation path will differ according to your HMI/SCADA software. Typically, the path will be through menu items to configure your project or application components.

2. Select the edit icon for an element style to show the **Element Style Override** window.
3. Click the **Outline** tab.
4. Select **Show Outline.**
5. To set the line pattern, select **Line Pattern** and select a line pattern from the adjacent list.
6. To set the line weight, select **Line Weight** and type a line weight in the adjacent box.
7. To set the line style:
   a. Click the color box next to **Line Color.**
   b. Select a style from the **Select Line Color** dialog box.
8. To set the line blink behavior:
   a. Select **Blink.**
   b. Select a blinking speed from the **Speed** list.
   c. Click the color box.
   d. Select a blink style from the **Select Blink Color** dialog box.

Preview an element style

The **Preview** area shows the appearance of an element style's current assigned visual styles.



To preview an element style

1. Navigate to the Style Library configuration dialog.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

The navigation path will differ according to your HMI/SCADA software. Typically, the path will be through menu items to configure your project or application components.

2. Select the **Styles** tab.
3. Select an element style to show the **Element Style Overrides** window.

   The **Preview** field updates to show the appearance of the selected element style.

Reset an element style to its default values

You can reset an element style to its original default visual styles.

Resetting an element style resets visual styles to their original default styles, not to any previous override settings.

To reset an element style

1. Navigate to the Style Library configuration dialog.

   The navigation path will differ according to your HMI/SCADA software. Typically, the path will be through menu items to configure your project or application components.
2. Select the **Element Styles** tab.
3. Select one or more element styles from the **Element Style Overrides** list.
4. Click **Reset to Default**. The selected element style properties are reset to their default styles.

Change the visual properties of user-defined element styles

The Style Library includes a set of 25 user-defined element styles. User-defined element styles appear towards the bottom of the list of the **Element Style Overrides** field and are named User_Defined_01 to User_Defined_25.

All visual properties of user-defined element styles are initially set to default values. Visual properties can be individually configured for each user-defined element style by setting overrides for text, fill, line, and outline as other predefined element styles.

Apply element styles to elements

You can apply element styles to one or more graphic elements. Unlike setting element style overrides that change the appearance of an element style's properties, applying an element style to a graphic element overrides the element's native properties.

Applying element styles to graphic elements can help standardize the appearance of those elements and show the current state of an object represented by a graphic. For more information, see Change the visual properties of an element style.

Element style list

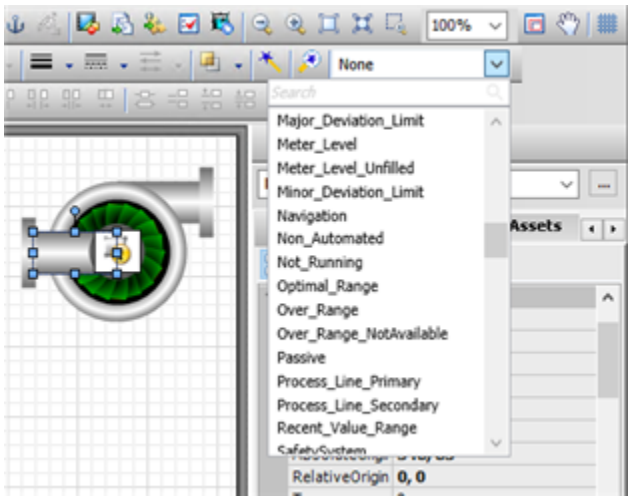
The Industrial Graphics Editor menu bar contains an **Element Style** list to select an element style and apply it to a selected element of a graphic.

To apply an element style to a graphic element

1. Open the graphic in the Industrials Graphic Editor.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

2. Select one or more elements from the graphic.
3. Select an element style from the **Element Styles** list to apply to the selected elements.



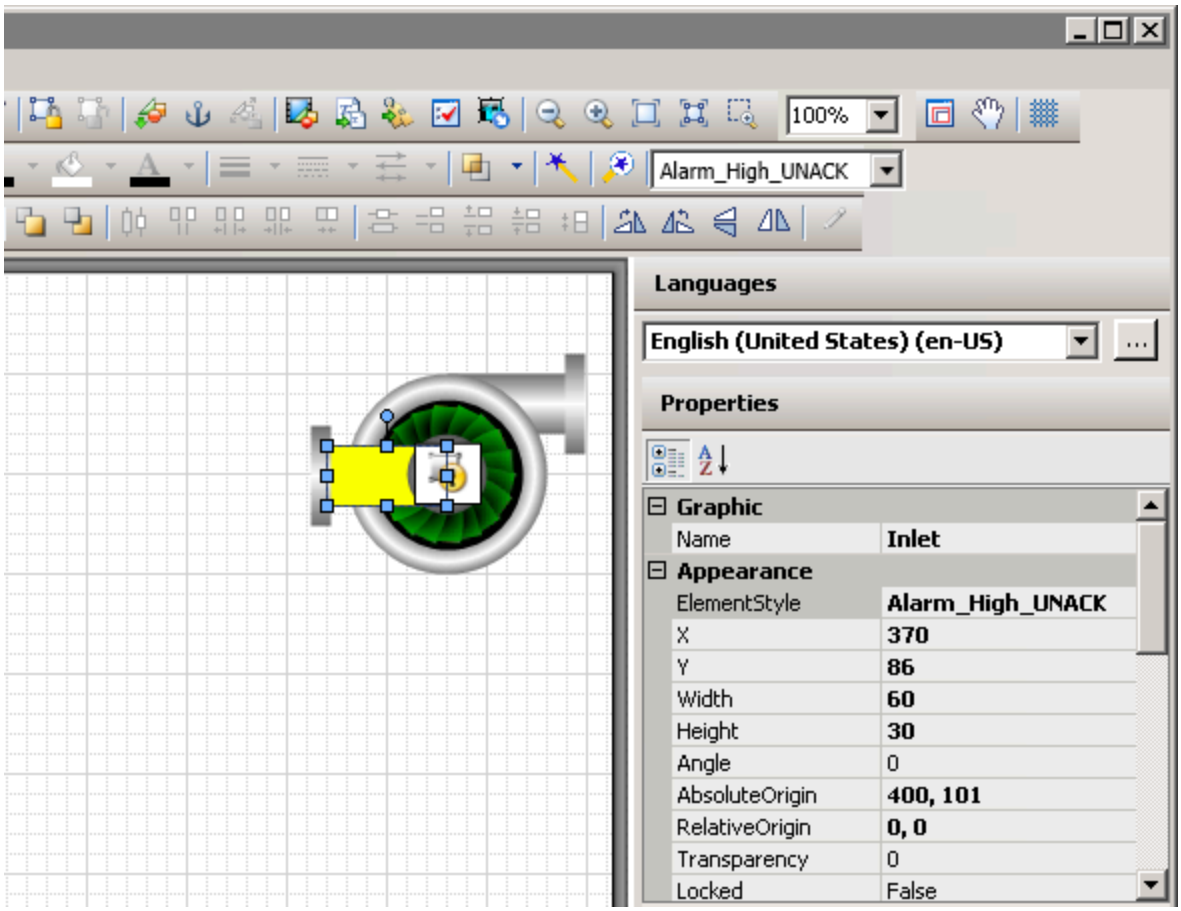
Properties grid

The Industrial Graphics Editor **Properties** view contains an **ElementStyle** Appearance item to select an element style and apply it to a selected element of a graphic.

To apply an element style from the properties editor

1. Open the graphic in the Industrial Graphic Editor.
2. Select one or more elements from the graphic.
3. In the **Appearance** category of the **Properties** Editor, select an element style from the **Element Style** list.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.



Format painter

You can use the Format Painter in the Industrial Graphics Editor to copy an element style from one graphic element to another.

To apply an element style using format painter

1. Open a graphic in the Industrial Graphics Editor.
2. Select the element with the element style you want to copy.
3. On the **Edit** menu, click **Format Painter**. The pointer appears as the Format Painter cursor.
4. Select the elements you want to apply the element style to. The element style is applied to the selected elements.

Clear an element style

When an element style is applied to an element, you cannot edit the element's styles that are controlled by the applied element style. However, you can clear the application of the element style so that all of the styles can be edited.

To clear an element style

1. Select the element.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

2. Select **None** in the element style list.

Set an element style as a default for a canvas

You can select an element style at the canvas level of the Industrial Graphics Editor. The selected element style is applied to any graphic element or groups that you create on the canvas.

Apply element styles to groups of elements

You can apply an element style on a group of elements in the same way that you apply an element style to a single graphic element. To enable group support for element styles, set the group's run-time behavior to TreatAsIcon.

Set run-time behavior for a group to TreatAsIcon

To apply an element style to a graphic element group, set the group's **TreatAsIcon** property to True. Otherwise, the element style lists are disabled when an element group is selected.

To set the TreatAsIcon property to true

1. Select the element group to which the element style will be applied.
2. On the **Properties** menu, click **Run-time Behavior** and click **TreatAsIcon**.
3. Select **True** from the drop-down list.

Element style behavior with a group of elements

- The element style applied to a group has higher precedence than the property styles applied to individual graphic elements in the group.
- If the element style applied to a group of elements has undefined property styles, then the element continues to use its element style or element-level settings for undefined property styles.
- If the element style that is applied to a group of elements has defined property styles, then those property styles override the property styles defined at the element level for elements in the group.
- An element style cannot be applied to a nested element group.
- If you add an element to a group that has a group-level element style applied, the group element style is applied to it.

Use element styles to configure an animation

You can configure an element or a group of elements with a:

- Boolean animation that applies element styles based on a binary True/False condition.
- Truth table animation that applies element styles based on a range of possible values.
- Bit State animation that applies element styles based on a bit state based conditions.

The truth table animation that applies element styles:

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

- Associates expressions of any data type supported by Application Server or InTouch to an element style.
- Defines as many conditions as required and applies a separate element style for each condition
- Defines the conditions to apply an element style by specifying a comparison operator (=, >, >=, <, <=) and a breakpoint, which itself can be a value, an attribute reference, or an expression.
- Arranges conditions in the order that element styles are processed.

Use element styles to configure a boolean animation

You can configure an element or a group of elements with a Boolean animation that uses only two element styles.

To configure an element or a group of elements with an element style that uses boolean animation

1. Open the graphic in the Industrial Graphics Editor.
2. Select the element or element group.
3. On the **Special** menu, click **Edit Animations**. The **Edit Animations** dialog box appears.
4. Click the **Add** icon and select **Element Style**. The Element Style animation is added to the Animation list and the **Element Style** state selection panel appears.
5. Click the **Boolean** button. The **Boolean Element Style** configuration panel appears.
6. In the **Boolean** text box, enter a Boolean numeric value, attribute reference, or an expression.
7. Clear **ElementStyle** in the **True, 1, On** area or **False, 0, Off** area if you don't want a different element style for the true or false condition than the default element style that is shown in the **Element Style** list.
8. In the **True, 1, On** area, select the element style to use when the expression is true.
9. In the **False, 0, Off** area, select the element style to use when the expression is false.
10. Click **OK**.

Use element styles to configure a truth table animation

You can configure an element or a group of elements with a Truth Table animation that can select multiple element styles based on a set of evaluated values or expressions.

Truth Table animation is typically used to set element styles for the different states of an object. For example, you can set Truth Table conditions to show different element styles that represent the following alarm conditions:

- When the attribute TankLevel_001.PV is 0 then no element style is applied.
- When the attribute TankLevel_001.PV is less than 20, then the element style is Alarm_Minor_Dev.
- When the attribute TankLevel_001.PV is greater than the attribute Standards.TankMax then the element style is Alarm_Major_Dev.

To configure an element or a group of elements with an element style that uses truth table animation

1. Open the graphic in the IDE Industrial Graphics Editor.
2. Select the element or group.
3. On the **Special** menu, click **Edit Animations**. The **Edit Animations** dialog box appears.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

4. Click the **Add** icon and select **Element Style**. The Element Style animation is added to the Animation list and the **Element Style** state selection panel appears.
5. Click the **Truth Table** button. The **Truth Table Element Style** configuration panel appears. The element style that is applied to the element is shown in the **Element Style** list at the bottom of the panel.
6. In the **Expression Or Reference** area:
   - Select the data type of the expression from the list.
   - Type a value, attribute reference or expression in the text box.
7. If the data type of the expression is string or internationalized string, you can specify to ignore the case by selecting **Ignore Case.**
8. In the **Truth Table**, select the **Element Style** check box and select the element style for one of the conditions to be defined in the truth table.
9. In the **Operator** column, select a comparison operator.
10. In the **Value or Expression** column, type a value, attribute reference, or expression.
11. To add other conditions:
    a. Click the **Add** icon. An additional condition is added to the truth table.
    b. Select the **Element Style** check box, select the element style for the condition, select an operator, and enter the condition value or expression.
12. After adding all truth table conditions, click **OK**.

Delete a condition from an animation truth table

You can delete a condition from an animation Truth Table to remove the associated element style from the animation.

To delete a condition from a truth table animation that uses element styles

1. Open the **Edit Animations** dialog box, **Truth Table Element Style** panel.
2. Select the condition you want to delete.
3. Click the **Remove** icon. The condition is removed.

Change the processing order of element styles in a truth table animation

You can change the processing order of element styles by moving the conditions up or down in the Truth Table list. The element style at the top of the Truth Table list is processed first. The remaining element styles are processed in order based on their position from the top of the list.

To change the processing order of element style conditions

1. Open the **Edit Animations** dialog box, **Truth Table Element Style** panel.
2. Select the condition you want to move up or down the condition list in order for it to be processed sooner or later.
3. Click the:
   - **Arrow up** icon to move the condition up in the truth table.
   - **Arrow down** icon to move the condition down in the truth table.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Using custom properties with Application Server

This section provides information specific to Application Server for using custom properties in Industrial Graphics. You can use custom properties to extend the functionality of graphics at run time, such as showing historical data, and to change references dynamically through binding and scripting.

You can use custom properties to assign values to attributes or tags. In AVEVA InTouch HMI, you can use scripts to assign bad or initializing quality values to tags. In AVEVA OMI ViewApps, you cannot use scripts to assign bad or initializing quality value to attributes of user defined objects.

Topics covered in this section include:

- Custom property examples
- Use custom properties to show historical summary data
- Use binding in custom properties
- Change the expression or reference of a custom property at runtime

For additional background information, see Using Custom Properties in the *Industrial Graphic Editor User Guide*.

Custom property examples

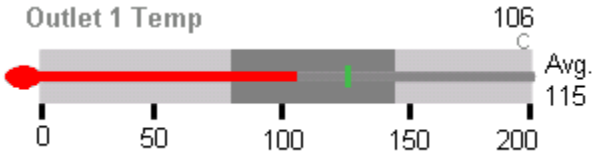
Possible uses for using custom properties are:

- A "TankLevel" custom property of type Writable Attribute can be given a value of "me.pv".
- A "MaxFillLevel" Custom Property of type Expression can be given a value of "Me.MaxCapacity - 200".

Use custom properties to show historical summary data

You can add a custom property to reference historical data collected over a specified period during runtime. The Historian can transform this data to create a set of analog or state statistics that can be shown by Industrial graphic animation during runtime.

For example, consider an example of a temperature meter graphic that shows an optimal temperature range and you would like to know what the average temperature has been over the last 15 minutes. You can add a Value Display animation that shows the average temperature derived from analog temperature data saved in the Historian and referenced by a custom property.



Analog statistical summary data

A custom property analog reference can subscribe to statistics from analog data collected over a defined summary period and saved to the Historian. The following table lists the analog historical statistical data that can be specified for a custom property.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Analog Statistical Data	Description
Average	A time-weighted average calculated from values within a summary period. The average is calculated at the end of the summary period.
Count	A value count calculated from values within a summary period. The count is calculated at the end of the summary period.
First	The first value that occurs within a summary period based on the actual timestamp within the summary period.
Integral	An integral value calculated from values within a summary period. The integral is calculated at the end of the summary period.
Maximum	The first maximum value that occurs within a summary period.
Minimum	The first minimum value that occurs within a summary period.
PercentGood	The ratio of labeled "good" quality data to all data within the summary period. The ratio is expressed as a percentage in the range 0 to 100. PercentGood is calculated at the end of the summary period.
StdDev	Time weighted standard deviation calculated from values within a summary period. The value is calculated using time weighted sums (Integrals) and time weighted sums of squares (IntegralOfSquares) values.
Last	The last value that occurred in the summary period based on the actual timestamp within the summary period.

State statistical summary data

The Historian stores and retrieves values, where every value gets stored with some timestamp and associated quality. This triplet of value, timestamp, and quality is called VTQ and constitutes the smallest addressable piece of data in the Historian data model.

State summary statistics summarize the states of a value. Four different state value data types are possible: analog (integer), Boolean, string, and Null.

Note: Possible Boolean state values are 0 or 1. True or False strings are not supported.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

A custom property state reference can subscribe to state statistics from the Historian as static text, an expression or reference, an aggregate function name, minutes, and state value.

The Historian returns the VTQ for one cycle of a specified state. The quality returned is always OpcQuality. The time returned is always the summary period start time. Value and Time differ based on the aggregate function.

The following table lists state historical statistical data that can be specified for a custom property.

State Statistical Data	Description
Average	Average time a state occurred and completed within a summary period. A partial state within a summary period is ignored for an average calculation. (StateTimeAvgContained)
Minimum	Minimum time a state occurred and completed within a summary period. A partial state is ignored. (StateTimeMinContained)
Maximum	Maximum time a state occurred within a summary period. (StateTimeMax)
Count	Number of times a state occurred and completed within a summary period. A partial state is not counted. (StateCountContained).
Percent	Percentage of the summary period that a state occurred. (StateTimePercent)
Total	Total time a state occurred within a summary period. (StateTimeTotal)

Historical summary period

An Industrial graphic custom property can show historical statistics from Historian data over a defined summary period. During design time, you must specify the summary period to collect Historian summary data by entering values for the **Duration** and **StartTime** options shown on the **Edit Custom Properties** dialog box.

These assigned values are passed as input parameters of the new custom property. A value in minutes must be assigned to the **Duration** option.

The **StartTime** option can be left blank. Auto refresh is applied if a **StartTime** value is not specified.

- If a start time is not specified, then the start and end times of the summary period are calculated as:

  Start Time = Current Time - Duration

  End Time = Current Time
- If a start time is specified, then the start and end times of the summary period are calculated as:

  Start Time = StartTime option value

  End Time = StartTime + Duration

The **Duration** option can accept a negative number when a start time is specified. When **Duration** is assigned a

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

negative number, the start time input parameter value becomes the end time of the summary period. The start time is calculated using the formula shown below:

End Time = StartTime assigned option value

Start Time = End Time + Duration (in this case it is negative value)

Duration can accept values from 1 minute to 10080 minutes, which is one week. **StartTime** must be within datetime Min and Max Value. During runtime, history summary data is auto refreshed at an interval that is 25 percent of its duration length when a **StartTime** value is not specified.

Show statistical summary data

The following procedure explains how to specify custom property options to subscribe to historical statistical data. Before completing this procedure, you must have object or other data that is saved to the Historian and enabled for statistics.

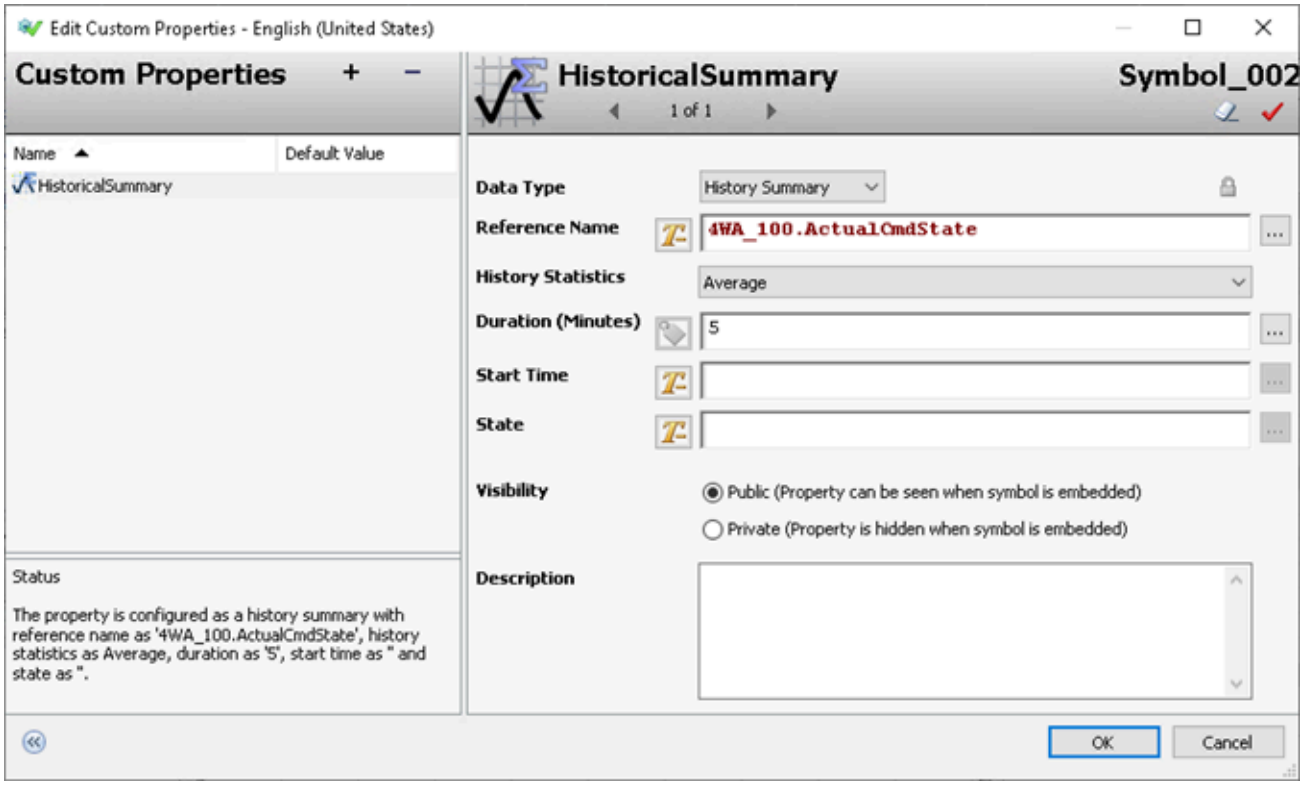
To show historical summary data using custom properties

1. Open the Industrial Graphic Editor.
2. Add a custom property to an Industrial graphic.
   a. Click the canvas to cancel any selected elements.
   b. On the **Special** menu, click **Custom Properties**. The **Edit Custom Properties** dialog box appears.
   c. Click the **Add** icon. A new line is added in the custom properties list.
   d. Type a name for the new historical summary custom property and press **Enter**.
   e. You can see the names of the graphic and custom property in the header of the dialog box.
3. Select **HistorySummary** from the **Data Type** field.

   Important: The History Summary data type only works with Application Server object attributes intended for AVEVA OMI ViewApps or InTouch HMI managed applications.

   The **Edit Custom Properties** dialog box updates to show fields specific to the **HistorySummary** data type.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.



4. Enter a reference to data saved in the Historian in the **Reference Name** field.

   The icon to the left of the **Reference Name** field toggles input to the field as **Static Text** or **Expression or Reference** mode.

   Auto-Detect

   The Historian server is auto-detected from the AppEngine on which the reference attribute is running. For example, if the **Reference Name** field is set to UDO.UDA1, the reference is set to the Historian server name configured for the AppEngine on which UDO is running.

   Expression

   When an expression or reference is typed in the **Reference Name** field, a connection is made to the specified Historian Server. The reference can be an external reference like an object attribute or a custom property.

   Note: A reference cannot be made to historical data from an InTouch tag.

5. Select the type of historical statistics by selecting an option from the drop-down list of **History Statistics.**

   Average is the default type of historical statistic. The following table shows the historical statistics options for analog and state summary data.

Historical Statistics	Analog Historical Data	State Historical Data
Average	I	I
First	I	
Minimum	I	I
Maximum	I	I

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Historical Statistics	Analog Historical Data	State Historical Data
Count	I	I
StdDev	I	
Integral	I	
PercentGood	I	
Percent		I
Total		I
Last	I	

6. Set the length of the summary historical period in minutes by entering a value in the **Duration** field.

   Acceptable **Duration** values are from 1 to 10080 and the default is 5. Duration can be specified as an integer, an expression, or a reference. For more information about possible **Duration** values, see Historical summary period.

7. Set the start time of the of the summary period in the **StartTime** field.

   A start time can be specified as static text, an expression, or a reference. The default start time is the current time.

   A time for **StartTime** is optional and can be left blank. Auto refresh is applied if a **StartTime** value is not specified.

   For more information about setting a start time, see Historical summary period.

8. Set the type of Historian summary data in the **State** field.

   A **State** value can be expressed as an integer constant, static text, an expression, or a reference.

   If a string value is provided, then string state summary data is queried from the Historian. If an integer value is entered, the Historian query is for analog state summary data. If a Boolean state summary value is provided, the **State** value must be 0 or 1.

   State can be left empty. If empty, the default query is for analog summary data.

   To get summary historical data for a Null state, enter "NULL" in the **State** field. The query checks for OpcQuality equal to opcnull and StringValue "NULL" in the result.

Use binding in custom properties

Application Server object scripting supports a type called "Indirect". It enables you to bind a variable to a reference and read and write to it. This is done using the BindTo() method.

Note: The BindTo() method binds a variable to a reference as long as the graphic is shown.

For example, the local script variable ptr is defined and bound to the reference ud1_001.Int1.

```
dim ptr as indirect;
ptr.BindTo("ud1_001.Int1");
```

Within the same script you can use the indirect variable pointer to read from and write to the attribute ud1_001.Int1.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Industrial Graphics also use scripting in the same way as the scripting of Application Server.

However, as an Industrial graphic can be embedded into an InTouch window and run anonymously, the time it takes to connect to the reference can be longer than one scan cycle.

For that reason, you cannot use the indirect variable immediately after it is bound to a reference to read from and write to it.

```
dim ptr as indirect;
ptr.BindTo("ud1_001.Int1");
ptr = 42;
```

In the example, the value 42 cannot be written to the reference ud1_001.Int1 as the binding takes longer.

To avoid this problem, you can modify your Industrial graphic script, using the Industrial Graphic Script Editor to write the value after it is ensured that the binding is complete. The completion of the binding is indicated by the quality of the indirect variable.

- Using artificial delays in the script is not recommended as this may result in script timeouts and would in turn affect UI refresh. For example, do not use a while loop that increments a variable from 1 to <60000. This could cause the runtime application (OMI or InTouch) to go blank blank or become unresponsive while the script is executing.
- Instead of artificial delays, allow the whole script to continue execution across multiple cycles, without any artificial delays in the script, and set the script's expression to false once the quality of all the attributes is good.

Create a named script, with a trigger set to ‘While True’ and query for the quality and use the indirect variable to read from and write to the reference when its quality is good. This script will try again every second until the indirect attribute returns with good quality. If the quality is good, then the script exits from the while loop. Once the quality is confirmed, you can use the indirect variable reference in other scripts or button animations.

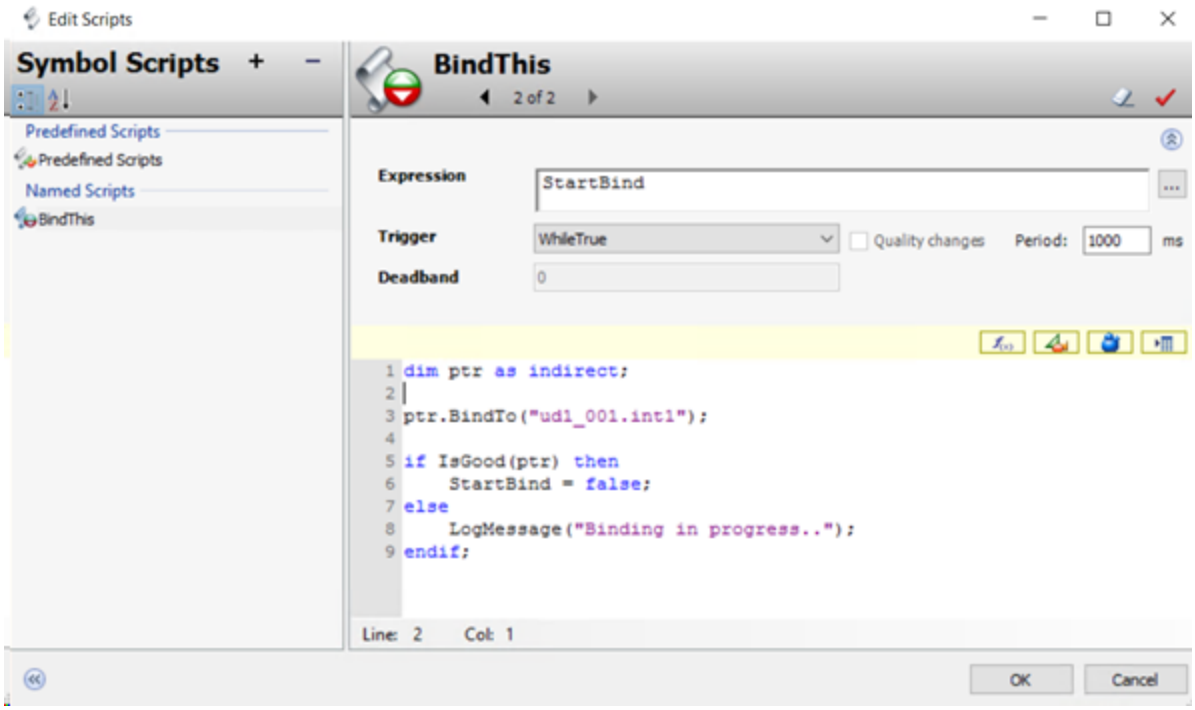
Note: Make sure to include an exit condition in your script, so that the script does not hang if the binding cannot be made.

The following example script shows you how to do this:

- Expression: Create a Boolean custom property StartBind with an initial value of True.
- Trigger: WhileTrue
- Period: 1000 ms

```
dim ptr as indirect;
ptr.BindTo("ud1_001.Int1");
if IsGood(ptr)then {if quality not good}
StartBind = false;
else
LogMessage("Bidnding in progress..");
endif;
```

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.



Note: Similar behavior can occur when you try to bind to a reference of an object that is hosted on a different AppEngine.

Change the expression or reference of a custom property at runtime

You can change the expression or reference of a custom property at runtime by calling the SetCustomPropertyValue() method on the graphic using a client script:

```
SetCustomPropertyValue(System.String name, System.String value, System.Boolean isConstant);
```

You can select this method using the Element Browser from within the Industrial Graphic Editor.

This method has three parameters:

- *name* - Name of the custom property to be modified on the graphic. This parameter is of type string, and it can be a reference or a constant.
- *value* - The new value to be set. This parameter is of type string, and it can be an expression, reference, or constant. If the value is given in quotes ("), then the value is considered a constant. If the value is given without quotes, then the value of the expression is considered a reference.
- *isConstant* - A flag that indicates whether the new value will be evaluated as a constant or a reference. This parameter is of type Boolean. If it is set to True (1), then the new value will be treated as a constant. If it is set to False (0), then the new value will be treated as a reference. This parameter only applies when the *value* parameter is a reference or constant and the custom property specified in the *name* parameter is a string or time type. This parameter has no meaning if the custom property is an integer, float, Boolean, or double type.

Note: The *isConstant* parameter does not override the type of input for the *value* parameter. The *value* parameter itself can be either a constant or a reference depending on whether it is enclosed in quotes. The *isConstant* parameter is only determining how the actual value (coming from the *value* parameter) is evaluated.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

The whole expression or reference of the custom property is replaced with the new value, regardless if it is overridden or not. No partial replacement is supported.

Only public custom properties on the graphic can be changed.

When the method executes, it overrides any modifications done by previous IOSetRemoteReference() calls from a native InTouch script.

For an example of configuring the custom property as a reference, say you have a Motor_001 object with the string attribute name "State" that stores the current state of the motor ("Running" or "Stopped"). You also have an Industrial graphic that has the string data type custom property "MotorState." The following script code will set the MotorState custom property to Motor_001.State in runtime:

```
GraphicA.SetCustomPropertyValue("MotorState","Motor_001.State",False);
```

As a result of the call, the function will set the string custom property GraphicA.MotorState to "Motor_001.State" as a reference. The string custom property GraphicA.MotorStatus will resolve that reference and update its value with the reference value ("Running" or "Stopped").

For an example of configuring the custom property as a constant, say you have a Motor_001 object with the Boolean attribute name "State" that reflects the current state of the motor (True or False). You also have an Industrial graphic that has the string data type custom property "MotorState." The following script code causes the MotorState custom property to hold the state of equipment—"Running" or "Stopped"—as text based on the value returned for Motor_001:

```
IF Motor_001.State THEN
GraphicA.SetCustomPropertyValue("MotorState","Running",True);
ELSE
GraphicA.SetCustomPropertyValue("MotorState","Stopped",True);
ENDIF;
```

As a result of the call, the function will set the string custom property GraphicA.MotorState to "Running" or "Stopped," depending on the value of Motor_001.State.

Working with the SignedWrite() function for secured and verified writes

This section provides information about the SignedWrite() function, SignedWrite() runtime behavior, scripting tips, and in-depth script examples.

For SignedWrite() scripting information including script syntax, parameters, and basic script examples, see the *Application Server Scripting Guide.*

You can write to an Automation Object attribute that is configured for Secured Write or Verified Write security classification by means of the Industrial Graphics SignedWrite() script function.

The SignedWrite() function can be used only in client scripts, not in Application Object scripts, and only on Attributes that have been configured for Secured Write or Verified Write. Attempting to use the function on an Attribute not so configured will result in an error message at runtime.

SignedWrite() runtime behavior

At runtime, the SignedWrite() function does the following:

1. Checks the target Attribute for Signed Write or Verified Write configuration.

   If not so configured, the following error message appears: **Operation Failed. Attribute does not have the correct security classification.**

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

2. Checks Galaxy security.

   If the Galaxy is not secured, the following error message appears: **Operation Failed. Galaxy is not secured.**
3. There are several ways to write to an Attribute configured with Secured Write or Verified Write security classification, it is possible to have multiple SignedWrite() and other Secured/Verified writes pending from the same script, or even from multiple scripts running side-by-side.
4. Determines which dialog is required—Secured Write or Verified Write—and pops up the appropriate dialog box.

   If Smart Cards are enabled, the function displays different versions of the Secured Write and Verified write dialog boxes.
5. Lists the predefined comments, if any, from the configured Predefined Comments list. Up to 20 comments are supported.
6. Enables comment editing if the Comment_Is_Editable parameter is configured and comment enforcement is other than PredefinedOnly.
7. Acquires the user credentials and authenticates them.

   If the user credentials are invalid, an error message appears. The function will attempt the write only if the credentials are valid.
8. Checks if comment enforcement is mandatory, and displays an error message if the comment is empty.
9. Performs the write if user credentials are valid and the comment entry satisfies the comment enforcement parameter.
10. Provides a return status.
11. Following a Secured or Verified Write a security Event is written to the event log, including the signee name, verifier name, if any, Type of write: "Secured Write" or "Verified Write", Comment, if any entered by user, Reason Description, if any provided, Field Attribute description, if any, or the Short Description of the Application Object, if no Field Attribute description exists.

Each call to SignedWrite() is distinct from any other. The success or failure of any individual write does not affect other attempted writes.

Entering user credentials for SignedWrite() is distinct from logging on to the client application. The user can modify attributes configured with Secured or Verified Write even if another user is logged on, without affecting the session of the logged-on user.

SignedWrite() script execution sequence at runtime

The SignedWrite() function goes into a queue and the script continues executing. The function is queued for operator entry. The script may complete prior to the operator completing the Secured or Verified Write operation.

By contrast, the SignedAlarmAck() script function executes completely synchronously, and waits for user input before proceeding to the next line in the script.

SignedWrite() scripting tips

Bound references in SignedWrite()

If the Attribute parameter string evaluates to the name of a Custom Property and that Custom Property is a bound reference to an Attribute, the SignedWrite() function will write to that indicated Attribute. The Attribute

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

must have the security classification of Secured Write or Verified Write.

The SignedWrite() function supports Custom Properties that are nested bound references. That is, if the string evaluates to the name of a Custom Property and that Custom Property is a bound reference to another Custom Property which itself is a bound reference, the SignedWrite() function will follow through the chain of bound references until it finds an item that is a value. If that item is an Attribute that has the security classification of Secured Write or Verified Write, the SignedWrite() function will write to that item.

SignedWrite() in WhileTrue, WhileFalse, or periodic type scripts

Using the SignedWrite() function with WhileTrue, WhileFalse, or Periodic type scripts can repeatedly execute the script, causing another secured write dialog box to pop up with each trigger. We do not recommend using the SignedWrite() function with WhileTrue, WhileFalse, or Periodic types.

SignedWrite() with OnShow and OnHide scripts

We do not recommend using the SignedWrite() function with OnShow and OnHide scripts. This can cause issues with window functionality, including the window title bar, windows losing correct focus, and windows opening on top of one another.

Examples of using the attribute parameter in the SignedWrite() function

Working from the overall syntax of the SignedWrite() function, the script examples in the following table illustrate a number of approaches to using the Attribute parameter in the SignedWrite() function.

The following String, Boolean, and Integer user defined attribute conditions apply to the script examples:

- User Defined object UD1_001
- String attribute UD1_001.udString1 with the value "WW". Security classification is set to Secured Write.
- Boolean attribute UD1_001.secBool1 with the value false. Security classification is set to Secured Write.
- Integer attribute UD1_001.secInt1 with the value 24. Security classification is set to Secured Write.
- User Defined object UD2_002

String attribute UD2_002.udString2 with a value "UD1_001.udString1" The following custom property conditions apply to the script examples:

- String custom property CP1 with a reference to UD1_001.udString1
- String custom property CP2 with a value "UD1_001.udString1"
- String custom property CP3 with a value "UD1_001"
- Boolean custom property CP4 with a reference to UD1_001.secBool1
- Integer custom property CP5 with a reference to UD1_001.secInt1
- String custom property CP6 with a reference to an attribute on Owning Object UD1_001 as me.udString1

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Script example	Function and result
SignedWrite("CP1", "AVEVA", "using redirect", true, 0, null);	Uses the CP1 reference UD1_001.udString1 and pokes to it the value "AVEVA". Result: The value in UD1_001.udString1 will change from "WW" to "AVEVA".
SignedWrite(CP2, "AVEVA", "using string value", true, 0, null);	Resolves CP2 string value "UD1_001.udString1" to a reference and pokes to it the value "AVEVA". Result: The value in UD1_001.udString1 will change from "WW" to "AVEVA".
SignedWrite(CP3+".udString1", "AVEVA", "using string expression", true, 0, null);	Resolves the string "UD1_001.udString1" to a reference and pokes to it the value "AVEVA". Result: The value in UD1_001.udString1 will change from "WW" to "AVEVA".
SignedWrite("UD1_001.udString1", "AVEVA", "using constant string", true, 0, null);	Resolves the string "UD1_001.udString1" to a reference and pokes to it the value "AVEVA". Result: The value in UD1_001.udString1 will change from "WW" to "AVEVA".
SignedWrite(UD2_002.udString2, "AVEVA", "using attribute containing string", true, 0, null);	Resolves the UD2_002.udString2 string value "UD1_001.udString1" to a reference and pokes to it the value "AVEVA". Result: The value in UD1_001.udString1 will change from "WW" to "AVEVA".
SignedWrite("CP" + "1", "AVEVA", "using redirect from string expression", true, 0, null);	Resolves the expression to "CP1" to use the CP1 reference UD1_001.udString1 and pokes to it the value "AVEVA". Result: The value in UD1_001.udString1 will change from "WW" to "AVEVA".
SignedWrite("CP4", true, "using redirect", true, 0, null);	Uses the CP4 reference UD1_001.secBool1 and pokes to it the value true. Result: The value in UD1_001.secBool1 will change from false to true
SignedWrite("CP5", 37, "using redirect", true, 0, null);	Uses the CP5 reference UD1_001.secInt1 and pokes to it the value 37. Result: The value in UD1_001.secInt1 will change from 24 to 37
SignedWrite("CP6", "AVEVA", "using redirect using relative reference", true, 0, null);	Uses the CP6 reference me.udString1 and resolves it to UD1_001.udString1 and pokes to it the value "AVEVA".

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Script example	Function and result
	Result: That the value in UD1_001.udString1 will change from "WW" to "AVEVA"

Configure the SignedWrite() script function

You can create a dashboard application to automate routine use of Secured and Verified Write by means of the SignedWrite() function.

To configure the SignedWrite() script function

1. Open the System Platform IDE.
2. Create a graphic and associate it with an attribute configured with Secured Write or Verified Write. For more information on associating attributes with graphic, see the Application Server User Guide, "Creating and Working with UDAs" topic.
3. Add the SignedWrite script function to the graphic. The following editor detail shows the buttons configured with scripts in the applied example:
4. Configure the scripted functionality you require. Scripts for the buttons shown in the example are as follows:
   a. Hard-coded DataUDO.SecUDA: The following example sets the value of 23 to DataUDO.SecUDA. The user optionally can enter a comment, but no pre-defined comment list is available.

      ```
      DataUDO.RetStatus=SignedWrite("DataUDO.SecUDA", 23, "Set the Value", True, 0, null);
      ```

   b. Attribute Pointer has DataUDO.SecUDA: The source to be written to is passed as a parameter to the function. Attribute_Pointer is a custom property whose value is set to DataUDO.SecUDA.

      The following example sets the value of 23 to DataUDO.SecUDA. The user optionally can enter a comment, but no pre-defined comment list is available.

      ```
      DataUDO.RetStatus=SignedWrite(Attribute_Pointer, 23, "Set the Value", True, 0, null);
      ```

   c. Attribute Pointer and Pre-Defined List: The pre-defined comment list is an array. This example extends the functionality of example b to force the user to enter a comment (Comment_Enforcement parameter set to 1) and also presents a pre-defined set of comments linked to the DataUDO.PreDefComments[ ] array.

      The following example will set the value of 23 to DataUDO.SecUDA. The user must enter a comment and may use one from the pre-defined comment list.

      ```
      DataUDO.RetStatus=SignedWrite(Attribute_Pointer, 23, "Set the Value", True, 1,
      DataUDO.PreDefComments[]);
      ```

   d. Variable Array: The predefined list is a pointer to an array. This example extends the functionality of example c to force the user to enter a comment (Comment_Enforcement parameter set to 1) and also presents a predefined set of comments linked to DataUDO.PreDefComments[ ] array.

      The value of custom property CP1 is "DataUDO.PreDefComments[ ]".

      The following example will set the value of 23 to DataUDO.SecUDA. The user must enter a comment and may use one from the pre-defined comment list.

      ```
      dim xInd as Indirect;
      xInd.BindTo(CP1);
      DataUDO.RetStatus=SignedWrite(Attribute_Pointer, 23, "Set the Value", True, 1,
      xInd);
      ```

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

e. All Parameters Variable: The predefined list array is built into the script. All parameters are passed as variables.

The following example will set the value of 23 to DataUDO.SecUDA . The user must enter a comment and may use one from the pre-defined comments list.

```
dim MyList[5] as string;
MyList[1] = "Batch Accepted";
MyList[2] = "Batch Rejected";
MyList[3] = "Batch on Hold";
MyList[4] = "Batch Resumed";
MyList[5] = DataUDO.PreDefComments[4];
DataUDO.RetStatus=SignedWrite(Attribute_Pointer, SignedWrite_Value_Ptr,
SignedWrite_Reason, Enable_Edit_Comment, Comment_Options, MyList[]);
```

Show/Hide content script functions

Note: The Show/Hide Content() script functions are for use with AVEVA OMI ViewApps only. These functions cannot be used with InTouch HMI applications.

The Show/Hide Content() script functions let you write scripts for AVEVA OMI layouts, Industrial Graphics, Application Server objects to display a graphic, layout, or external content item (for example, a web page, video, or text file) inside a specific pane within an AVEVA OMI ViewApp. The Show/Hide Content() script functions are complementary to the Show/Hide Graphic script functions and the Show/Hide Graphic animation feature.

- Use the Show/Hide Graphic script functions to load or hide a graphic in a modal or modeless popup, in an InTouch HMI application or an AVEVA OMI ViewApp.
- Use the Show/Hide Content script functions to load or hide a graphic, layout, or an external content item into a pane in an AVEVA OMI ViewApp. When a Layout linked to an asset is added to a pane in another layout, any empty panes in the linked Layout that have the "Use for Autofill" flag enabled will attempt to fill the empty panes with content from the linked asset, using the “Current Only” autofill setting.

The Show/Hide Content functions let you specify:

- The specific content to show or hide in a pane.
- The pane in which to show or hide the content.
- The screen that contains a pane in which to show or hide the content.
- The type of content you want to show or hide.
- Any property overrides to apply when showing the content.
- Whether to confine the search for a pane in which to show or hide the content to the source screen, the primary screen, or to search all screens, and whether to search in the top level layout or a nested (embedded) layout.

Configure the Show/Hide Content script functions

Use the Show/Hide Content functions in layout scripts, Industrial Graphic scripts, or Application Server object scripts to populate a pane with a specific graphic, layout, or external content object in an AVEVA OMI ViewApp. You can call the same script multiple times to refresh content, or to show the content in a different pane. If you choose to display the same content in different panes, you can alter its settings through the ParameterOverride

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

parameter.

If ShowContent calls the same content item while it is open, ShowContent searches for open content items that match the specified parameters in the call. Any matching items are closed and then reopened. If the content has changed, the latest changes are shown when the content is redisplayed.

Important: The Show/HideContent functions can be used in a graphic or layout action script, named script and pre-defined script. Although the system allows you to include it in a server script, such as Start Up, On Scan, Off Scan, Shut Down and Execute, you will not be able to execute the function at runtime.

To include a script that contains the Show/Hide Content functions within a script

1. Open the System Platform IDE.
2. Open a new or existing graphic or layout.
   - To add a layout script, click the **Script** tab.
   - To add a graphic script, create a graphic and open it fro editing. Double-click it to open the **Edit Animations** page, then open the script editor and click the **Display Script Function Browser** icon. The **Script Function Browser** appears. Click the ShowContent() script function in the **Graphic Client** list, and then click **OK.** The following code snippet is added:

     ```
     Dim contentInfo as aaContent.ContentInfo;
     contentInfo.Content = "<ContentName>";
     ShowContent( contentInfo );
     ```

   Note: You can click **Help** to view the Help after you have selected any Graphic Client script function.
3. Modify the script as needed. ContentInfo is a predefined structure that contains data members listed below. Content is the only required parameter. For more information about the Show/Hide Content, see the AVEVA Scripting Guide.
   - **Content:** name of the content (graphic, layout, or external content object) to be loaded into the pane. Content is a string and is the only required parameter.
   - **Name:** name property of the content. This is automatically created when the content is added to the layout.
   - **ScreenName:** name of the screen that contains the pane in which to load the content. ScreenName is a string.
   - **SearchScope:** when ScreenName is not specified, **SearchScope** determines which screens within the Screen Profile are searched for a pane in which to place the content. SearchScope is an enum that can be set to one of the following values:
     - **Self:** This is the default. Searches for matching content within the panes of the layout from which the ShowContent call was made. Self will search from a nested (embedded) layout, if the call was made from that layout. The remaining SearchScopes search ONLY the top level if there are nested layouts.
     - **AllScreens:** Searches the top level layout on all screens, starting with the source screen, then the primary screen, and then any remaining screens in alphabetical order. The search stops as soon as a matching pane is found. If a matching pane is not found, Content is placed in the default pane of the source screen.
     - **SourceScreen:** Searches the top level layout only in the source screen (the screen that made the ShowContent call).
     - **PrimaryScreen:** Searches the top level layout only in the primary screen, as designated in the Screen Profile.
   - **PaneName:** name of the pane in which to load the content. PaneName is a string.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

- **ContentType:** specifies the type of content to be loaded. The ContentType can be matched against the Content Type designation that was set for a pane. ContentType is a string.
- **PropertyOverrides:** specifies overrides for custom properties. This parameter is only valid if a graphic is the designated content. It is not valid if a layout has been designated. PropertyOverrides is a ValuePair structure.
- **OwningObject:** specifies an automation object as an owning object.

Example

```
Dim contentInfo as aaContent.ContentInfo;
Dim cpValues [2] as aaContent.PropertyOverrideValue;
cpValues[1] = new aaContent.PropertyOverrideValue("CP1", "20", true);
cpValues[2] = new aaContent.PropertyOverrideValue("CP2", "Pump.PV.TagName", false);
contentInfo.Content = "S1";
contentInfo.ContentType = "Overview";
contentInfo.OwningObject = "Enterprise";
contentInfo.PaneName = "Pane1";
contentInfo.ScreenName = "Wall";
contentInfo.PropertyOverrideValues = cpValues;
contentInfo.SearchScope = aaContent.SearchScope.PrimaryScreen;
ShowContent ( contentInfo );
```

About the best pane match algorithm

When some or all of the optional parameters are not provided or are invalid, ShowContent follows a set of rules to determine in which pane the specified content should placed. In addition, the SearchScope parameter can be used to limit which screens will be searched for a pane.

- If SearchScope is set to SourceScreen, only the screen that made the ShowContent call is searched. If no matching pane is found, Content is loaded into the default pane of the SourceScreen.
- If SearchScope is set to PrimaryScreen, only the primary screen in the Screen Profile is searched. If no matching pane is found, Content is loaded into the default pane of the PrimaryScreen.
- If SearchScope is set to AllScreens, every screen contained in the Screen Profile can be searched. This is the default behavior. Searches progress as follows and stop at the first match:
  - SourceScreen
  - PrimaryScreen
  - Any additional screens are searched in alphabetical order.
  - If no matching pane is found, Content is loaded into the default pane of the SourceScreen.
- If SearchScope and ScreenName are not specified and ShowContent is called from an embedded layout:
  - ShowContent searches locally for a matching pane within the embedded layout.
- If SearchScope or ScreenName is specified and ShowContent is called from an embedded layout:
  - ShowContent searches for a pane globally and uses the SearchScope and/or ScreenName settings to determine which screens within the Screen Profile associated with the ViewApp will be searched.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Order of precedence for determining content placement

Content is the only required parameter. Three values are also checked according to the following order of precedence to determine placement of the content:

1. PaneName: optional parameter
2. ContentType: optional parameter
3. Type of content: this is part of the Content definition and is not specified in the ShowContent call.

Search steps (SearchScope not specified or set to AllScreens)

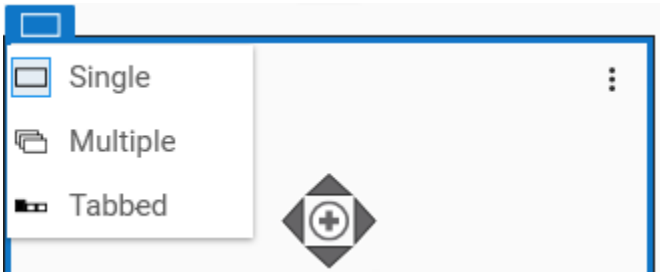
Scenario 1: PaneName PaneName is specified	Assumes SearchScope not set or set to AllScreens. Otherwise, search is confined to the named screen. 1. Search SourceScreen for PaneName. 2. If not found, search PrimaryScreen for PaneName. 3. If not found, search any additional screens in alphabetical order for PaneName. The search stops as soon as a matching PaneName is found.
Scenario 2: ContentType PaneName is not specified OR matching PaneName not found AND ContentType is specified	Assumes SearchScope not set or set to AllScreens. Otherwise, search is confined to the named screen. 1. Search SourceScreen for a pane that supports the ContentType. 2. If not found, search PrimaryScreen for pane that supports the ContentType. 3. If not found, search any additional screens in alphabetical order for a pane that supports the ContentType. The search stops as soon as a pane that supports the ContentType is found.
Scenario 3: Type of content PaneName is not specified OR matching PaneName not found AND ContentType not specified or not found	Assumes SearchScope not set or set to AllScreens. Otherwise, search is confined to the named screen.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

	1. Evaluate type of content for the specified Content. 2. Search SourceScreen for a pane that supports the type of content. 3. If not found, search PrimaryScreen for pane that supports the type of content. 4. If not found, search any additional screens in alphabetical order for a pane that supports the type of content. The search stops as soon as a pane that supports the type of content is found.
Scenario 4: No matching pane No matches found from previous searches (scenarios 1 through 3)	When SearchScope is not set or set to AllScreens: If no match is found using PaneName, ContentType, or type of content, Content is loaded into the default pane of the SourceScreen. When SearchScope is set to SourceScreen: Only the SourceScreen is searched and if no match is found, Content is loaded into the default pane of the SourceScreen. When SearchScope is set to PrimaryScreen: Only the PrimaryScreen is searched, and if no match is found, Content is loaded into the default pane of the PrimaryScreen.

Content display rules

The way content is displayed differs slightly between panes that are configured as single content panes versus those configured as multi-content panes. The differences are described below. Pane configuration (single or multi-content) is set by opening the **Presentation Style** selector at the top left of the pane in the **Layout Editor**. See Set the Presentation Style of a Pane in the *System Platform Help* for additional information about single and multiple content modes.



If there are multiple ShowContent() calls, but with different parameters, each ShowContent() call opens a different pane or tab. The content in each pane or tab is configured with the parameters that are specified in the

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

call.

Content display rules – single content pane

If the pane already contains content, the existing content is closed and the content specified by ShowContent() then is loaded into the pane. This occurs even if the existing content is the same as the specified content.

Content display rules – multi-content or tabbed pane

For panes that are configured as multiple-content or tabbed panes,the exact behavior will depend on the structure of the pane.

- If the specified Content is already open in the pane or a tab, the pane or tab remains open and the existing Content is replaced with a new instance of the same Content.
- If the pane or tab has been split (that is, contains child panes), or contains a nested layout, existing panes and/or tabs remain open, and the specified content is loaded into a new tab or pane and the focus is set to the new tab or pane.
  - This is true as long as the number of tabs or child panes is less than the maximum available. The default setting allows up to 20 panes or tabs, but you can increase this to 50.
  - If the maximum number of tabs or child panes has been reached, the pane that has been updated last (oldest content) is closed and a new tab or pane is opened in its place.

Languages and localization

This section describes the features and procedures for configuring languages and enabling run-time language switching.

Define and configure galaxy languages

System Platform language features enable you to develop applications that can be switched to another language at run time. To enable runtime language switching, you must:

- Configure multiple languages for the application.
- Export your application text for offline translation.
- Translate one or more exported dictionary files.
- Import one or more translated dictionary files into the Galaxy.

You can set the Galaxy default language and you can define additional languages for purposes of switching graphic and alarm languages at runtime.

The default language is set and languages are added only at the Galaxy level.

Graphics language switching

Runtime language switching applies to all portions of the graphics or animations that you create or configure using the Industrial Graphic Editor or InTouch WindowMaker. The language to display is set when the application is being shown at runtime.

You can only view translations at design time for Industrial Graphics in the Graphic Editor. You cannot switch

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

languages in WindowMaker at design time to see the translations.

Alarm language switching

The alarm language switching feature allows you to format and export various alarm messages and fields from attributes, including:

- Alarm messages
- Group names (area names)
- State messages (on/off labels)
- Event messages, such as:
  - Attribute description field, for attributes that have the Log Event feature enabled
  - Group (area) names
  - State labels
  - Standard event messages, for example, "Write success," Write access denied," etc.

When you export alarm messages to an external file, you can produce localized files to support language switching at run time. Alarm language switching is supported in both InTouch HMI and AVEVA OMI run-time applications. After translating the exported alarm file, simply import the translated file back into the galaxy.

If you change an alarm comment after exporting the alarm comments, you must re-export the alarm comments. The translated alarm comments imported into Application Server and displayed at run time do not change dynamically when the alarm comment is edited.

InternationalizedString attributes

Attributes of type InternationizedString can be localized:

- Aliasname

  Aliasname is also used by the AVEVA OMI NavTree and Breadcrumb navigation apps, and these apps recognize localized Aliasname attributes.
- Attribute.Description
- Boolean OnMsg
- Boolean OffMsg
- EngUnits
- ShortDesc
- Any User Defined Attribute that is of type InternationalizedString

To export InternationalizedString attributes for translation, and to import the translated strings, see the following sections:

- Export files for offline translation
- Import translated language files

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Configure languages for a Galaxy

The language settings of the Galaxy control which languages are available to graphics and alarm and event messages. You cannot add a language at the graphic or attribute level. Languages are only added at the Galaxy level using the IDE.

When you open a graphic for editing using the Graphic Editor, open a managed InTouch application in WindowMaker, or open an AVEVA OMI ViewApp, the language settings are retrieved from the Galaxy.

For example, you configure English and French languages for the Galaxy. You open graphic G1 in the Graphic Editor. graphic G1 is now configured with English and French languages. Using the IDE, you add German to the list of configured languages. You must close the Graphic Editor and open graphic G1 again to see the German language available for the graphic.

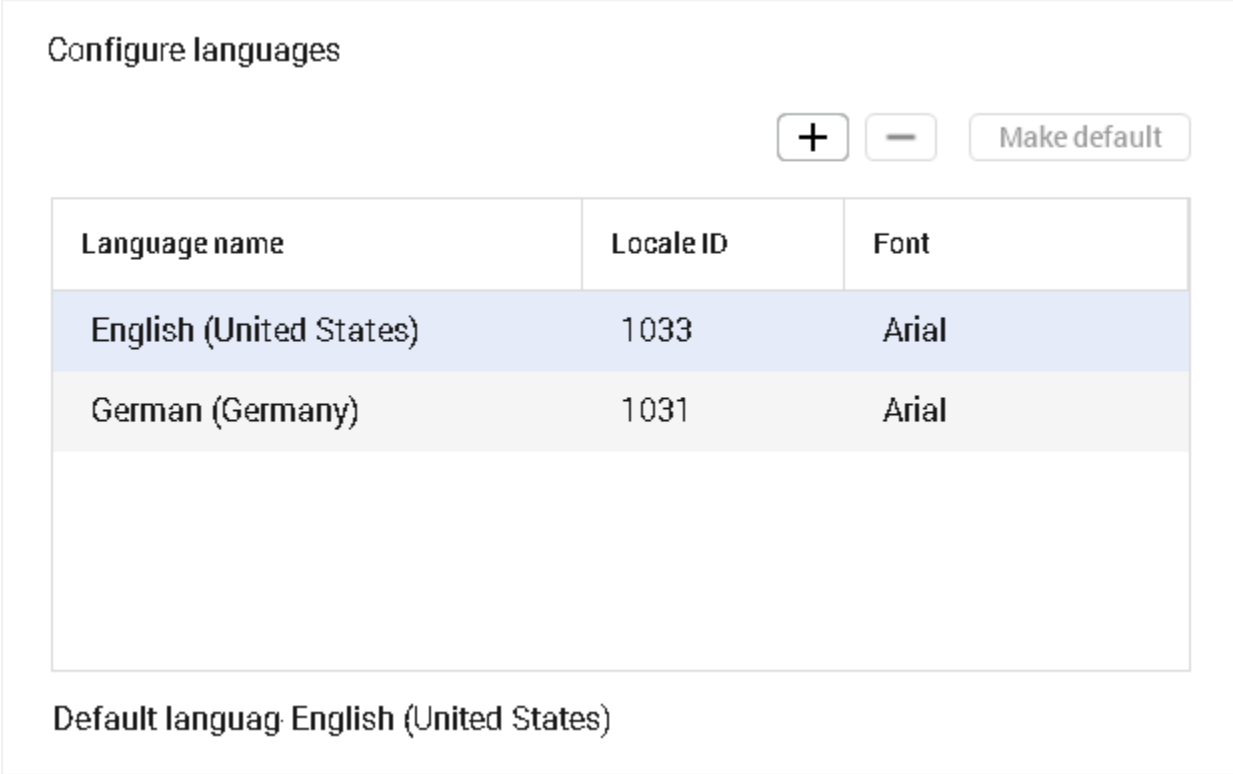
For more information about language switching in Industrial Graphics, see the *Creating and Managing Industrial Graphics User Guide*.

Add a language to a Galaxy

Every Galaxy is associated with a base language. You must configure any additional languages that you want to be supported by the Galaxy.

To add a language to a Galaxy

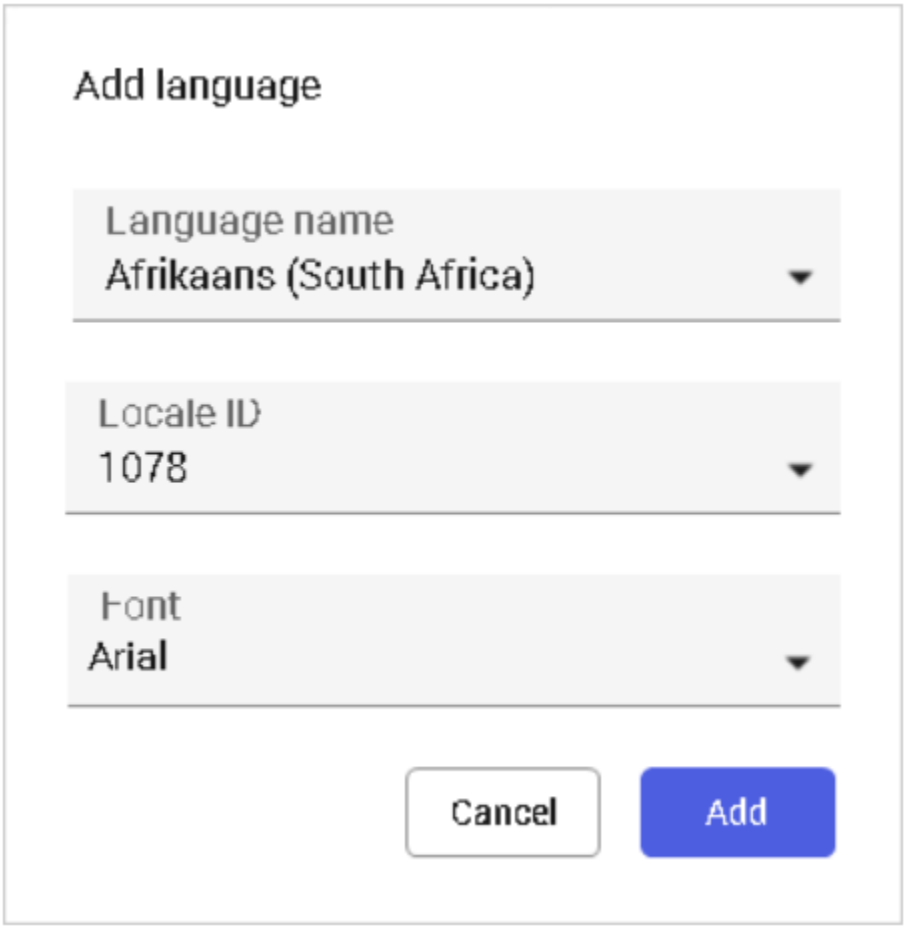
1. Using the System Platform IDE, open the Galaxy that you want to add a language to.
2. From the ribbon, select **Galaxy**, then select **Configure**.
3. Select **Galaxy**, then **Languages**. The **Configure Languages** dialog box appears.



© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

The **Configure Languages** dialog box shows the base (default) language of the Galaxy.

4. Click the **Add [+]** button. The **Add Language** dialog box appears.



5. Specify the language and font for the translated text.
   - In the **Language name** or the **Locale ID** list, select the language to add. If you select the language by name, the corresponding locale ID appears in the **Locale ID** list, and vice versa.
   - In the **Font** list, select a font.
6. Click **Add** to close the **Add Language** dialog box. The language you selected now appears in the **Configure Languages** dialog box.
7. To add more languages, repeat steps 4 through 6.
8. To specify a particular language as the default, select the row in the list and then click **Make Default**.
9. When you are done, click **Save**.

Remove a language from a Galaxy

You cannot remove the default language. At least one language must be configured for a Galaxy.

To remove a language for a Galaxy

1. Using the System Platform IDE, open the Galaxy from which you want to remove a language.
2. From the ribbon, select **Galaxy**, then select **Configure**.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

3. Select **Galaxy**, then **Languages**. The **Configure Languages** dialog box appears.
4. Select the language to remove and then click the **Remove[-]** button.
5. When the **Confirm Delete** dialog box appears, click **Delete**.
   - The dialog box provides an additional option to delete the alarm comment translations for the language selected. Leave the check box empty (unselected) if you want to keep the alarm comment translations for the selected language, but just want to remove the language selection option from the galaxy.
6. Click **Delete** to confirm deletion of the selected language (and the alarm comment translations for that language if selected)**.**

Modify the font for a language

The default font for all languages is Arial. You can only set the font for a language at the time you add it. To change the font setting for a previously configured language, you must first delete it and then re-add it as described in Add a language to a Galaxy.

You cannot edit the font for the default language. To change the font for the default language:

1. Set a different language as the default.
2. Delete the language for which you want to change the font.
3. Add back the language and set the font.
4. Set the language you just added as the default.

The configured font for a language is used in design time when a specific translation is supplied for a language. It is also propagated to all translations for secondary languages when the fonts are not specifically overridden by you.

For example, if you create a text element and provide translation for the secondary language without modifying the font, the text is shown using the font from the Galaxy. However, if you manually modify the font, then the text will always use the chosen font. This same behavior also applies to runtime.

For an Industrial graphic to show the updated font from the Galaxy in an InTouch ViewApp, WindowViewer must be restarted. No notification is provided to WindowViewer when the Galaxy font changes.

All managed InTouch applications are updated to use the new font for the selected language.

Change the default language for a Galaxy

When you change the default language for the Galaxy:

- The new default language is shown when the Graphic Editor opens.
- The base language for translation export is changed to the new default language.
- You cannot import translations for that language.
- Graphics opened in design time use the default language when showing text that does not have a specific translation in a secondary language.
- Graphics shown at runtime use the default language if specific translated values are not provided for the currently viewed language.
- Thumbnails for graphics are shown using the default language.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

To change the default language of a Galaxy

1. From the ribbon, select **Galaxy**, then select **Configure**.
2. Select **Galaxy**, then **Languages**. The **Configure Languages** dialog box appears.
3. Select the language you want to set as the new default language.
4. Click **Make Default**.

   The selected language is now the default language for the Galaxy.

Types of language dictionary files

Export operations to support localization can create the following types of dictionary files:

- Dictionary file (type XML) for all Visualization folder graphics, template graphics, and AutomationObject graphics. The file naming convention for graphics is: *<GalaxyName>_<LanguageID>*.xml. For example, if the Galaxy name is TestSample and the language being exported is French (Language ID = 1036) then the file name is TestSample_1036.xml.
- Dictionary file (type TXT) for internationalized string attributes. The file naming convention for internationalized strings is: Galaxy_*<GalaxyName>_<LanguageID>*_Internationalized_strings.txt
- Dictionary file (type TXT) for alarm messages, group names, state messages, and event messages. The file naming convention for alarm messages is: Galaxy_*<GalaxyName>_<LanguageID>.*Alarm_Comments.txt
- Dictionary file (type XML) for each managed InTouch application.
- Dictionary file for each SmartSymbol (type XML) in managed InTouch applications.

Note: Alarm export files are text files and are therefore named and formatted differently from graphic dictionary files, which are XML files. For further information see Export alarm comments for offline translation.

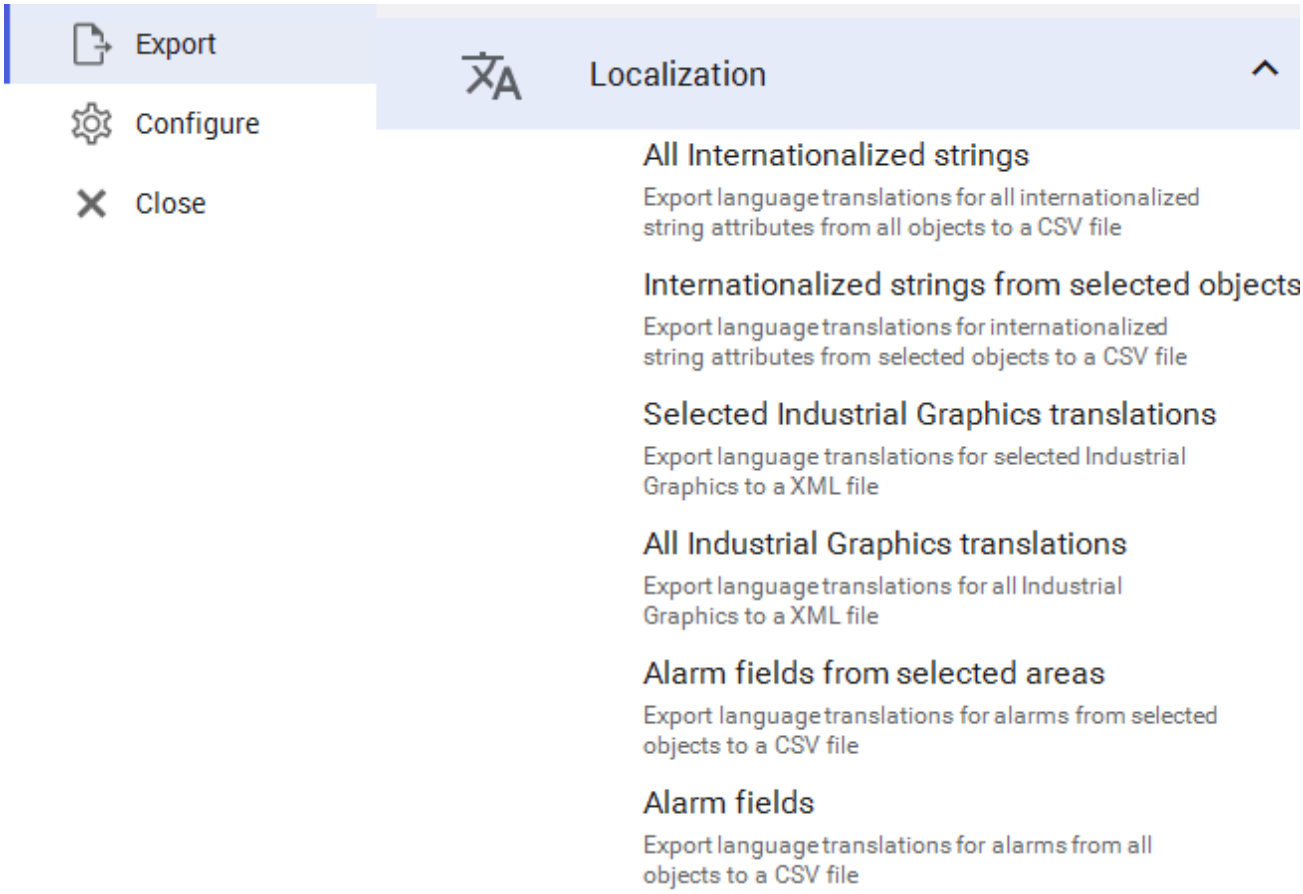
Export files for offline translation

Typically, text strings are sent out for bulk translation. You can export translation files and modify them using a text editor or a spreadsheet program like Microsoft Excel.

Application Server has the following export options for localization:

- All Internationalized strings
- Internationalized strings from selected objects (one or more objects must be selected before this option can be used)
- Selected Industrial Graphics (select the graphics you want to localize before you use this option)
- All Industrial Graphics
- Alarm Fields from selected areas (select the area objects you want to localize before you use this option)
- All Alarm Fields

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.



If you make changes to your application after you export your dictionary files, you must export the dictionary file again. For more information, see Export text to an existing dictionary file.

You can only export the text strings for one language at a time. You cannot export Industrial Graphics text strings for the default language.

In order to localize a string attribute, it must be of type InternationalizedString. Not all attributes are this type. For a list of InternationalizedString attributes, see InternationalizedString attributes .

Each graphic in a Galaxy is only exported one time, no matter how many times it is referenced.

When you export the text, you specify a folder for the dictionary files. Creating a new folder to export phrases for each language makes it easy to manage dictionary files. For example, ...\Galaxy1\My German Files\.

Also, alarm message files are exported as tab-delimited text files and have the following convention: Galaxy_*<GalaxyName>*_*<LanguageID>*.txt. If you will be exporting language data for different objects at different times, use separate target directories to prevent subsequent exports from overwriting the first export.

When you export the dictionary for an application, you can copy and paste the text file into Microsoft Excel.

About localization workflow

The typical workflow for adding translations to a Galaxy consists of the following:

1. From the **Galaxy** menu, select **Export**, select **Localization**., then choose which option you want to use. If you did not select any objects in the IDE before opening the **Export** menu, the options for choosing **selected**

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

graphics, alarm fields, and internationalized strings will not be active:

- All internationalized strings
- Internationalized strings from selected objects (inactive if no objects were selected)
- Selected Industrial Graphics (inactive if no objects were selected)
- All Industrial Graphics
- Selected Alarm Fields (inactive if no objects were selected)
- All Alarm Fields

2. When exporting Industrial Graphics or internationalized strings, select the language you are exporting for translation and a destination folder.

   When exporting alarm fields, select as appropriate:

   - Export all alarm messages to a single file
   - Create one translation file per area
   - You can also limit the export to only alarm messages that have not yet been translated.

     Note: For applications that only target sub-areas of the Galaxy, exporting by Area is a more optimal workflow.

   When exporting internationalized strings, you can also limit the export to strings that have not yet been translated.
3. After exporting the file, add translations for each field to be localized in the file.
4. Import the translated file(s) back into the Galaxy.
5. For InTouch HMI, import the translated language files into InTouch.
   - In stand-alone InTouch, you can perform a single import operation.
   - In managed InTouch, you must import the language file into each managed InTouch application.
   - For InTouch applications that only target subareas of the Galaxy, exporting by Area is a more optimal workflow.

   Since AVEVA OMI applications access the imported files directly from Application Server, there is no additional import required for AVEVA OMI applications.

Export language data for internationalized strings

You can export language data for all industrialized strings contained in a Galaxy, or just those in selected objects. The following attributes can be localized:

- Aliasname

  Aliasname is also used by the AVEVA OMI NavTree and Breadcrumb navigation apps, and these apps recognize localized Aliasname attributes.
- Attribute.Description
- Boolean OnMsg
- Boolean OffMsg
- EngUnits
- ShortDesc
- Any User Defined Attribute that is of type InternationalizedString

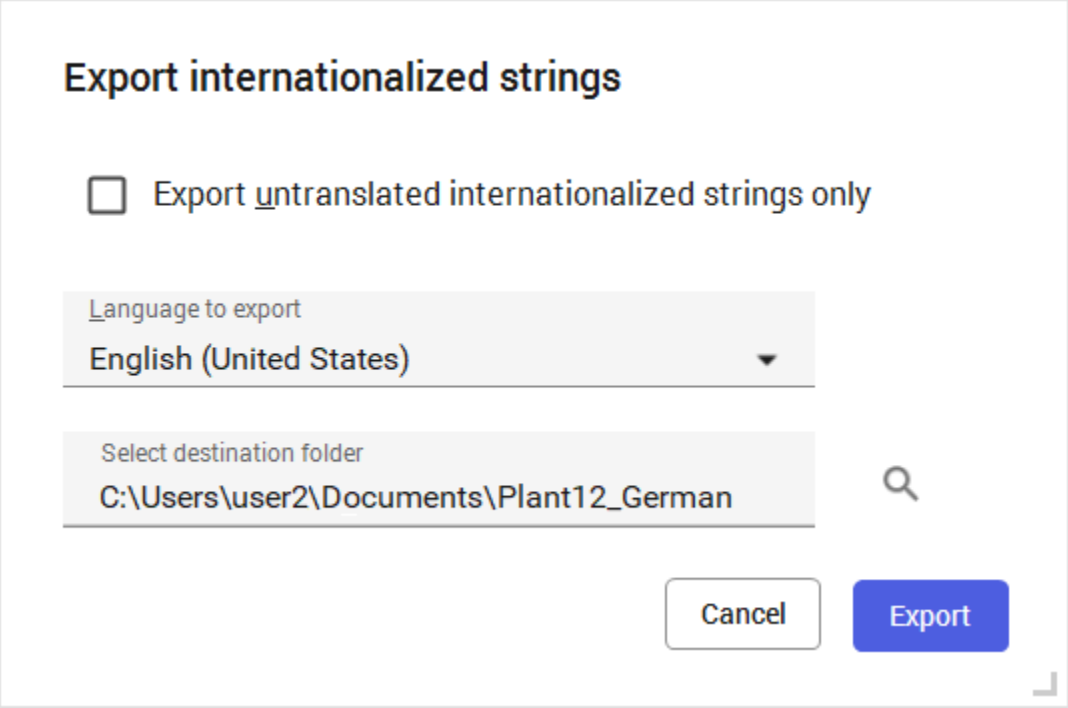
© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

The export operation only applies to objects that are checked in.

To export InternationalizedString attributes

1. Open the Galaxy from which you want to export strings.
2. If you only want to export the attributes contained in certain objects, select the objects now.
3. From the ribbon, select **Galaxy**, then select **Export**.
4. Select **Localization**. Select either:
   - **All Internationalized strings** to export all attributes of type in the Galaxy.
   - **Internationalized strings from selected objects** to export InternationalizedString attributes from the selected objects.

   The **Export internationalized strings** dialog box appears.



5. Configure the export settings.
   - In the **Language to export** list, select the language file to export. Only languages that have been configured for the Galaxy are listed.
   - In the **Destination folder** box, enter a full path name, or browse to the folder to which you want to export the language file. Click the browse icon to select an existing folder or create a new folder.
   - To export only attributes that have not been previously translated (differential export), enable the checkbox to "Export untranslated internationalized strings only."
6. Click **Export**. The export progress is shown as the file is created.
7. When file creation is complete, click **Close**.

Export language data for all Industrial Graphics in a Galaxy

You can export language data for all Industrial Graphics in a Galaxy at one time. The export contains language

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

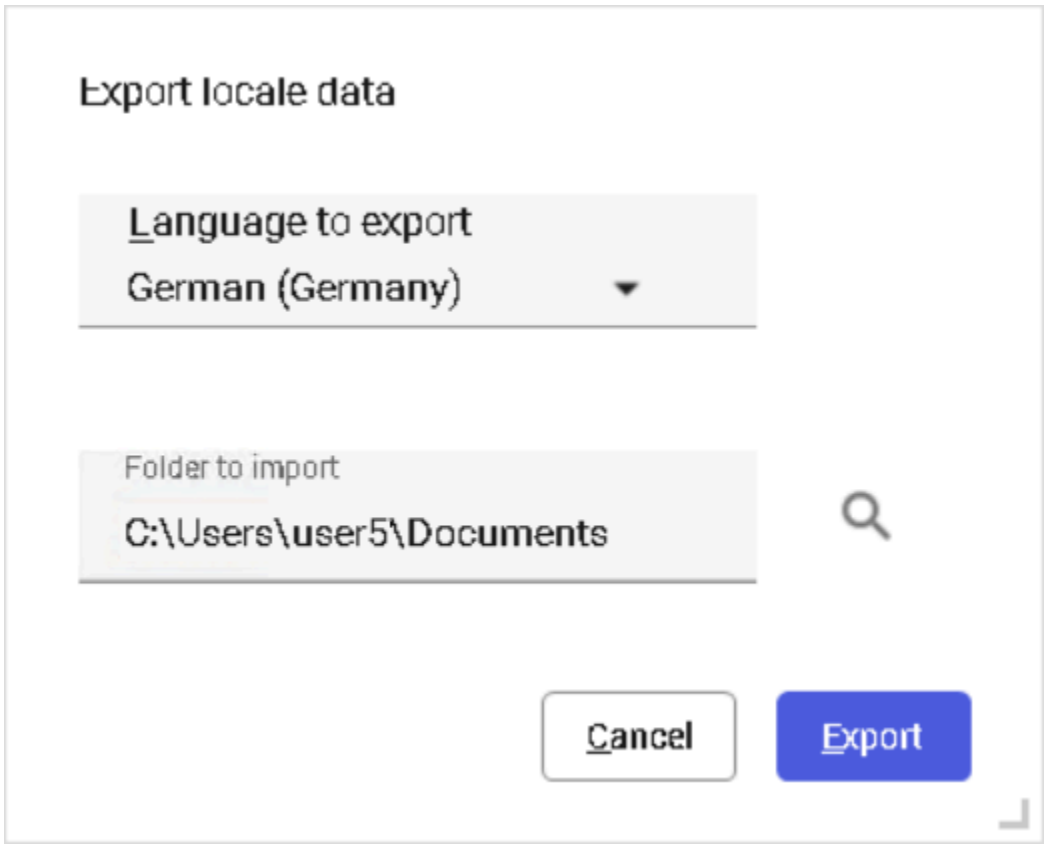
data for:

- Visualization folder graphics.
- Graphics contained in AutomationObject templates, except for an $InTouchViewApp template.
- Graphics contained in AutomationObject instances.

The export operation only applies to graphics that are checked in.

To export language data for all Industrial Graphics

1. Using the System Platform IDE, open the Galaxy for which you want to export graphic text.
2. From the ribbon, select **Galaxy**, then select **Export**.
3. Select **Localization**, then **All Industrial Graphics**. The **Export Locale Data** dialog box appears.



4. Configure the export settings.
   - In the **Languages to export** list, select the language dictionary to export. The default language is not listed.
   - In the **Path** box, type the folder to which you want to export the dictionary. Click the browse icon to select an existing folder or create a new folder.
5. Click **Export**. The export progress is shown as the XML file is created.
6. When file creation is complete, click **Close**.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Export language data for specific objects

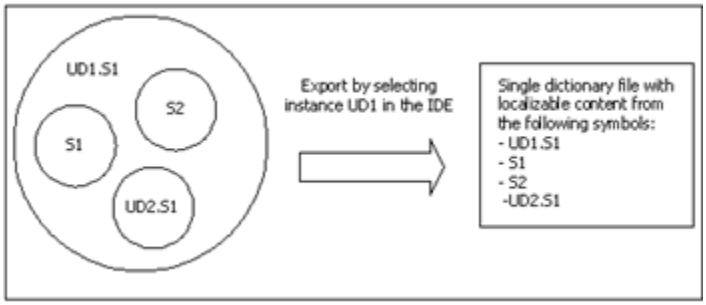
You can export language data for the following types of Galaxy objects:

- Visualization folder graphics
- AutomationObject templates and instances
- $InTouchViewApp templates and instances.

All graphics in an instance or template are exported.

When you export the language data for a graphic, the language data for all graphics referenced by the graphic is also exported. The exported graphics can be referenced through direct embedding or through a show graphic animation.

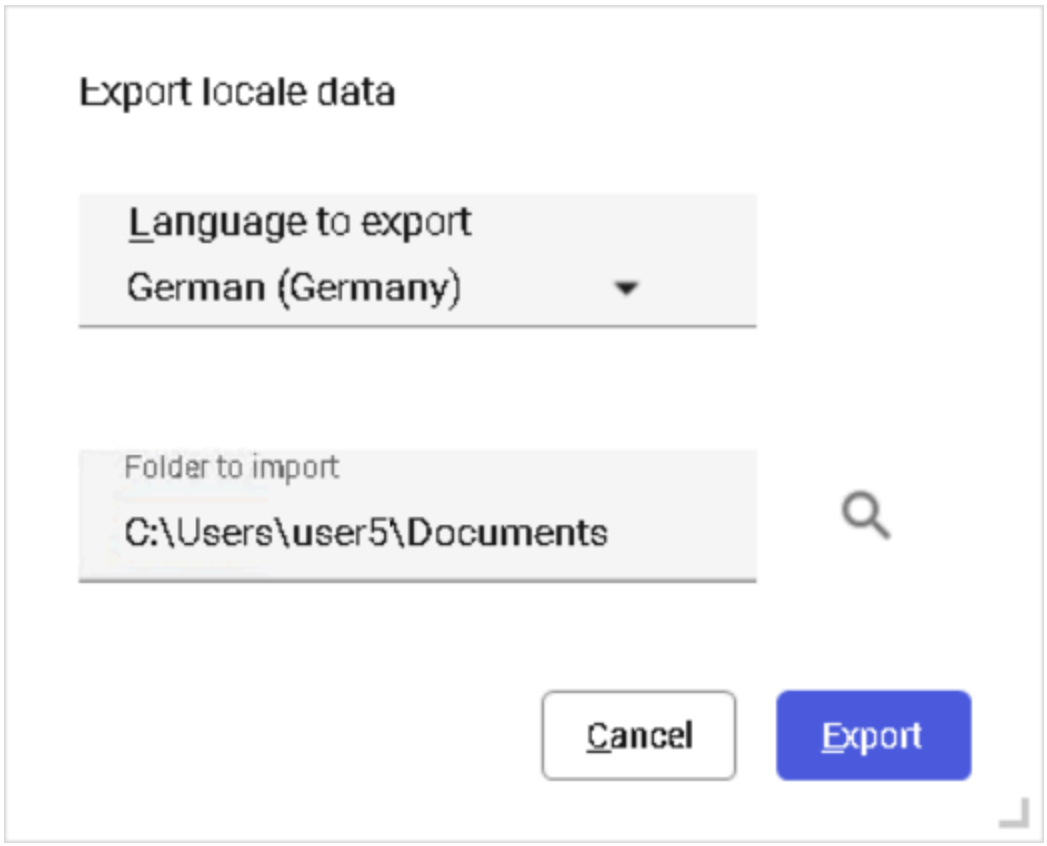
For example, a Galaxy has Symbol1 and Symbol2 defined in the Visualization folder. There are two instances called UserDefined1 and UserDefined2. UserDefined1 includes Symbol1. UserDefined2 includes Symbol1 and Symbol2. The instance graphic UserDefined1.Symbol1 embeds Symbol1 and Symbol2 from the Visualization folder and one instance graphic UserDefined2.Symbol1. If you select the instance UserDefined1 and export the language data, then the language data would also be exported for the graphics Symbol1, Symbol2, and UserDefined2.Symbol1.



To export language data for specific objects

1. Using the System Platform IDE, open the Galaxy for which you want to export graphic text.
2. Select one or more objects to export.
3. From the ribbon, select **Galaxy**, then select **Export**.
4. Select **Localization,** and then select **Selected Industrial Graphics**. The **Export Locale Data** dialog box appears.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.



5. Configure the export settings.
   - In the **Language to export** list, select the language dictionary to export. The default language is not listed.
   - In the **Path** box, type the folder to which you want to export the dictionary. Click the browse icon to select an existing folder or create a new folder.
6. Click **Export**. The export progress is shown as the XML file is created.
7. When file creation is complete, click **Close**.

Export graphic language data for a managed InTouch application

You export language data for a managed InTouch application using the IDE. You cannot export translations from within the InTouch HMI.

The export includes strings for:

- All InTouch windows
- All SmartSymbols
- Industrial Graphics referenced by the $InTouchViewApp template

The export causes a cascade export of all referenced Industrial Graphics.

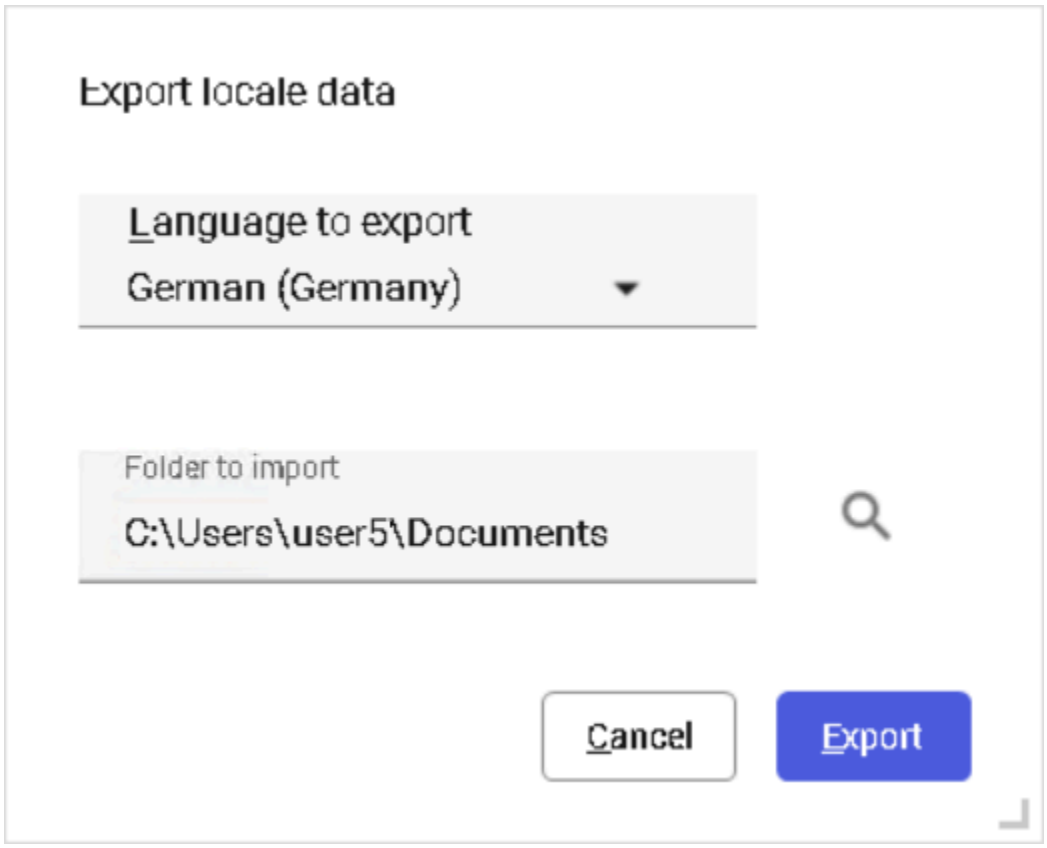
When you export language data for a managed InTouch application, the default language for the Galaxy is ignored. The InTouch default locale is always the installed InTouch locale. If the InTouch installed locale and the Galaxy default language are not the same, the export is skipped for the selected InTouchViewApp and a message

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

is logged.

To export language data for a managed InTouch application

1. Using the System Platform IDE, open the Galaxy for which you want to export graphic text.
2. Select one or more managed InTouch applications.
3. From the ribbon, select **Galaxy**, then select **Export**.
4. Select **Localization,** and then select **Selected Industrial Graphics**. The **Export Locale Data** dialog box appears.



5. Configure the export settings.
   - In the **Language to export** list, select the language dictionary to export. The default language is not listed.
   - In the **Path** box, type the folder to which you want to export the dictionary. Click the browse icon to select an existing folder or create a new folder.
6. Click **Export**. The export progress is shown as the XML file is created.
7. When file creation is complete, click **Close**.

Export graphic language data for a published InTouch application

If you export languages for a published InTouch application, the following are not included:

- Embedded Industrial Graphics
- Custom property overrides

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

- String overrides.

The export only includes native InTouch translations.

Export text to an existing dictionary file

If you change your application after you translate the text strings, you need to export the text again. For more information, see Export files for offline translation.

If you export a file with the same name to the same directory, you will be asked to confirm that you want to overwrite the existing file.

If you click **Yes**, the existing file is deleted and the current text for graphics in the Galaxy are exported as a new file.

Any existing translations for a graphic are reflected in the new .xml file.

Export alarm comments for offline translation

You typically send alarm comment text out for bulk translation. You can export alarm comments and modify them using a text editor or a spreadsheet program such as Microsoft Excel. After translation, the files can be re-imported to the Galaxy for runtime alarm comment language switching.

Guidelines and recommendations

The following guidelines and recommendations will help you make best use of the alarm comment language switching feature:

Important: The exported file name is generated automatically and must not be changed.

Organize your export

Follow these recommendations when exporting alarm comments for translation:

- You must export alarm comment text for only one language at a time.
- When you export text, you must specify a folder to place the language files. We recommend that you create a separate folder for each language. For example, ...\Galaxy01\ChineseFiles\.
- For large Galaxies, exporting files for each area, after being translated and re-imported in the Galaxy, will perform faster when the alarm clients access those files.
- For small Galaxies, we recommend that you export the entire Galaxy for translation.

Translation file formatting and editing

Microsoft Excel is recommended for translation file formatting and editing, since it supports large worksheets and provides enhanced formatting capabilities to convert the text file into tabular format. See Export alarm comments from very large galaxies.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Reimport

- We highly recommend re-importing the translated files to the Galaxy after translation. Although re-import is not mandatory, you can use the re-import to optimize the export files.
- If you make changes to your alarm comments after you export comment text, you must export the alarm comments again. For this reason, we further recommend using the import functionality to reimport the language file after each translation. This practice will avoid the need to re-translate alarm comments you have already translated.

For further information and procedures, see the following sections:

Export alarm comments from very large galaxies

Export alarm comments by area

Import translated alarm comment language files

About the alarm comments language file

The alarm comment language switching feature exports alarm comments, messages and other data to a .txt file. That file can be edited directly, but working in a text editor is typically more difficult and prone to errors.

- All files export as Unicode. We recommend that you save the exported files as Unicode.
- We recommend that you edit the .txt file using Microsoft Excel. Be sure to save the edited file as a .txt file.

The language file name is not user-configurable. The file naming convention for alarm comment export is:

Galaxy_<GalaxyName>_<localeID>_Alarm_Comments.txt

or

Galaxy_<GalaxyName>_<AreaName>_<localeID>_Alarm_ Comments.txt

where the locale ID is the selected language decimal identification number.

For example, if the Galaxy name is "TestSample" and the exported language is Chinese, the language file name would be Galaxy_TestSample_2052_Alarm_Comments.txt. The locale ID for Chinese is 2052.

For example, if the Galaxy name is "TestSample" and the Area name is "Area001", and the exported language is German, the language file name would be Galaxy_TestSample_Area001_1031_Alarm_Comments.txt.

See Translate exported alarm comment language files for further information about alarm comment language files.

Export alarm comments from very large galaxies

You can export language data in one operation for all alarm comments in a Galaxy. However, very large Galaxies with numerous alarm comments require longer export processing time and can result in a very large .txt language file. For handling large Galaxies and a large number of alarm comments, we recommend that you export alarm comments by Area rather than exporting alarm comments for the entire Galaxy to a single language file. See Export alarm comments by area.

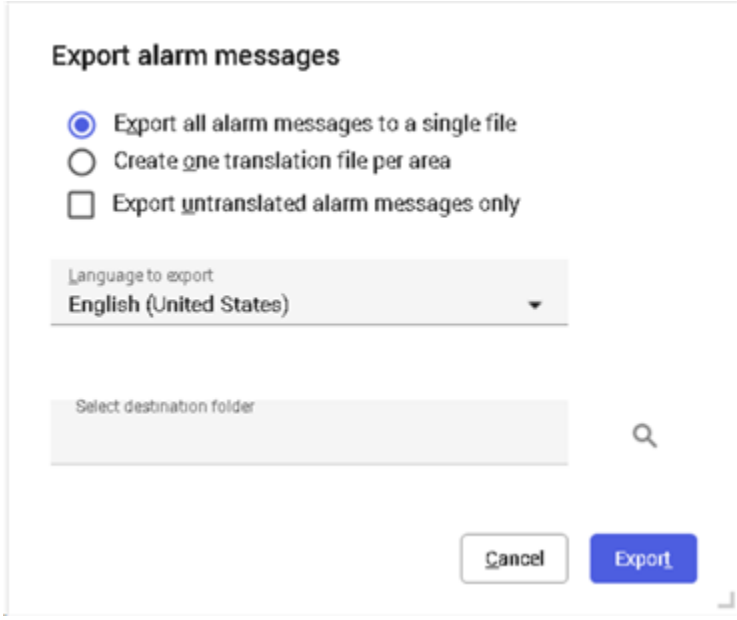
© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Export all galaxy alarm comments

You can export language data for all alarm comments in a Galaxy at one time. The export operation only applies to objects that are checked in.

To export all Galaxy alarm comments to a single file

1. Using the System Platform IDE, open the Galaxy in which you want to export alarm comments.
2. From the ribbon, select **Galaxy**, then select **Export**.
3. Select **Localization**, then select **Alarm Fields**. The **Export Alarm Messages** dialog box appears.



4. Configure the export settings:
   a. Select the **Export all alarm messages to a single file** radio button.
   b. Check the **Export untranslated alarm messages only** check box if you have already exported, translated and re-imported your language file and you want to export only newer, untranslated messages. Otherwise, leave the check box unchecked.
   c. In the **Language to export** pull-down list, select the language to export. Make sure that you have configured the Galaxy for the language that you want to export, since the export file will contain the locale ID of the language you select. You can export only one language at a time.
   d. In the **Path** box, type the folder to which you want to export the language file. Click the browse icon to select an existing folder or create a new folder.
5. Click **Export**. The export progress box appears.
6. Click **Close** when the export completes.

Export alarm comments by area

You can export all alarm comments by Area or you can export Alarm comments for a specific Area.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

About file names

The *Galaxy_<GalaxyName>_<AreaName>_<localeID>_Alarm_ Comments*.txt file name convention applies to specific Area alarm comment exports. Area names will be included the file name. For example, given a Galaxy name "TestSystem", an Area name "Area001", and an export to Chinese (language designation 2052), the language file name would be Galaxy_TestSystem_Area001_2052_Alarm_Comments.txt.

Export objects not assigned to an area

System Objects typically are not assigned to an Area. These are:

- Platform Objects
- Device Integration Objects
- Engine Objects

For purposes of language export, these System Objects form their own "Area" or become notification distributors for the purpose of sending alarms. Exporting System Objects follows these important conventions:

- System Objects can only be exported from the **Galaxy** menu. They cannot be selected and exported using a context menu.
- System Objects can only be exported by exporting alarm comments for all areas.

To export alarm comments for all areas

1. Using the System Platform IDE, open the Galaxy for which you want to export alarm comments.
2. From the ribbon, select **Galaxy**, then select **Export**.
3. Select **Localization**, then select **Alarm Fields**. The **Export Alarm Messages** dialog box appears.
4. Configure the export settings:
   a. Select the **Export all alarm messages to a single file** radio button.
   b. Check the **Export untranslated alarm messages only** check box if you have already exported, translated and re-imported your language files by Area and you want to export only newer, untranslated messages. Otherwise, leave the check box unchecked.
   c. In the **Language to export** pull-down list, select the language to export. You can export only one language at a time.
   d. In the **Destination folder** box, type the folder to which you want to export the language file. Click the browse icon to select an existing folder or create a new folder.
5. Click **Export**. The export progress box appears.
6. Click **Close** when the export completes.

Translate exported graphic language files

The procedures and tools for translating the exported language files are similar for both graphic text and for alarm comments. However, there are important differences. Procedures for each are described in this section.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Translate exported graphic text dictionary files

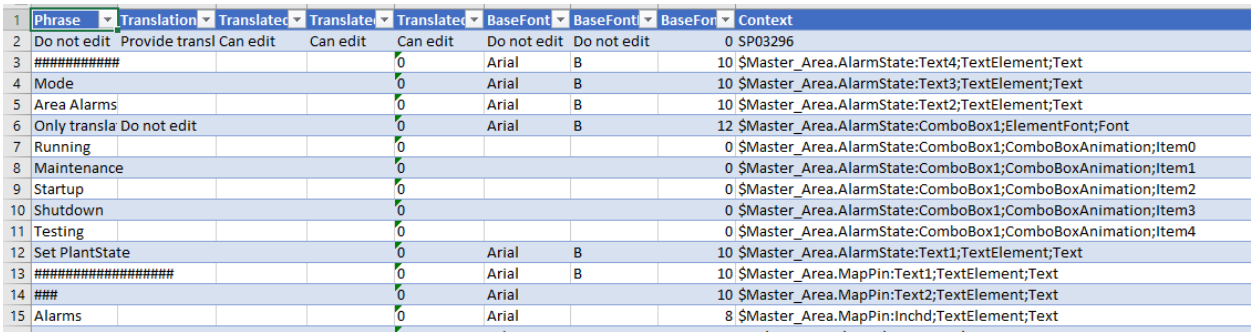
After you export the dictionary file containing your application text, use Microsoft Excel to edit the text.

To translate an exported dictionary file

1. Open the XML file in Excel. The **Open XML** dialog box appears.
2. Click **As an XML** list, then click **OK**. A message may appear informing you that an XML schema will be created.
3. Click **OK**.

   The XML file opens in Excel with columns for the:

   - Phrases in your application.
   - Translated phrases from the translator.
   - Translated font name.
   - Translated font properties.
   - Translated font size.
   - Base font properties.
   - Base font size.
   - Context, phrase ID, language ID and foreign language ID.



	Phrase	Translation	Translated	Translate	Translated	BaseFont	BaseFont	BaseFon	Context
2	Do not edit	Provide transl	Can edit	Can edit	Can edit	Do not edit	Do not edit	0	SP03296
3	##########				0	Arial	B	10	$Master_Area.AlarmState:Text4;TextElement;Text
4	Mode				0	Arial	B	10	$Master_Area.AlarmState:Text3;TextElement;Text
5	Area Alarms				0	Arial	B	10	$Master_Area.AlarmState:Text2;TextElement;Text
6	Only transla	Do not edit			0	Arial	B	12	$Master_Area.AlarmState:ComboBox1;ElementFont;Font
7	Running				0			0	$Master_Area.AlarmState:ComboBox1;ComboBoxAnimation;Item0
8	Maintenance				0			0	$Master_Area.AlarmState:ComboBox1;ComboBoxAnimation;Item1
9	Startup				0			0	$Master_Area.AlarmState:ComboBox1;ComboBoxAnimation;Item2
10	Shutdown				0			0	$Master_Area.AlarmState:ComboBox1;ComboBoxAnimation;Item3
11	Testing				0			0	$Master_Area.AlarmState:ComboBox1;ComboBoxAnimation;Item4
12	Set PlantState				0	Arial	B	10	$Master_Area.AlarmState:Text1;TextElement;Text
13	##################				0	Arial	B	10	$Master_Area.MapPin:Text1;TextElement;Text
14	###				0	Arial		10	$Master_Area.MapPin:Text2;TextElement;Text
15	Alarms				0	Arial		8	$Master_Area.MapPin:Inchd;TextElement;Text

Important: Only modify data in the **Translation**, **TranslatedFontSize**, **TranslatedFontName**, and **TranslatedFontProperty** columns. Do not change any column header. Do not insert or delete rows.

4. Type the language-specific text in the **Translation** column in the row that corresponds with the base language string in the **Phrase** column.
5. If necessary, change the font parameters for the translated strings. If you only provide a translation, the Galaxy-configured font for the language is used to render the text after the translation is imported. If you specify a font, it overrides the Galaxy-configured font.
   - In **TranslatedFontName** column, type the font name.
   - In the **TranslatedFontProperty** column, type the notation for the font properties:

     B = **bold**

     I = *italic*

     U = underline

     For example, if you want to the text to be bold, type **B** in the **TranslatedFontProperty** column. If you want the text to be bold and underlined, type **BU** in the **TranslatedFontProperty** column.
6. Save the file using XML Data as the file type.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Important: If you save as another file type, such as XML Spreadsheet, Excel changes the schema and the Galaxy cannot load the file. If you change the name of the XML file, the file will not import properly into the Galaxy.

Translate exported alarm comment language files

Alarm comments are exported to a .txt file, located in the selected directory. You can edit the files with a text editor, but we recommend Microsoft Excel as the editor. Using the format-as-table feature in Excel greatly simplifies editing the specific text to be translated.

To translate an exported alarm comment file

1. Open the .txt file in Excel. The Excel **Text Import Wizard** appears. Follow the wizard instructions to import the text file in tab delimited, general column data format. The file appears unformatted in Excel.

Tip: You can open the file in text editor, and then copy and paste the file contents into a blank Excel worksheet instead of using the import wizard. The results will be the same

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

	A	B	C	D	E	F	G	H	I	J	K	L
1	Unique Phrases:											
2												
3	PhraseId (	Default M	Translated Message (EDIT THIS COLUMN)									
4	3	AppEngine										
5	4	Enterprise										
6	5	OMI_ViewApp_Desktop										
7	6	Plant										
8	7	Plant_Area										
9	8	Platform										
10	9	Simulator										
11	10	Site										
12	11	Sys										
13	12	The AppEngine hosts and schedules execution of Application Objects, Areas, and Device Integration Objects.										
14	13	The OPCClient provides connectivity to OPC servers.										
15	15	ViewEngine										
16	16	Device write failed -										
17	17	Device write failed - Array index is out of range -										
18	18	Device write failed - Attribute not writable -										
19	19	Device write failed - Data value out of range -										
20	20	Device write failed - Incorrect data type -										
21	21	Device write failed - Invalid Attribute -										
22	22	Device write success -										
23	23	started										
24	24	started successfully.										
25	25	Stopped										
26	26	stopped successfully.										
27	27	Write access denied -										
28	28	Write access denied - Alarm ACK has already been done -										
29	29	Write access denied - Array index is out of range -										
30	30	Write access denied - Attribute not writable -										
31	31	Write access denied - Data value out of range -										
32	32	Write access denied - Incorrect data type -										
33	33	Write access denied - User does not have Alarm ACK privileges -										
34	34	Write access denied - User does not have permission to modify this type of attribute -										
35	35	Write access denied - User does not have permission to write to this type of attribute										
36	36	Write access denied - Users cannot modify configuration attributes while object is onscan -										
37	37	Write access denied - Verifier does not have permission to verify write for this type of attribute										
38	38	Write failed - Invalid Attribute -										
39	39	Write success -										
40												
41												
42	Alarms:											
43												
44	PhraseId (	Alarm (DO NOT EDIT)										
45	3	[_GROUP_NAME_] AppEngine										

2. Format the text as a table in Excel:
   a. Select all of the rows containing data in Column A through Column C.
   b. Select the **Home tab**, then select **Format as Table**.
   c. Select the color pattern you want and click **OK** in the **Format as Table** dialog box. Excel will format the region to be edited. Resize the columns as needed.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

	A	B	C
1	Column1	Column2	Column3
2	Unique Phrases:		
3			
4	PhraseId (DO NOT EDIT)	Default Message (DO NOT EDIT)	Translated Message (EDIT THIS COLUMN)
5	3	AppEngine	*<localized text message 1>*
6	4	Enterprise	*<localized text message 2>*
7	5	OMI_ViewApp_Desktop	*<localized text message 3>*
8	6	Plant	*<localized text message 4>*
9	7	Plant_Area	*<localized text message 5>*
10	8	Platform	*<localized text message 6>*
11	9	Simulator	*<localized text message 7>*
12	10	Site	*<localized text message 8>*
13	11	Sys	*<localized text message 9>*
14	12	The AppEngine hosts and schedules execution of Application Objects, Areas, and Device Integration Objects.	*<localized text message 10>*
15	13	The OPCClient provides connectivity to OPC servers.	*<localized text message 11>*
16	15	ViewEngine	*<localized text message 12>*
17	16	Device write failed -	*<localized text message 13>*
18	17	Device write failed - Array index is out of range -	*<localized text message 14>*
19	18	Device write failed - Attribute not writable	*<localized text message 15>*
20	19	Device write failed - Data value out of range -	*<localized text message 16>*
21	20	Device write failed - Incorrect data type -	*<localized text message 17>*
22	21	Device write failed - Invalid Attribute -	*<localized text message 18>*
23	22	Device write success -	*<localized text message 19>*
24	23	started	*<localized text message 20>*
25	24	started successfully.	*<localized text message 21>*
26	25	Stopped	*<localized text message 22>*
27	26	stopped successfully.	*<localized text message 23>*
28	27	Write access denied -	*<localized text message 24>*
29	28	Write access denied - Alarm ACK has already been done -	*<localized text message 25>*
30	29	Write access denied - Array index is out of range -	*<localized text message 26>*
31	30	Write access denied - Attribute not writable -	*<localized text message 27>*
32	31	Write access denied - Data value out of range -	*<localized text message 28>*
33	32	Write access denied - Incorrect data type -	*<localized text message 29>*
34	33	Write access denied - User does not have Alarm ACK privileges -	*<localized text message 30>*

The Excel display is divided into four regions:

- Column 1 contains Phrase ID numbers used internally in the language file. **Do not edit** this column.
- Column 2 contains the exported alarm messages. **Do not edit** this column.
- Column 3 contains the alarm message translations. Edit this column with your translation text.
- Below the alarm message text columns are the alarms and phrase IDs, which map to column 1 and are used internally in the language file. Do not edit this area.

3. Translate the alarm message text. Type the alarm message translations into the appropriate rows in column 3.

Important: Only edit the text in column 3, translations. No other text in the file may be edited. You can edit the original alarm comment text only within the System Platform IDE.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

4. Save the file as a .txt file in the directory you originally selected for export.

Important: Do not change the file name or the file type, otherwise it will not import correctly into the Galaxy.

Import translated language files

You can import alarm comment language files for re-export by area to facilitate the export of new, untranslated alarm comments, and to facilitate re-export of previously translated alarm comments that have been changed.

For graphic text, you must import the translated dictionary files for each language to enable runtime language switching for those languages. All dictionary files for a given language should be placed in the same folder.

You can import files for only one language at a time. When you import, you select the desired language and specify the files to import.

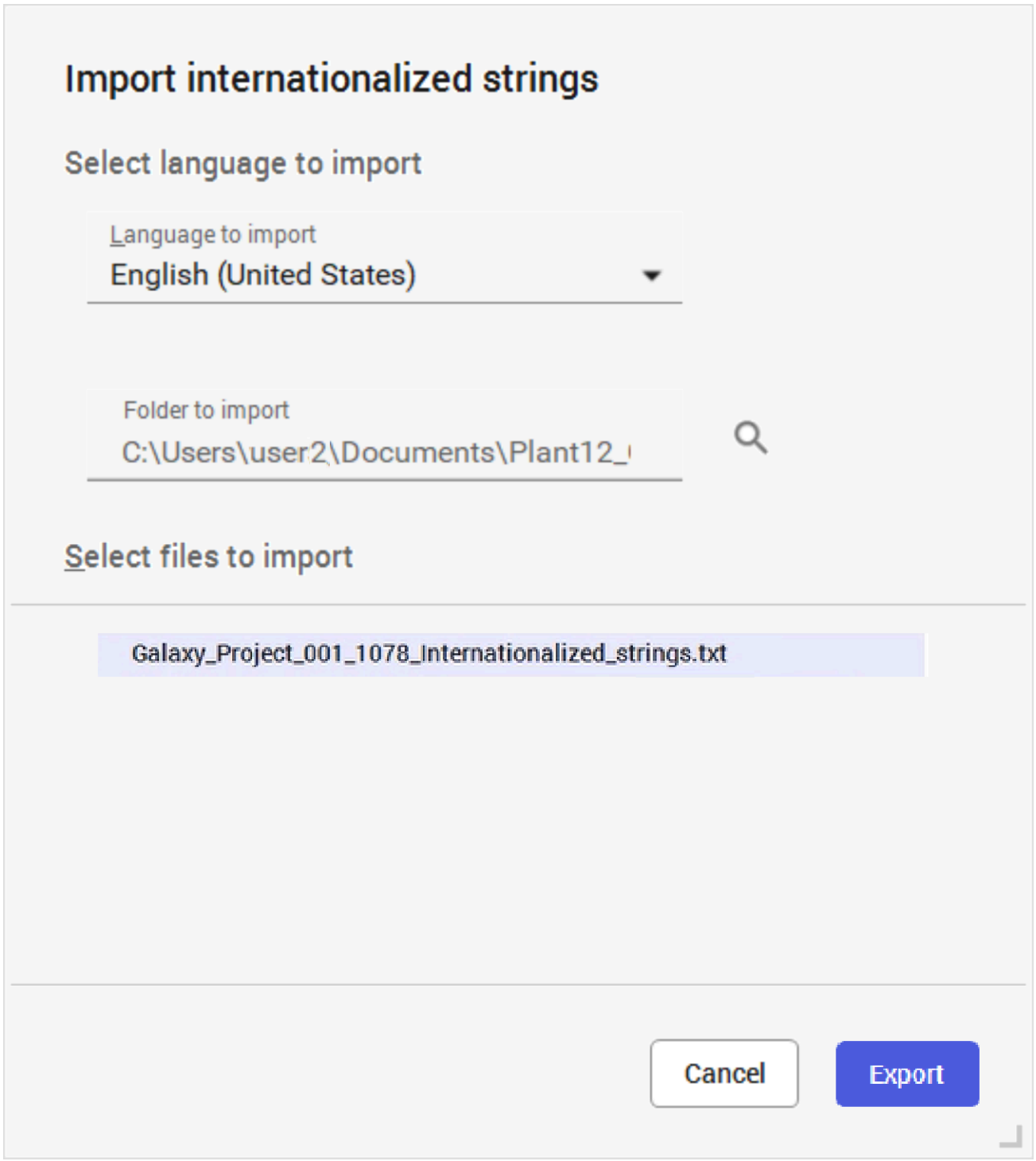
Importing Translated International Strings

To import internationalized string files, you must first have exported the strings for translation of some or all of the internationalized strings.

To import a translated internationalized string file

1. Open the Galaxy into which you want to import translated internationalized strings.
2. On the **Galaxy** menu, select **Import**, then select **Localization**, and then select the **Internationalized strings** menu option. The **Import Internationalized strings** dialog box appears.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.



3. Configure the import settings:
   a. In the **Language to import** pull-down list, select the language to import. You can import only one language at a time.
   b. In the **Folder to import** box, type or browse to the folder where you previously exported your language file. The available language files appear in the **Select files to import** list.
   c. Select a file or files to import.
4. Click **Import**. The import progress box appears.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

5. Click **Close** when the import completes.

Note: If an internationalized string does not match, a message is displayed for each failed string import.

Import translated graphic dictionary files

All affected graphics and relevant objects are checked out before the import begins and are checked in when the import is done. If an affected graphic or object is already checked out, a message appears in the progress dialog box and the import is skipped for the checked-out object.

For a published InTouch application, only the native InTouch translations are imported.

You can configure how you want Galaxy and graphic/object mismatches handled during the import.

The import is skipped if:

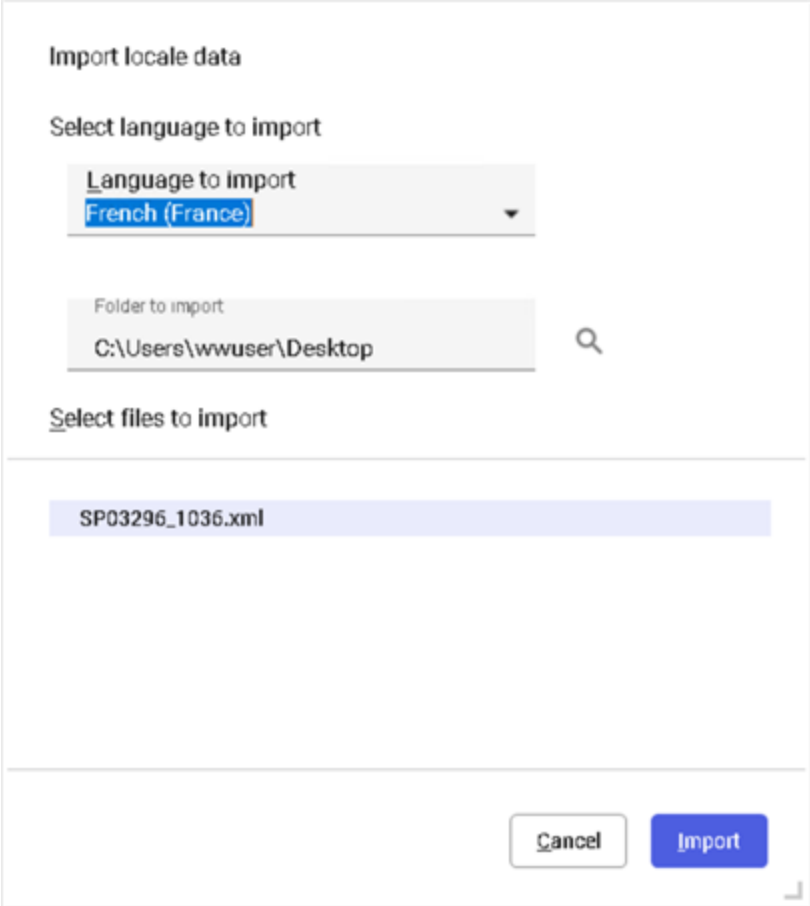
- The default locale specified in the .xml translation file does not match the current default locale of the Galaxy.
- An InTouchViewApp's installed locale does not match the current default locale of the Galaxy.

Important: You cannot cancel the import after it starts.

To import a translated dictionary file

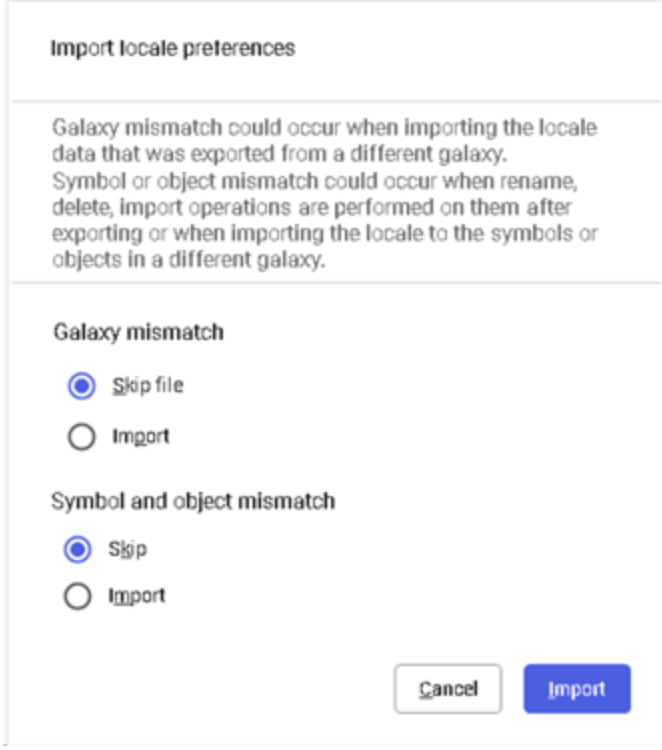
1. Using the System Platform IDE, open the Galaxy for which you want to import graphic text.
2. Close all editors and check in all Galaxy objects.
3. From the ribbon, select **Galaxy**, then select **Import**.
4. Select **Localization**, then select **Industrial Graphics translation**. The **Import locale data** dialog box appears.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.



5. Configure the import settings.
   - In the **Language to import** list, select the language dictionary to import.
   - In the **Folder to import** box, specify the folder that includes the dictionary file to import.
   - In the **Select files to Import** box, select the .xml files to import. Only files that include the current Galaxy name and the locale ID for the selected language are shown.
6. Click **Import**. The **Import Locale Preferences** dialog box appears.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.



7. In the **Galaxy mismatch** area, configure how you want Galaxy mismatches handled. A Galaxy mismatch occurs when you try to import a translation file that was exported from a different Galaxy. The Galaxy name in the translation .xml file is used to match the current name of the Galaxy.
   - Click **Skip file** to skip all the files that do not contain the current Galaxy name in the file name.
   - Click **Import** to import all the selected files, regardless of what the Galaxy name is in the .xml filename.
8. In the **Symbol/Object Mismatch** area, configure how you want graphic and object mismatches handled. A graphic or object mismatch occurs when the name of the graphic and the internal ID (GObjectID) of the graphic do not match what is within the .xml file. Objects include AutomationObjects and InTouchViewApp objects.
   - Click **Skip** to skip the graphics and objects that have mismatched names or IDs in the .xml file
   - Click **Import** to import a graphic or object only if it has a matching name or matching ID. If the name resolves to one object, and the ID resolves to another object, then the import is skipped.

   For examples, see Examples of graphic or object mismatch handling during language imports.
9. Click **Import**. The import progress is shown.
10. Click **Close**. The **Import Language Dictionary Files** dialog box appears.
11. Click **Check In**. The check in progress is shown.
12. Click **Close**. A summary of the import is shown.
13. Click **OK**.

Examples of graphic or object mismatch handling during language imports

The following table shows an example of handling mismatch conditions for a Visualization folder graphic. The bold name/ID is the matching name/ID in the .xml file and current Galaxy during import.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

	Graphic Name and ID in the Galaxy while Exporting	Graphic Name and ID in the Galaxy while Importing	Change made after export and before import	Skip Option Selected	Import Option Selected
1	S1, 100	**S1, 100**	No change	Import to S1	Import to S1
2	S1, 100	S2, **100**	Rename S1 to S2	Skip	Import to S2
3	S1, 100	**S1**, 200	Delete S1, Create/Import S1	Skip	Import to S1
4	S1, 100	**S1**, 200 S2, **100**	Rename S1 to S2 Create/Import S1	Skip	Ambiguous, skip import
5	S1, 100	No S1 and No 100 in the Galaxy	Deleted S1	Skip	Skip import

The following table shows an example of handling mismatch conditions for an AutomationObject graphic. The bold name/ID is the matching name/ID in the .xml file and current Galaxy during import.

	Graphic Name and ID in the Galaxy while Exporting	Graphic Name and ID in the Galaxy while Importing	Object Name and ID in the Galaxy while Importing	Change made after export and before import	Skip Option Selected	Import Option Selected
1	Pump1, 10	S1, 100	**Pump1, 10**	No change	Import to S1 only if Pump1 has S1 with an ID of 100	Import to S1 only if Pump1 has S1 with an id of 100
2	Pump1, 10	S1, 100	Pump2, **10**	Rename Pump1 to Pump2	Skip	Skip import if S1 and 100 pointing to two different graphics in Pump2 (ambiguous) Import to S1 only if Pump2 has S1 or a graphic with id of 100.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

	Graphic Name and ID in the Galaxy while Exporting	Graphic Name and ID in the Galaxy while Importing	Object Name and ID in the Galaxy while Importing	Change made after export and before import	Skip Option Selected	Import Option Selected
3	Pump1, 10	S1, 100	**Pump1**, 20	Delete Pump1 Create and Import Pump1	Skip	Skip import if S1 and 100 pointing to two different graphics in Pump1 (ambiguous) Import to S1 only if Pump1 has S1 or a graphic with id of 100.
4	Pump1, 10	S1, 100	**Pump1**, 20 Pump2, **10**	Rename Pump1 to Pump2 Create and Import Pump1	Skip	Ambiguous, skip import
5	Pump1, 10	S1, 100	No Pump1 and No 10 in the galaxy	Deleted Pump1	Skip	Skip import S1

Language data handling for galaxy operations

If you import or export objects, all language data within the associated graphics is imported or exported.

You can export all objects in the Galaxy, selected instances/templates, or selected graphics from the Visualization folder. All language data is exported as part of the graphics definition, regardless of the languages configured in the Galaxy.

You can import Galaxy objects that contain language data from the same Galaxy or a different Galaxy. All language data in the Industrial Graphics is imported, regardless of what languages are configured for the target Galaxy.

No Industrial Graphic or InTouchViewApp language data is read or modified during the import operation.

If you import an unmanaged InTouch application, the configured locales in the Galaxy are applied to the unmanaged InTouch application. No element translations in the InTouch application are removed during import, even though languages may no longer be visible in the InTouch HMI or available at runtime in WindowViewer.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

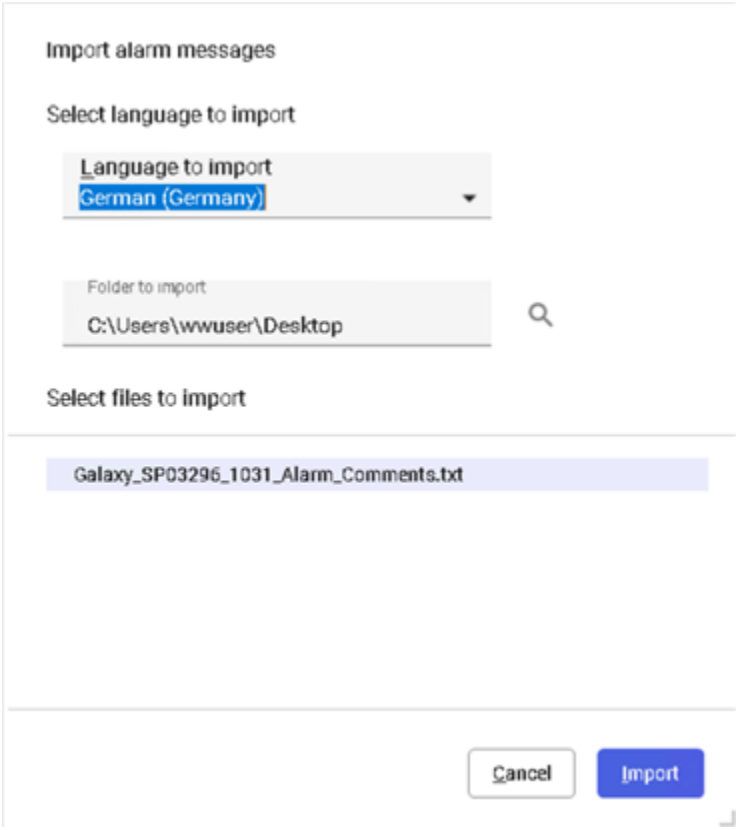
Import translated alarm comment language files

You can import translated alarm comment language files only from the **Galaxy** menu, not from the context menu for specific Areas.

To import alarm comment language files, you must first have exported alarm comments for translation of some or all of the alarm comments.

To import a translated alarm comment language file

1. Using the System Platform IDE, open the Galaxy in which you want to import translated alarm comments.
2. On the **Galaxy** menu, select **Import**, then select **Localization**, and then select the **Alarm Fields** menu option. The **Import Alarm Messages** dialog box appears.



3. Configure the import settings:
   a. In the **Language to import** pull-down list, select the language to import. You can import only one language at a time.
   b. In the **Folder to import** box, type or browse to the folder where you previously exported your language file. The available language files appear in the **Select files to import** list.
   c. Select a file or files to import.
4. Click **Import**. The import progress box appears.
5. Click **Close** when the import completes.

Note: If an alarm name or message do not match, the alarm comment import will display a message for each failed alarm message import.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Re-export alarm comments

After exporting and translating alarm comments, you can re-import them into Application Server to help maintain up-to-date alarm comment language files.

After adding new alarm comments that require translation or after modifying existing, already translated alarm comments, you must re-export the language file to update the translations.

Export new untranslated alarm comments

In the **Export Alarm Messages** dialog box, select the **Export untranslated alarm messages only** check box. This creates a file named _untranslated.txt. For example, exporting untranslated alarm comments in Chinese for a Galaxy name "TestSystem" would create a file named Galaxy_TestSystem_2052_Alarm_Comment_Untranslated.txt.

Export modified existing alarm comments

You can make changes to alarm comments that you previously exported, translated and imported back into Application Server. You must re-export the alarm comment language file(s) to update the translations.

In the **Export Alarm Messages** dialog box, select the same language option and folder you selected for the previous export. When asked, confirm that you want to overwrite the existing file.

Test the language switching functionality at run time

We recommend that you test the runtime language switching functionality. Follow the procedures outlined in this chapter to enable runtime language switching in your application. Then import the appropriate language file into InTouch. The import step applies to both stand-alone and managed InTouch applications.

To test the language switching functionality

1. Open the application in WindowViewer.
2. On the **Special** menu, point to **Language**, and then click the name of the language to switch to.

   The information from the corresponding translated dictionary file (if one exists and has been imported into InTouch) loads and appears.
3. When you are done, click **Close**.

Sequencer Object: Configuration and Use

This section helps you understand the basic features of the Sequencer object, what they do, and how they interact.

The Sequencer object is an application object. It lets you configure, execute, and manipulate a sequence of operations associated with attributes contained in an Application Server application.

You can use it to automate:

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

- Repetitive manufacturing procedures with a finite number of steps
- Supervisory processes with a finite number of steps

Sequencer object structure
Step structure
Steps and aliases naming conventions
Limitations

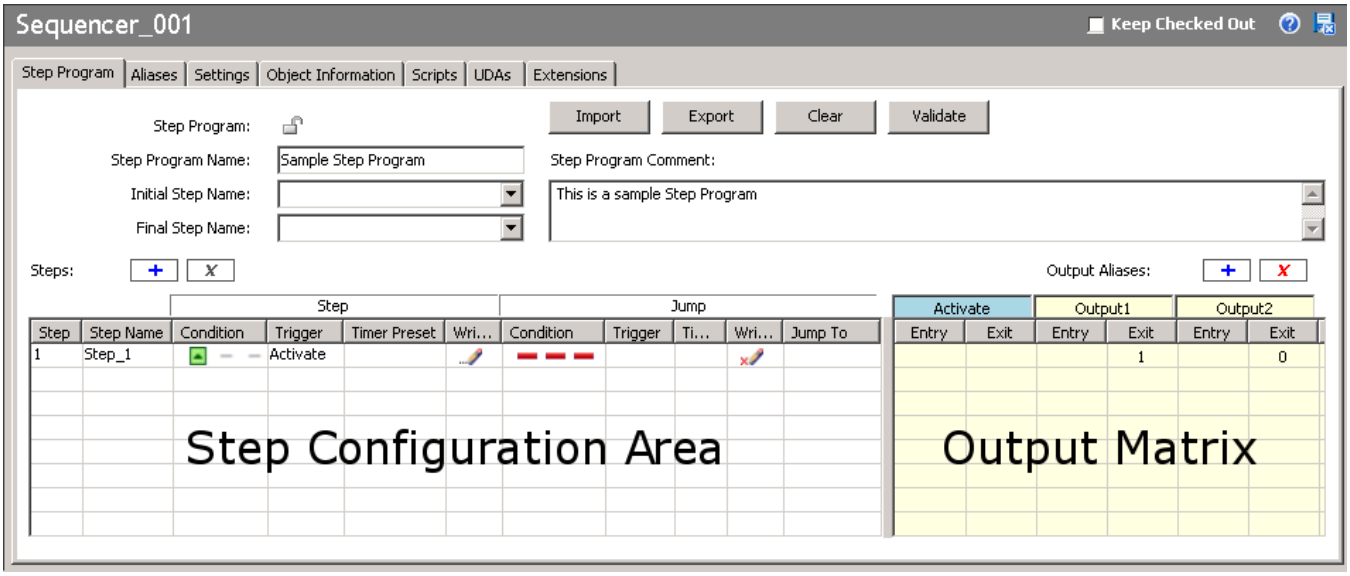
Sequencer object structure

A Sequencer object consists of three main parts:

- **Step program** - a collection of steps that define the reading and writing operations associated with object attributes via aliases. When in operation, the Sequencer object advances cyclically through the steps or jumps based on various events, conditions, and timers. Inputs are evaluated and values are written to defined outputs.
- **Aliases** - a list of names that are mapped to object attributes. The Sequencer object uses these names internally to associate inputs and outputs with the attributes.
- **Settings** - other settings that control the behavior of the Sequencer object.

Step program

The Sequencer object hosts a step program that has a finite number of steps. You can configure the step program in the following panel:



Steps

Steps are processed sequentially unless certain conditions are met that instruct the processing to continue at a specified step (Jump To).

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

The step that is being processed at any given time, is called the **Current Step**. There is only one current or active step at any given time.

The conditions for processing the next step or any specified step are controlled by events (triggers, timers, or a combination of triggers and timers).

The following diagram shows this:

Conditions for processing steps

Outputs

You can write values to attributes whenever a certain step becomes active or before Sequencer jumps to the next or another step.

Note: To write to an attribute, you must first map it to an Alias. For specifics, see Aliases.

Every step of the step program allows you to associate the step entry and/or the step exit with values that are written to the **Aliases**. These values form an **output matrix**.

An output value can be one of two types:

- **literal value** (number, string, date time, elapsed time or boolean)
- **value of another Alias** (referencing another object attribute)

Empty cells in the matrix mean that no write to the associated Alias takes place.

Literal values

You can configure a step to write literal values to attributes either when a step becomes active or after the step is triggered, either by the step condition or the jump condition. Some examples of literal values are 3.141, 17, "Start", and "Pause".

Note: Literal values that are either string, date time, or elapsed time data type must be enclosed by double quotes to be recognized as such. If they are not enclosed by double quotes they are considered to be Aliases. You can embed double quotes in literal strings without any escape character. For example: "Joe said "Hello"". Most special characters are accepted within double quotation marks, except for tabs and carriage returns.

Aliases

You can configure a step to write the value of another attribute to the output. In this case the output causes the step program to pass the value of one attribute to another attribute. This applies to any Alias you have defined in your step program.

Aliases

At run time, the Sequencer object interacts with attributes of other objects. These attributes can be:

- **Inputs** that cause the step condition or jump condition to be fulfilled. Typical inputs are sensory devices that trigger the step execution when a certain condition in a factory device is fulfilled. For example, a tank level sensor.

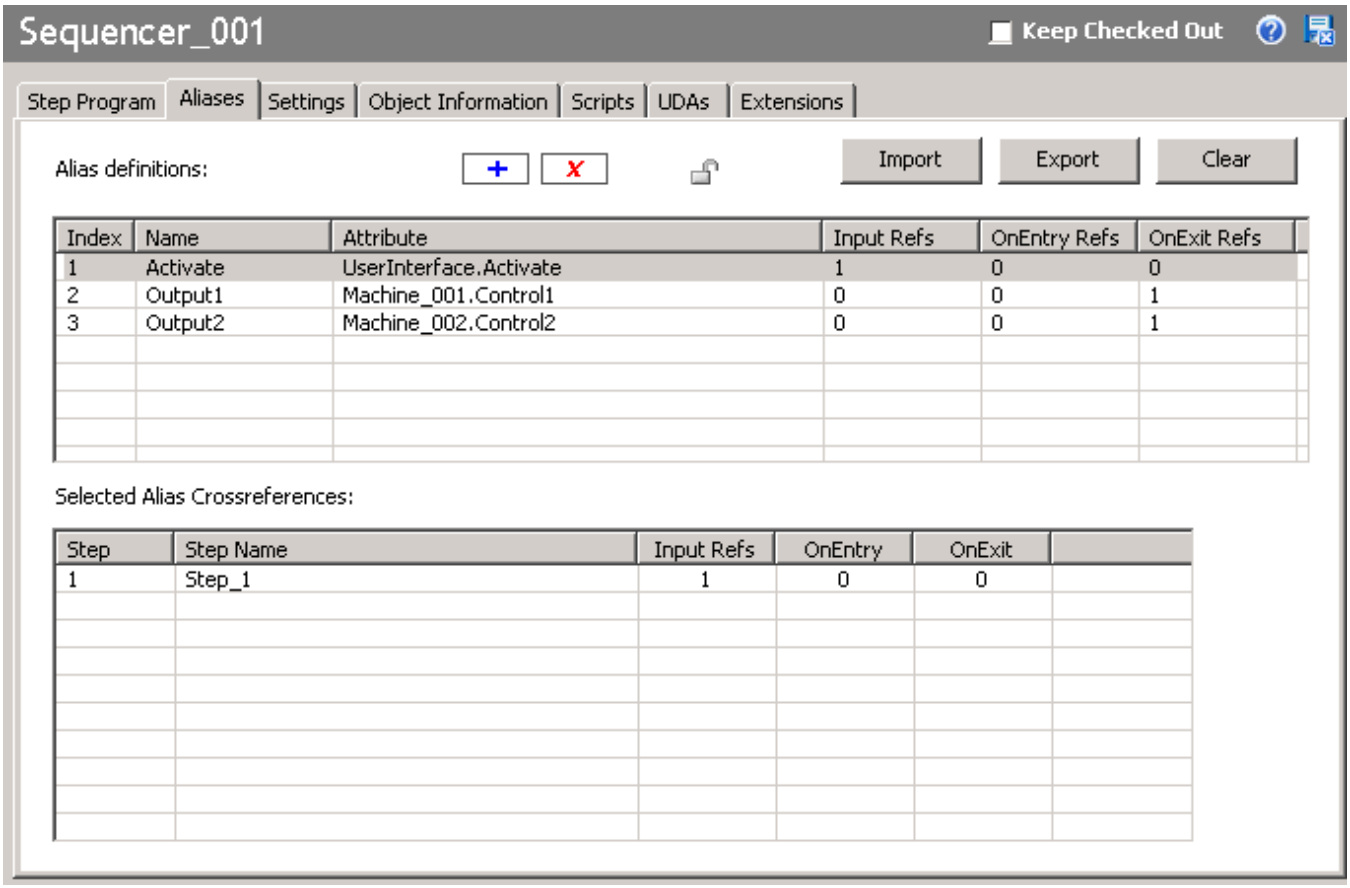
© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

- **Outputs** are written either before or after the step or jump condition is fulfilled. Typical outputs are to device controls, such as valve controls.
- **Inputs and outputs** at the same time. A typical example is a condition reset.

The step program can reference the same attribute multiple times in the step program. Instead of referencing these attributes directly, the Sequencer object uses a list of **Aliases** to reference the attributes. Using Aliases simplifies the appearance of the step program and saves time when the referenced attribute needs to be updated.

Note: Writes are not guaranteed. Only invalid references are reported. The Sequencer Object does not request or evaluate write confirmations.

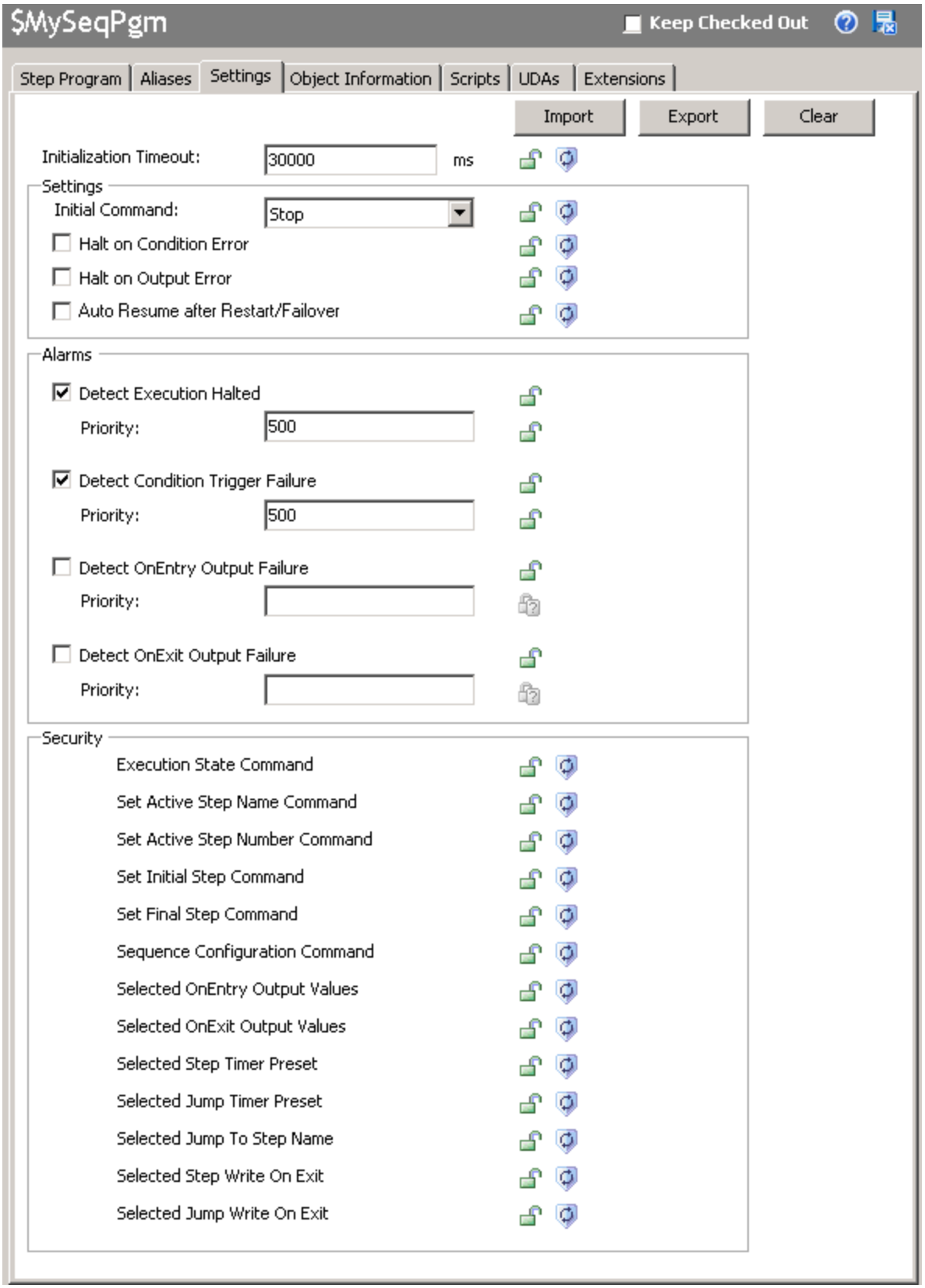
In the Sequencer object, you can configure the mapping of Aliases to attributes on the Aliases page.



Settings

The following screen capture shows the Settings panel:

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.



You can set generic parameters of the Sequencer object in the Settings panel, such as:

- **Settings** - you can set the initial command and the Sequencer behavior after errors and failover

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

- **Alarms** - you can instruct the Sequencer to generate an alarm if an error or a warning occurs
- **Security** - you can lock access to certain Sequencer features at design time and run time

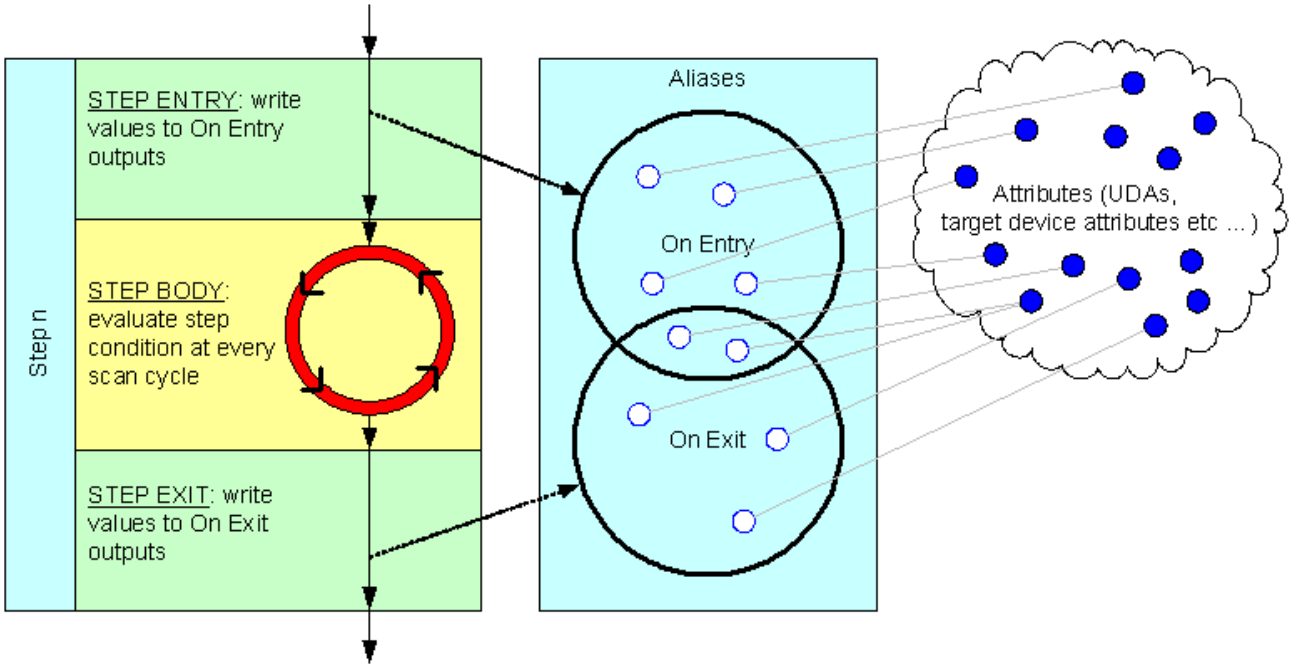
Step structure

This section describes the functionality embedded in every step.

Step entry, step body and step exit

Every step includes a:

- **Step entry** section where values are written to attributes after the step is activated. The On Entry outputs are processed on the first scan cycle when the step becomes active.
- **Step body** section where the step condition and jump condition are continuously evaluated. Step and jump conditions are not executed until the second scan of the step.
- **Step exit** section where values are written to attributes after the step condition or jump condition is fulfilled. The On Exit outputs are processed in the same scan cycle when the step condition or jump condition is fulfilled.



Note: Once a step is active, it is evaluated once per scan cycle until it is no longer the active step.

Minimal steps

Minimal steps are empty steps with

- an Always True step condition
- no jump condition

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

- no values being written to outputs

The minimal step can be considered a No Operation (NOP) Step. The Sequencer stays in an empty step exactly two engine execution cycles before advancing to the next step. No Operation (NOP) steps are useful for inserting dummy steps for future enhancements.

Step condition and jump condition

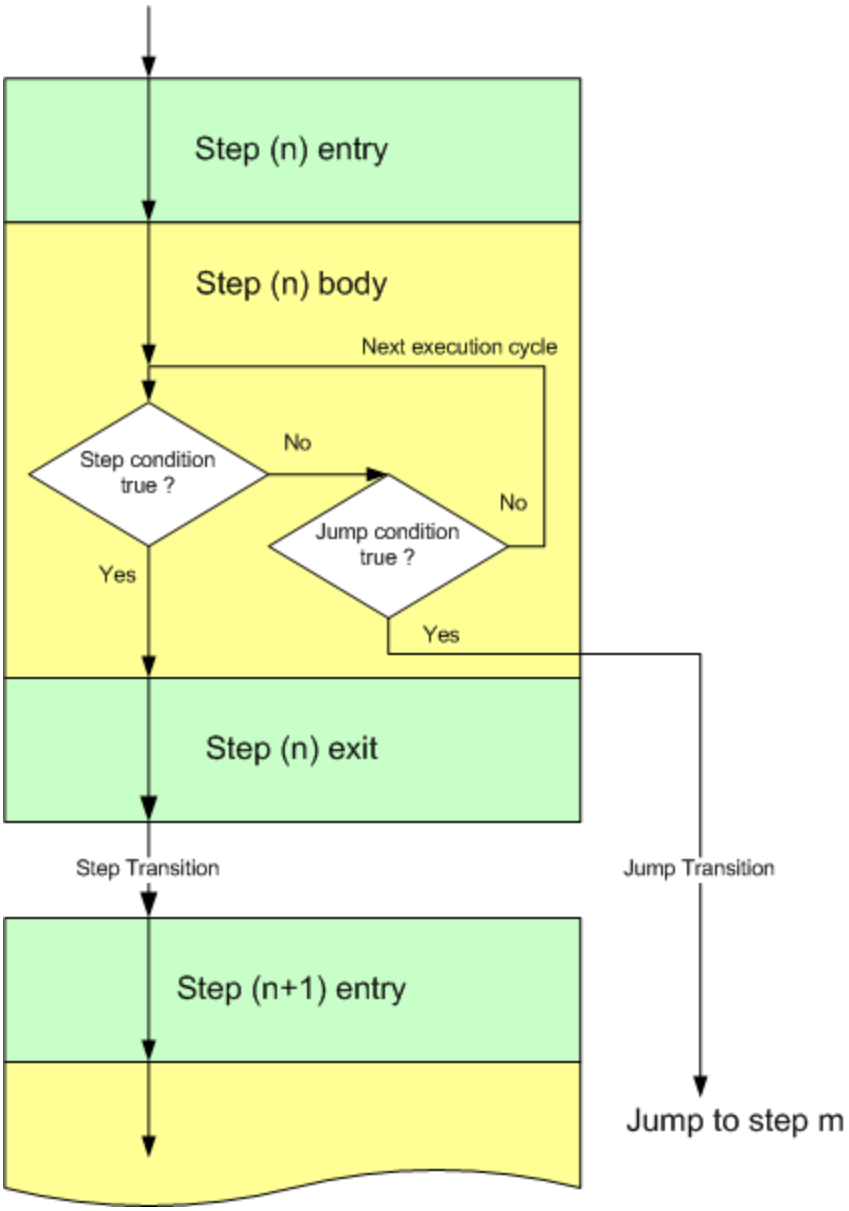
Every step body contains a:

- **Step condition** - if the step condition is fulfilled, Sequencer continues processing at the next step
- **Jump condition** - if the jump condition is fulfilled, Sequencer continues processing at a specified step (which can also be the current step).

  Note: If both the step condition and the jump condition are true, advancing to the next step takes precedence over the jump.

The following diagram shows this:

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.



By default:

- **Jump Write OnExit** is False, which means On Exit output values are not written for a jump transition.
- **Step Write OnExit** is True, which means On Exit output values are written for a step transition.

You can change this setting for any given step in your program.

A single step with a jump condition is processed as follows:

1. The step becomes active. On Entry values are written to Outputs.
2. Sequencer cycles continuously at the step body, evaluating the step condition and the jump condition at every scan cycle.
3. When the step condition is met, the On Exit output values are written if the **Step Write OnExit** is True, and the Sequencer advances to the next step.
4. When the jump condition is met, the On Exit output values are written if **Jump Write OnExit** is True, and the

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Sequencer moves the active step to the specified step.

Tip: Jumps can be forward, backward, or to the current step. If the current step is the target of the jump, the on entry outputs are written once just like at any other step destination.

You can use Jump transitions for:

- Conditional Program Flow
- Error Handling
- Monitoring

Triggers

In the step program, triggers cause the step or jump condition of the active step to be met and the step program processing to continue. Triggers are linked to Aliases that reference object attributes.

Note: Triggers are usually linked to Boolean attributes, but can also use any other numeric data type (integer, float, double). The trigger value is coerced to True or False.

In the simplest While True case, when a Trigger (Boolean attribute) is true, the condition the Condition evaluates to true, the On Exit Output are written and the step is ready to transition the next scan. The **Aliases** panel maps attributes to Aliases. The Aliases are mapped to steps via values to be written at that step.

Step and Jump Conditions are triggered by referenced Boolean attributes in different ways, such as when the Boolean attribute:

- is true: **While True**
- becomes true while the step is active: **On True**
- is false: **While False**
- becomes false while the step is active: **On False**
- changes from true to false or vice-versa: **Data Change**

A condition can also be

- never fulfilled: **Always False**
- always fulfilled: **Always Tru**e

without using any Aliases.

Timers

Timers can either be used on their own or in combination with triggers to fulfill a certain processing requirement.

Simple timer

A Simple timer is started when the step becomes active. When the timer elapses, the condition is set to true.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Cyclic timers

Cyclic timers are pulse timers that evaluate to true for one scan at a certain time each month, each week, each day, each hour or each minute.

Cyclic timers do not retain the fired state. If a cyclic timer event did not lead to a firing condition due to combinations with triggers, the cyclic timer enters the non-firing state until the next cyclic timer event occurs.

Combined triggers and timers

You can combine a trigger and timer to form a combined condition. This is done by using a logical operators (AND, OR) or a special operator called DELAY.

Delay timer

The step or jump condition is fulfilled if the delay timer has elapsed. Delay timers can either be:

- Delay simple timer - is started when the trigger condition is met. Once started, it runs down unconditionally. When the timer elapses, the condition is set to true.
- Retentive - allows summing up the times a trigger condition is true until it reaches the specified interval.
- Non-retentive - tests for a minimum time a trigger condition needs to be continuously true before triggering.

Always true and always false conditions

The **always true** condition instructs Sequencer to always execute the step, or in the case of a jump condition, to enforce a jump to a specified step.

The **always false** condition instructs Sequencer to ignore the step execution, or to never jump, in the case of a jump condition.

Write on exit

For the step condition and the jump condition there is a **write on exit** option. You can use the **write on exit** option to instruct the Sequencer to write the outputs to the Aliases either when the step or jump condition is fulfilled.

By default the **write on exit** option is set for the step condition, and not set for THE JUMP CONDITION.

Jump to

The **Jump to** parameter of any step specifies the step name where Sequencer jumps to if the jump condition is fulfilled and the step condition is not fulfilled.

Steps and aliases naming conventions

Steps and aliases follow the same naming conventions:

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

- They can only consist of alphanumeric characters, underscore ("_") and period (".").
- They can only start with an alphanumeric character and an underscore ("_").
- They must include at least one alphabet character.
- They can be up to 32 characters in length.
- They are case-insensitive (for example it is not possible to create the aliases "Alias1" and "aLiAs1" in the same alias definition list).
- Capitalization is allowed and maintained (for example it is possible to rename "Step1" to "sTeP1" and the change persists)
- Alias names cannot be true or false.

Limitations

A Sequencer object has the following limitations:

- maximum of 1000 steps
- maximum of 1000 Aliases
- maximum of 250 outputs (OnEntry and OnExit) per step
- maximum of 10,000 outputs (OnEntry and OnExit) across all steps

Define steps

This section shows you how to configure the steps of your Sequencer step program.

Using the step program panel, you can

- add steps to the step program at any position
- configure or modify steps in the step program, such as the conditions or aliases
- change the order of steps in the step program
- validate steps in the step program for correct syntax and structure
- delete steps from the step program.

Add steps

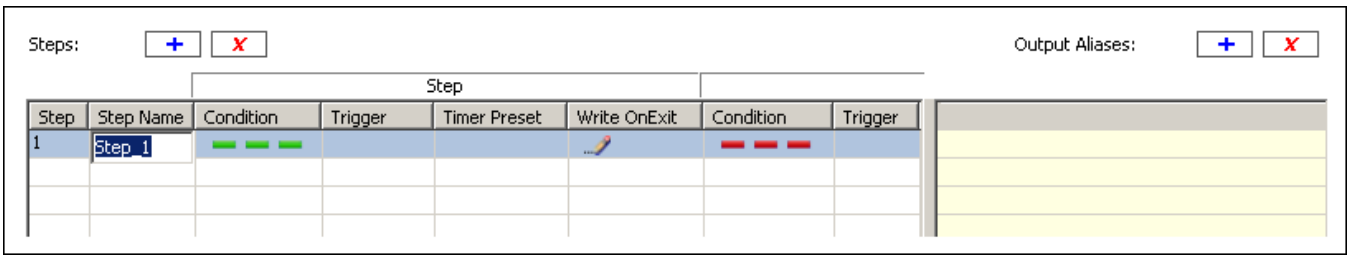
You can either add steps by appending them to the end of the step program or by inserting them at any given location in the step program.

Note: You can create up to 1000 steps in a single Sequencer object.

To add steps to a step program

1. Click the **Step Program** tab. The **Step Program** panel appears.
2. Click the **+** button next to **Steps**. A template line is added to the step program.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.



Configure steps

Configuring a step tells the step program what to do when this step becomes active. You can configure the following:

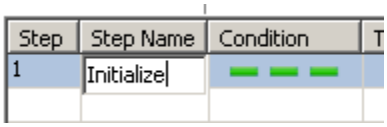
Step name	Arbitrary name you can use for easier identification.
Step section	
Condition	The combination of a trigger and/or timer required for the step to be executed. The condition is evaluated at each scan cycle until the condition is fulfilled, or any other action is taken.
Trigger	The trigger that is associated with the condition. The trigger is represented by an Alias which is defined in the Aliases panel.
Timer Preset	The time setting that is associated with the step condition. This can either be an actual time or an offset.
Write On Exit	Determines whether the step program writes the On Exit values to the outputs when the step condition is met.
Jump Section	
Condition	The combination of a trigger and/or timer required for Sequencer to jump to a different step. The jump condition is evaluated at each scan cycle. It causes Sequencer to jump, if the jump condition is met.
Trigger	The trigger that is associated with the condition. The trigger is represented by an Alias which is defined in the Aliases panel.
Timer Preset	The time setting that is associated with the jump condition. This can either be an actual time or an offset.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Write On Exit	Determines whether the step program writes the On Exit values to the outputs when the jump condition is met.
Jump To	The name of the step Sequencer jumps to, if the jump condition is met.

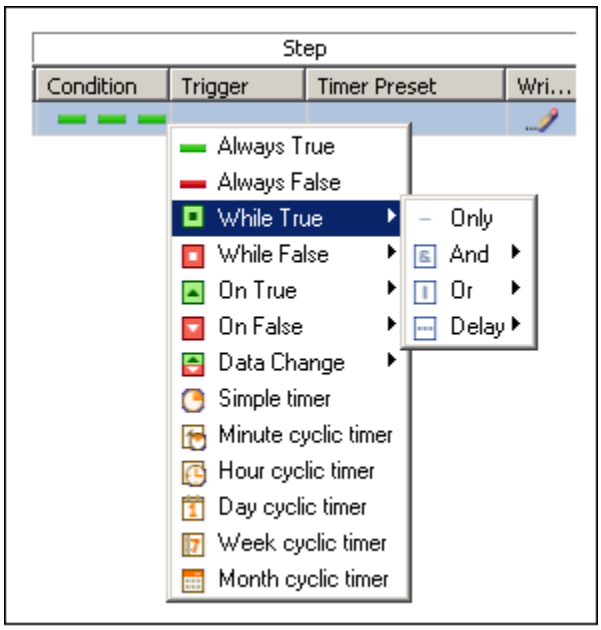
To configure or modify a step name

1. Click the **Step Program** tab. The **Step Program** panel appears.
2. Locate the step to configure or change.
3. Click the step name cell of that step. Enter a new step name and press Enter.



To configure or modify the step or jump condition

1. Click the **Step Program** tab. The **Step Program** panel appears.
2. Locate the step to configure or modify the condition.
3. Click the Condition cell of that step. Select the trigger and/or timer.



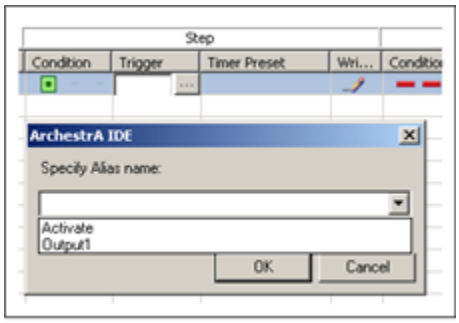
For more information about the types of triggers and timers, see Define step and jump conditions.

To assign a step or jump trigger to a condition

1. Click the **Step Program** tab. The **Step Program** panel appears.
2. Locate the step to configure or modify the condition.
3. In the Trigger cell of either the Step section or the Jump section, enter the name of the Alias or select it from

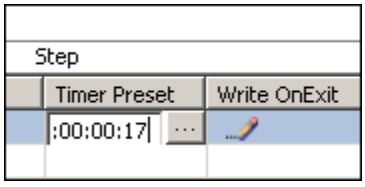
© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

the list.



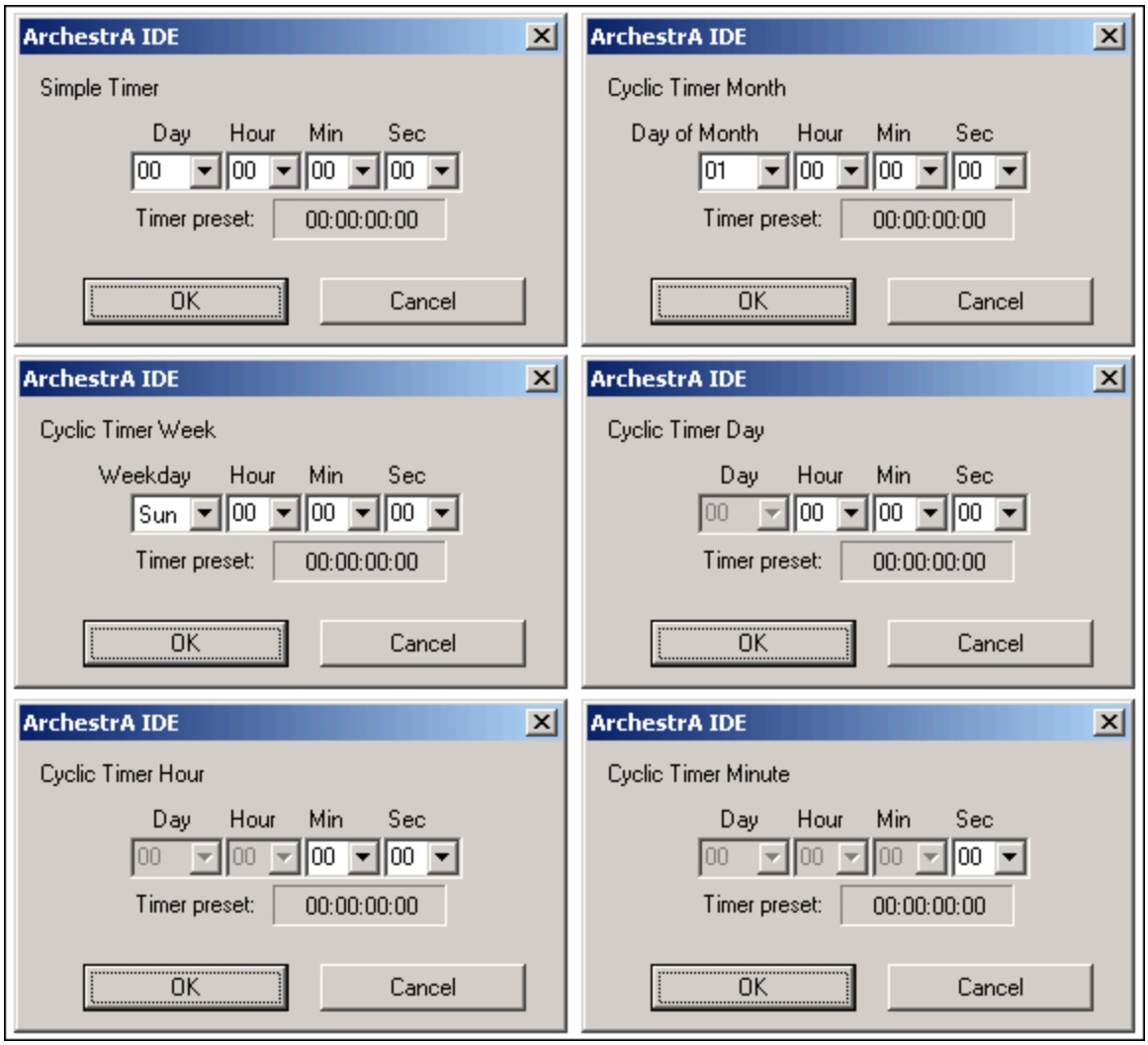
To configure or modify the Step Timer Preset or Jump Timer Preset

1. Click the **Step Program** tab. The **Step Program** panel appears.
2. Locate the step to configure or modify the timer preset.
3. Click on the **Step Timer Preset** or **Jump Timer Preset** cell of that step.



4. Click the **Browse** button. Depending on the type of timer you selected in the timer condition, one of the following IDE dialog boxes appears:
   - Simple Timer
   - Cyclic Timer Month
   - Cyclic Timer Week
   - Cyclic Timer Day
   - Cyclic Timer Hour
   - Cyclic Timer Minute

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.



Select the time to set as a timer preset in the **Simple Timer** or **Cyclic Timer** dialog box. You can also manually enter the time. The format is: dd.hh.mm.ss (dd=days, hh=hours, mm=minutes, ss=seconds).

Click **OK**.

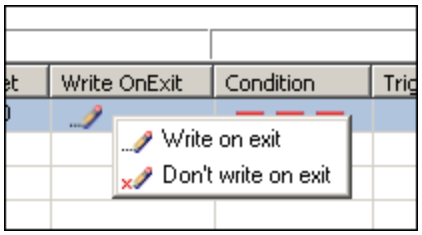
5. When you are done, press **Enter**.

Note:
- In the case of Cyclic Timer Month, the time setting is a time offset from the beginning of the month. For example: A cyclic timer setting for the 12th day of each month at 2:35:07 PM would be 11:14:35:07.
- Sunday is considered the first day of the week.

To configure or modify whether On Exit values are written when the step is executed or a jump is initiated

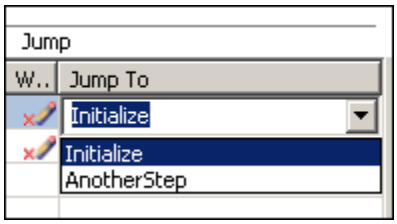
1. Click the **Step Program** tab. The **Step Program** panel appears.
2. Locate the step to configure or modify the **Write OnExit** property.
3. Click on the icon. From the list, select either **Write on Exit** or **Don't write on Exit**.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.



To configure the location where Sequencer jumps to

1. Click the **Step Program** tab. The **Step Program** panel appears.
2. Locate the step to configure or modify the jump location.
3. Click on the **Jump To** cell for that step. Select a step name from the list. You can also type in the name of the step Sequencer jumps to.



Rename a step

You can rename your steps at any time. Jump references to the renamed step are automatically updated by the step program editor.

To rename a step

1. Click the **Step Program** tab. The **Step Program** panel appears.
2. Locate the step to change. Click the step name cell of that step.
3. Enter a new step name and press Enter.

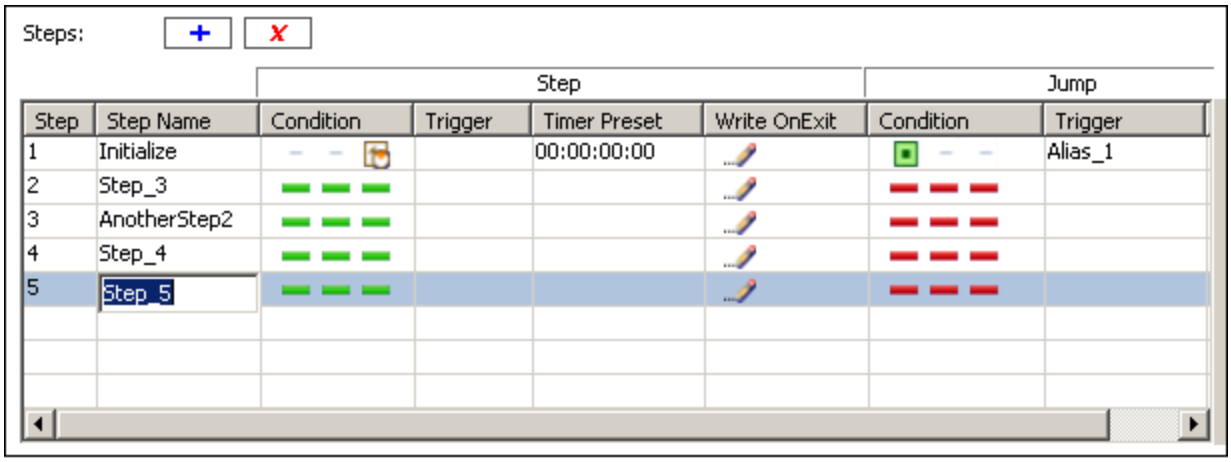
Append steps

You can add additional steps to your step program at the end of the step program. Any step can be moved to any position within the step program by dragging and dropping.

To add additional steps to a Step Program

1. Click the **Step Program** tab. The **Step Program** panel appears.
2. Click the **Add** + button next to **Steps**. An additional template line is added to the step program.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

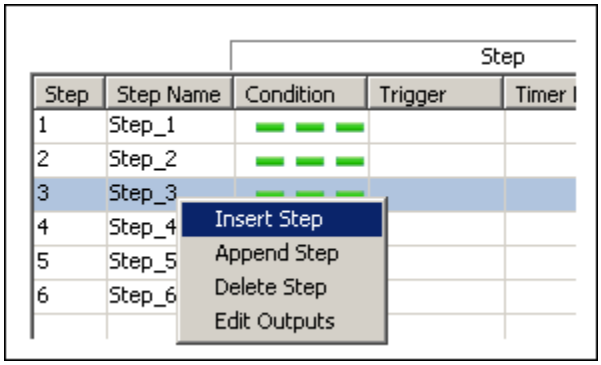


Insert a step

You can add additional steps to your step program at any position in the step program. You can drag any step to a new position.

To insert steps in a Step Program

1. Click the **Step Program** tab. The **Step Program** panel appears.
2. Right-click on the line in the step program above where you want to insert an additional step. A context menu appears.



3. Click **Insert Step** on the context menu. A template line is added to the step program at the selected position and the other steps in the step program are re-arranged accordingly.

Reorder steps

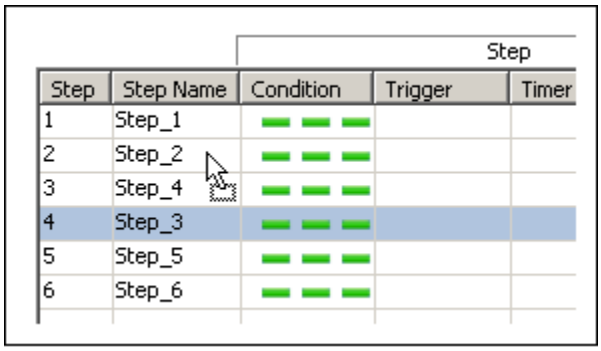
The position of the steps in the step program is crucial to the flow of the step program logic. After you create or update a step program, some steps may not be in the correct position. You can change the position of steps in the step program with the Sequencer object editor.

To change the position of steps

1. Click the **Step Program** tab. The **Step Program** panel appears.
2. Locate the step to move.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

3. Click on the Step Number cell of that step. Drag it to its destination line.



Delete steps

You can delete single steps either by the delete icon or by the context menu. You can also delete the whole step program at once.

To delete single steps from your step program

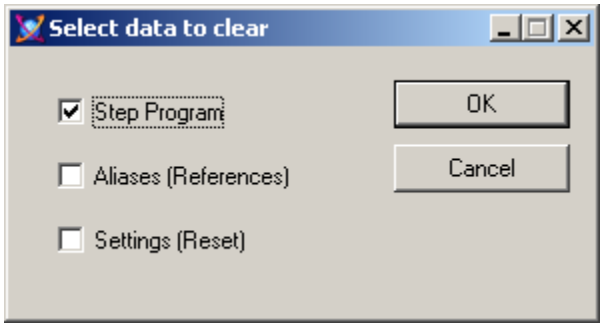
1. Click the **Step Program** tab. The **Step Program** panel appears.
2. Select the step to delete by clicking on the Step Number cell. Click the **Delete** button.

   You are asked to confirm this action as no undo is available.

   Caution: Before you start, make sure you correctly select the steps you want to delete. After they are deleted, you cannot undelete them.

To delete all steps from a step program

1. Click the **Step Program** tab. The **Step Program** panel appears.
2. Click the **Clear** button at the top of the panel. The **Select data to clear** dialog box appears.



3. Select the **Step Program** check box and click **OK**.

   You are asked to confirm this action as no undo is available.

Define minimal steps

You can define minimal steps to act as placeholders for steps that are to be included in the future. In programming these steps are also known as No Operation steps (NOP). These minimal steps take exactly two scan cycles to execute.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

To define a minimal step

- Create a new step and make sure that
  - the condition is set to Always True
  - there are no values in the output matrix that belong to this step

Note: You can configure the Sequencer to skip No Operation steps by configuring an Always True jump condition on the step before the first No Operation step. The jump destination can be the step just after the last No Operation step.

Define the initial and final steps

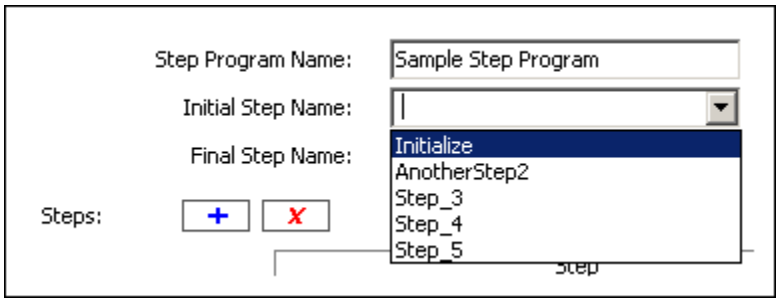
You can instruct Sequencer to begin the program at a specified step, the **Initial Step**. You can also instruct Sequencer to finish processing after a specified step, the **Final Step**.

When you deploy the Sequencer object or when the **ExecutionStateCmd** is set to **Reset**, Sequencer starts at the Initial step and runs through the remaining steps. If the Final Step is not specified, it runs through Step n and continues at Step 1.

Note: You can change the initial step and final step at run time by writing to the attributes PrgStepInitialCmd and PrgStepFinalCmd. For more information, see Set initial and final steps.

To configure the Step Program to start at a specific step

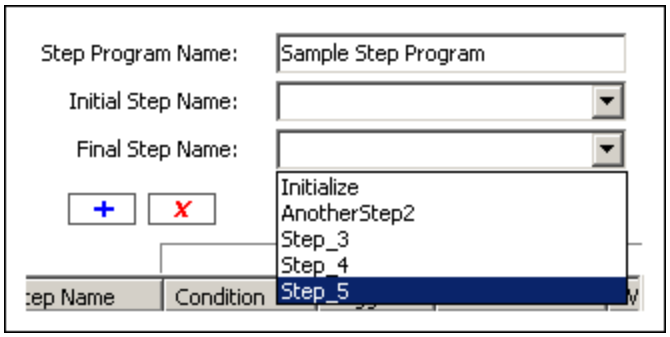
1. Click the **Step Program** tab. The **Step Program** panel appears.
2. From the **Initial Step Name** list, select the step name where you want to start the processing or enter the name manually.



To configure the Step Program to stop after a specific step

1. Click the **Step Program** tab. The **Step Program** panel appears.
2. From the **Final Step Name** list, select the step name you want the processing to stop after (or enter the name manually).

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.



Name the step program

You can give your step program a meaningful name to help you identify it. The Step Program name does not have any functional significance.

To name the Step Program

1. Click the **Step Program** tab. The **Step Program** panel appears.
2. Enter a name for your Step Program in the **Step Program Name** text field.



Note: For more information, see the section on Steps and Aliases Naming Conventions as it also applies to the Step Program Name.

Validate the step program

Step validation helps you track down errors by checking the syntax of your step program and reporting any missing settings. The results of the validation appear in the **Sequencer Program Validation Output** area at the bottom of the Sequencer Editor panel.

Common errors are:

- If a Jump Trigger or Jump Timer Mode is assigned to a step without a Jump Destination Step being assigned
- If transitions that are configured as Retentive or Non-Retentive Delay do not have Triggers assigned
- If "On True", "On False" and "Data Change" Trigger Types are used for Non-Retentive or Retentive Timer modes
- If Jump Trigger or Jump Timer Mode is selected for each step and Jump Step Name is blank

Common warnings are:

- Input and Output references are empty.
- Input and Outputs references configured as "---.---"

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

To validate the step program

1. Click the **Step Program** tab. The **Step Program** panel appears.
2. Click the **Validate** button.
3. Check for possible errors and warnings in the Sequencer Program Validation Output area at the bottom of the **Sequencer Editor** panel.



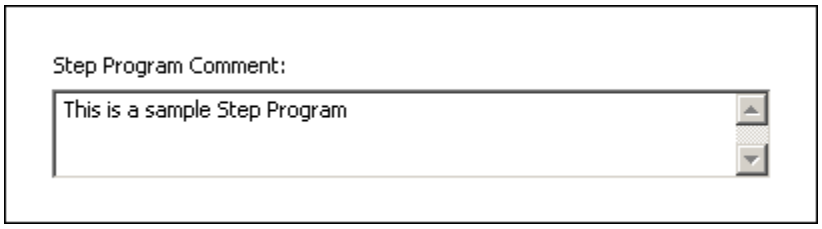
Caution: If you update the attributes PrgStepProgram and PrgAliasConfig with GRAccess, invalid entries, such as duplicate or invalid step names are not detected by the Sequencer object. Make sure you only pass valid Sequencer configuration to these attributes when using GRAccess.

Add comments to Sequencer objects

You can give your step program a meaningful comment to help you describe it. The Step Program comment does not have any functional significance.

To add or modify a comment of a Step Program

1. Click the **Step Program** tab. The **Step Program** panel appears.
2. In the **Comment** box, enter a comment.



Lock the step program

You can lock your step program so that

- the step program cannot be edited in derived instances and templates.
- changes are propagated from templates to instances.
- users cannot change the step program configuration at run time.

The following settings are locked if the step program is locked:

- Step Program Name
- Initial Step Name
- Final Step Name

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

- Step Program Comment
- Configuration of all steps
- Output values of all steps
- Importing Step Program
- Clearing Step Program
- Removing and renaming of Aliases that are used in the step program, even if the Alias configuration is not locked.

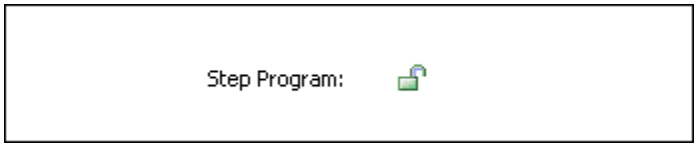
To lock the step program

1. Click the **Step Program** tab. The **Step Program** panel appears.
2. Click the Lock icon near the Step Program entry so that it is locked.
   The step program for any derived templates or instances from the Sequencer object template are locked. Any attributes related to the step program cannot be changed.



To unlock the step program

1. Click the **Step Program** tab. The **Step Program** panel appears.
2. Click the Lock icon near the Step Program entry so that it is unlocked.



Note: When either the Step Program or the Alias configuration is locked, certain constraints apply. When the Step Program template is locked, an Alias referenced by the Step Program cannot be deleted or renamed and importing an Alias configuration that does not include all of the Aliases referenced by the Step Program will fail.

Caution: When the Step Program is locked, at run time, attempts to change it by the Selected.* attributes or the PrgSeqConfigCmd attribute are not allowed.

Define step and jump conditions

This section shows you how to configure conditions for your Sequencer step program.

Define steps with trigger-based conditions

You can use triggers to control the processing of each step. Triggers are associated with Aliases that reference attributes.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

To configure a step with a trigger

1. Click the **Step Program** tab. The Step Program panel appears.
2. Locate the step for which you want to change the step condition or jump condition.
3. Click the **Condition** cell or Jump Condition cell of that step. From the menu, select either **While True**, **While False**, **On True**, **On False** or **Data Change**.

	While True
	While False
	On True
	On False
	Data Change

4. Click the **Trigger** cell to the right of the step condition cell or jump condition cell.
5. From the list, select the Alias you want to use as a trigger or enter and define a new Alias.

Define steps with timers

You can configure a step with a timer that only delays step execution. This timer is called a Simple Timer. You can use the Simple Timer to make Sequencer wait a certain time period before writing values to the On Exit outputs.

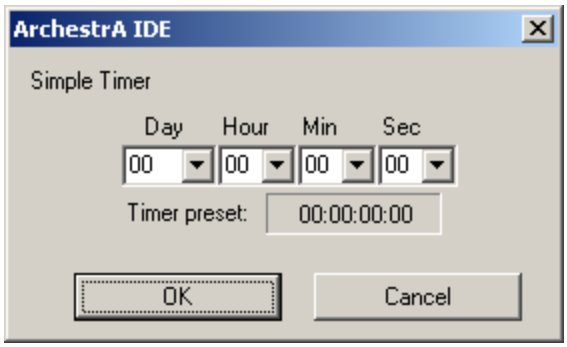
Tip: In the **Timer Preset** cell, you can enter the time delay in dd:hh:mm:ss (dd=days, hh=hours, mm=minutes, ss=seconds).

Note: The time format in the Sequencer editor is dd:hh:mm:ss does not correspond to the ElapsedTime format of the equivalent attribute (such as seen in Object Viewer), which is "d hh:mm:ss".

To configure steps with a simple timer

1. Click the **Step Program** tab. The **Step Program** panel appears.
2. Locate the step for which you want to change the step condition or jump condition.
3. Click the **Condition** cell in the Step section or Jump section for that step and select **Simple Timer** .
4. Click the **Timer Preset** cell to the right of the condition cell.
5. Click the **Browse** button. The **Simple Timer** dialog box appears.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.



6. Configure the timer and click **OK**.

Define steps with a cyclic timer

You can configure a step condition or jump condition to be fulfilled on a periodic basis. For example, you can specify the 3rd day, at 10:12:40 each month or 42 seconds after every minute. These timers are called **cyclic timers**.

Timers can also be combined with Boolean attributes. For more information, see Define trigger and timer combinations.

There are 5 different cyclic pulse timers.

	Minute-based Timer	On for one scan once a minute at the top of the minute plus the preset (xx:xx:xx:00)
	Hourly Timer	On for one scan once an hour at the top of the hour plus the preset (xx:xx:00:00)
	Daily Timer	On for one scan once a day at midnight plus the preset (xx:00:00:00)
	Weekly Timers	On for one scan once a week on Sunday at midnight plus the preset (00:00:00:00)
	Monthly Timers	On for one scan once a month on the first day of the month at midnight plus the preset (00:00:00:00)

Note:
- The time format in the Sequencer editor is dd:hh:mm:ss does not correspond to the ElapsedTime format of the equivalent attribute (such as seen in Object Viewer), which is "d hh:mm:ss".
- If the Day, Week or Month Cyclic Timer is configured to have an offset of 1 hour (timer event occurs at 1am) and if the step is entered between 1 AM and 2 AM on the day of the fallback daylight saving time schedule, then when the time rolls back from 2 AM to 1 AM, the timer condition is not fired at 1 AM. To work around this, configure the Cyclic Timer to have an offset of 1 hour and 1 second.

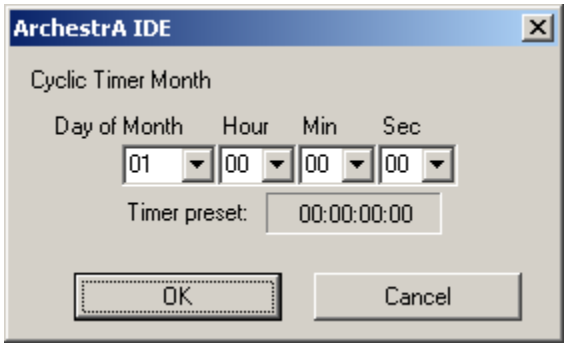
© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Define steps with a monthly timer

You can use monthly timers to fulfill a step condition or jump condition once a month. You can specify at what time precisely you want the step condition or jump condition to be fulfilled.

To configure a step with a monthly timer

1. Click the **Step Program** tab. The **Step Program** panel appears.
2. Locate the step for which you want to change the step condition or jump condition.
3. Click the Condition cell or Jump Condition cell of that step and select **Cyclic Timer Month**.
4. Click on the **Step Timer Preset** or **Jump Timer Preset** cell near the condition cell.
5. Click the **Browse** button. The **Cyclic Timer Month** dialog box appears.



6. Select the activation time from the pulldown menus and click **OK**.

Note: In the **Timer Preset** field, the value for **Day** is calculated as day of month - 1. For example: The 5th day of the month appears as 04:00:00:00 in the Timer Preset Field.

Important: Specifying an offset of days greater than or equal to the number of days in the current month causes the event to fire on the last day of the month. However, if the step is entered on the last day of the month and exceeds the timer preset, the value does not fire in this month. For example: An activation time of 30:5:15:10 entered on February 28 at 6:00AM (non leap year) does not fire in February because it exceeds the preset time of 5:15:10. In this scenario, the event triggers on March 31 at 5:15:10.

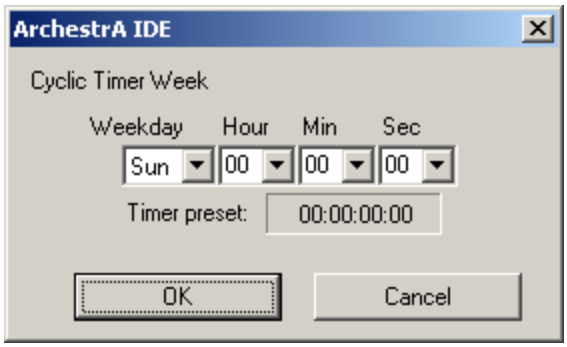
Define steps with a weekly timer

You can use weekly timers to fulfill a step condition or jump condition once a week. You can specify at what time precisely you want the step condition or jump condition to be fulfilled.

To configure a step with a weekly timer

1. Click the **Step Program** tab. The **Step Program** panel appears.
2. Locate the step for which you want to change the step condition or jump condition.
3. Click on the Condition cell or Jump Condition cell of that step and select **Cyclic Timer Week**.
4. Click on the **Step Timer Preset** or **Jump Timer Preset** cell near the condition cell.
5. Click on the ellipsis. The **Cyclic Timer Week** dialog box appears.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.



6. Select the activation time from the pulldown menus and click **OK**.

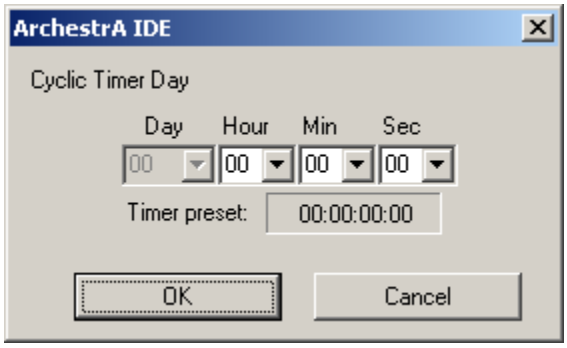
Tip: In the **Timer Preset** cell, you can enter the activation time in DD:hh:mm:ss (DD=Day of the week [00=Sunday, 01=Monday, 02=Tuesday, 03=Wednesday, 04=Thursday, 05=Friday, 06=Saturday], hh=hours, mm=minutes, ss=seconds).

Define steps with a daily timer

You can use daily timers to fulfill a step condition or jump condition once a day. You can specify at what time precisely you want the step condition or jump condition to be fulfilled.

To configure a step with a daily timer

1. Click the **Step Program** tab. The **Step Program** panel appears.
2. Locate the step for which you want to change the step condition or jump condition.
3. Click on the Condition cell or Jump Condition cell of that step and select **Cyclic Timer Day**.
4. Click on the **Step Timer Preset** or **Jump Timer Preset** cell near the condition cell.
5. Click on the ellipsis. The **Cyclic Timer Day** dialog box appears.



6. Enter the activation time and click **OK**.

Tip: In the **Timer Preset** cell, you can enter the activation time in xx:hh:mm:ss (xx=ignored, hh=hours, mm=minutes, ss=seconds).

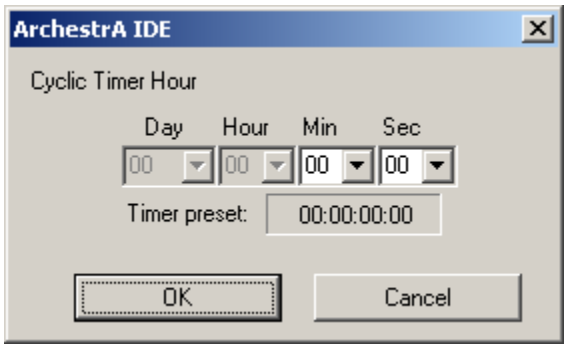
Define steps with an hourly timer

You can use hourly timers to fulfill a step condition or jump condition once an hour. You can specify at what time precisely you want the step condition or jump condition to be fulfilled.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

To configure a step with a hourly timer

1. Click the **Step Program** tab. The **Step Program** panel appears.
2. Locate the step for which you want to change the step condition or jump condition.
3. Click on the **Condition** cell or **Jump Condition** cell of that step and select **Cyclic Timer Hour**.
4. Click on the **Step Timer Preset** or **Jump Timer Preset** cell near the condition cell.
5. Click on the ellipsis. The **Cyclic Timer Hour** dialog box appears.



6. Enter the activation time and click **OK**.

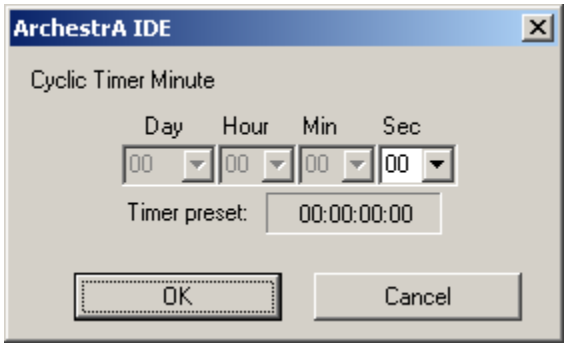
Tip: In the **Timer Preset** cell, you can enter the activation time in xx:xx:mm:ss (xx=ignored, mm=minutes, ss=seconds).

Define steps with a minute-based timer

You can use minute-based timers to fulfill a step condition or jump condition once a minute. You can specify at what time precisely you want the step condition or jump condition to be fulfilled.

To configure a step with a minute-based timer

1. Click the **Step Program** tab. The **Step Program** panel appears.
2. Locate the step for which you want to change the step condition or jump condition.
3. Click on the **Condition** cell or **Jump Condition** cell of that step and select **Cyclic Timer Minute**.
4. Click on the **Step Timer Preset** or **Jump Timer Preset** cell near the condition cell.
5. Click on the ellipsis. The **Cyclic Timer Minute** dialog box appears.



6. Enter the activation time and click **OK**.

Tip: In the **Timer Preset** cell, you can enter the activation time in xx:xx:xx:ss (xx=ignored, ss=seconds).

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Define steps with always true or always false conditions

You can configure a step so that its step condition or jump condition is always or is never true.

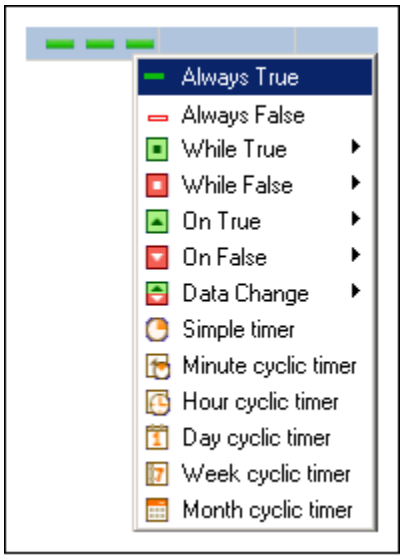
This can be done with the following two conditions:

	Always True	The condition always evaluates to true.
	Always False	The condition always evaluates to false.

Note: If both the Step condition and the Jump condition are configured as Always True, then the Step transition occurs since the Step takes precedence over the Jump.
On the other hand, if the Step condition and the Jump condition are configured as Always False, then the Sequencer remains in the active step until an Advance ExecutionStateCmd or an StepNumCmd or StepNameCmd is issued.

To configure a step or jump to always execute

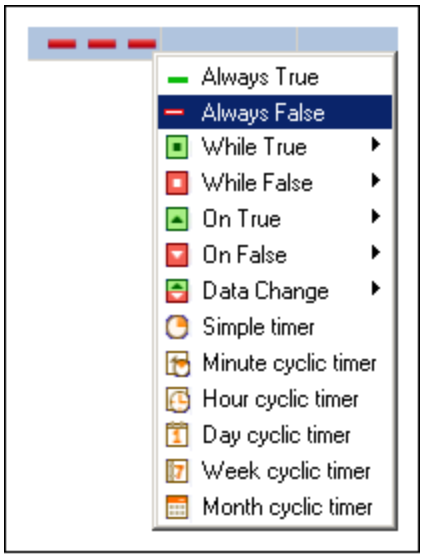
1. Click the **Step Program** tab. The **Step Program** panel appears.
2. Locate the step for which you want to change the step condition or jump condition.
3. Click the **Condition** cell or **Jump Condition** cell of that step and select **Always True**.



To configure a step or jump to never execute

1. Click the **Step Program** tab. The **Step Program** panel appears.
2. Locate the step for which you want to change the step condition or jump condition.
3. Click the Condition cell or Jump Condition cell of that step and select **Always False**.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.



Define trigger and timer combinations

Combining triggers and timers gives you more power over the conditional functionality of a step. Besides the triggers **While True**, **While False**, **On True** and **On False** and the simple timers and cyclic timers, you can also define

- logical operator to associate triggers with timers
- retentive and non-retentive timers

Use operators to connect triggers

You can use the following operators to logically connect event-based triggers and time-based triggers:

	AND	Use this icon to logically connect an event-based trigger and a time-based trigger with the AND operator.
	OR	Use this icon to logically connect an event-based trigger and a time-based trigger with the OR operator.
	DELAY	Use this icon to logically connect an event-based trigger and a time-based trigger with the DELAY operator.

Retentive and non-retentive timers

Retentive timers count only the time the trigger condition is true. The timer pauses when the trigger condition changes to false and resumes when the trigger condition changes to true again.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Non-retentive timers count only the time the trigger condition is continuously true. If the trigger condition is no longer true, the timer is reset.

Configure steps with combined conditions

You can define steps with combinations of triggers and timers. For more information on the numerous combinations of triggers and timers and what the combinations mean, see Overview of trigger and timer combinations.

To configure steps with combined conditions

1. Click the **Step Program** tab. The **Step Program** panel appears.
2. Locate the step for which you want to change the step condition or jump condition.
3. Click the **Condition** cell or **Jump Condition** cell of that step and select the combination you want from menus.

Overview of trigger and timer combinations

The following table shows a complete list of combinations of timers and triggers.

Trigger Condition	The Trigger condition is true, when...
While True AND Simple Timer	...the trigger is true and a specified time period has passed since step activation
While True AND Minute Cyclic Timer	...the trigger is true and the specified offset (in seconds) is reached
While True AND Hour Cyclic Timer	...the trigger is true and the specified offset (in minutes:seconds) is reached
While True AND Day Cyclic Timer	...the trigger is true and the specified offset (in hours:minutes:seconds) is reached
While True AND Week Cyclic Timer	...the trigger is true and the specified offset (in days:hours:minutes:seconds) is reached

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

While True AND Month Cyclic Timer 	...the trigger is true and the specified offset (in days:hours:minutes:seconds) is reached
While True OR Simple Timer 	...the trigger is true or a specified time period has passed since step activation
While True OR Minute Cyclic Timer 	...the trigger is true or the specified offset (in seconds) is reached
While True OR Hour Cyclic Timer 	...the trigger is true or the specified offset (in minutes:seconds) is reached
While True OR Day Cyclic Timer 	...the trigger is true or the specified offset (in hours:minutes:seconds) is reached
While True OR Week Cyclic Timer 	...the trigger is true or the specified offset (in days:hours:minutes:seconds) is reached
While True OR Month Cyclic Timer 	...the trigger is true or the specified offset (in days:hours:minutes:seconds) is reached
While True DELAY Simple Timer 	...the trigger is true and a specified time period has passed after the trigger became true, regardless if the trigger becomes false again
While True DELAY Retentive Timer 	...the trigger is true and remains true for a specified time period. If the trigger becomes false, this timer is halted and continued when it becomes true.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

While True DELAY NonRetentive Timer 	...the trigger is true and remains true continuously for a specified time period. If the trigger becomes false before this time is reached, the timer is reset.
While False AND Simple Timer 	...the trigger is false and a specified time period has passed since step activation
While False AND Minute Cyclic Timer 	...the trigger is false and the specified offset (in seconds) is reached
While False AND Hour Cyclic Timer 	...the trigger is false and the specified offset (in minutes:seconds) is reached
While False AND Day Cyclic Timer 	...when the trigger is false and the specified offset (in hours:minutes:seconds) is reached
While False AND Week Cyclic Timer 	...the trigger is false and the specified offset (in days:hours:minutes:seconds) is reached
While False AND Month Cyclic Timer 	...the trigger is false and the specified offset (in days:hours:minutes:seconds) is reached
While False OR Simple Timer 	...the trigger is false or a specified time period has passed since step activation
While False OR Minute Cyclic Timer 	...the trigger is false or the specified offset (in seconds) is reached

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

While False OR Hour Cyclic Timer 	...the trigger is false or the specified offset (in minutes:seconds) is reached
While False OR Day Cyclic Timer 	...the trigger is false or the specified offset (in hours:minutes:seconds) is reached
While False OR Week Cyclic Timer 	...the trigger is false or the specified offset (in days:hours:minutes:seconds) is reached
While False OR Month Cyclic Timer 	...the trigger is false or the specified offset (in days:hours:minutes:seconds) is reached
While False DELAY Simple Timer 	...the trigger is false and a specified time period has passed after the trigger became false, regardless if the trigger becomes true
While False DELAY Retentive Timer 	...the trigger is false and remains true for a specified time period. If the trigger becomes true, this timer is halted and continued when it becomes false again.
While False DELAY NonRetentive Timer 	...the trigger is false and remains false continuously for a specified time period. If the trigger becomes true before this time is reached, the timer is reset.
On True AND Simple Timer 	...the trigger becomes true from false and a specified time period has passed since step activation
On True OR Simple Timer 	...the trigger becomes true from false or a specified time period has passed since step activation

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

On True OR Minute Cyclic Timer 	...the trigger becomes true from false or the specified offset (in seconds) is reached
On True OR Hour Cyclic Timer 	...the trigger becomes true from false or the specified offset (in minutes:seconds) is reached
On True OR Day Cyclic Timer 	...the trigger becomes true from false or the specified offset (in hours:minutes:seconds) is reached
On True OR Week Cyclic Timer 	...the trigger becomes true from false or the specified offset (in days:hours:minutes:seconds) is reached
On True OR Month Cyclic Timer 	...the trigger becomes true from false or the specified offset (in days:hours:minutes:seconds) is reached
On True DELAY Simple Timer 	...the trigger becomes true from false and a specified time period has passed since, regardless if the trigger becomes false in the meantime
On False AND Simple Timer 	...the trigger becomes false from true and a specified time period has passed since step activation
On False OR Simple Timer 	...the trigger becomes false from true or a specified time period has passed since step activation
On False OR Minute Cyclic Timer 	...the trigger becomes false from true or the specified offset (in seconds) is reached

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

On False OR Hour Cyclic Timer 	...the trigger becomes false from true or the specified offset (in minutes:seconds) is reached
On False OR Day Cyclic Timer 	...the trigger becomes false from true or the specified offset (in hours:minutes:seconds) is reached
On False OR Week Cyclic Timer 	...the trigger becomes false from true or the specified offset (in days:hours:minutes:seconds) is reached
On False OR Month Cyclic Timer 	...the trigger becomes false from true or the specified offset (in days:hours:minutes:seconds) is reached
On False DELAY Simple Timer 	...the trigger becomes false from true and a specified time period has passed since, regardless if the trigger becomes true in the meantime
Data Change AND Simple Timer 	...the trigger changes from true to false (or from false to true) and a specified time period has passed since the step became active
Data Change OR Simple Timer 	...the trigger changes from true to false (or from false to true) or a specified time period has passed since step activation
Data Change OR Minute Cyclic Timer 	...the trigger changes from true to false (or from false to true) or the specified offset (in seconds) is reached
Data Change OR Hour Cyclic Timer 	...the trigger changes from true to false (or from false to true) or the specified offset (in minutes:seconds) is reached

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Data Change OR Day Cyclic Timer 	...the trigger changes from true to false (or from false to true) or the specified offset (in hours:minutes:seconds) is reached
Data Change OR Week Cyclic Timer 	...the trigger changes from true to false (or from false to true) or the specified offset (in days:hours:minutes:seconds) is reached
Data Change OR Month Cyclic Timer 	...the trigger changes from true to false (or from false to true) or the specified offset (in days:hours:minutes:seconds) is reached
Data Change DELAY Simple Timer 	...the trigger changes from true to false (or from false to true) and a specified time period passes since, regardless if the trigger changes again

Define outputs for a step

This section shows you how to define outputs for the steps of your step program.

Create aliases

The steps in your step program use aliases to reference object attributes. Aliases are used to

- write outputs once a step is active (On Entry Write)
- trigger the step or jump condition
- write outputs after the step or jump condition is fulfilled (On Exit Write)
- specify the output value to be written when an Alias appears in the output matrix as a pass by reference value.

You must define an alias before using it to read from or write to an attribute.

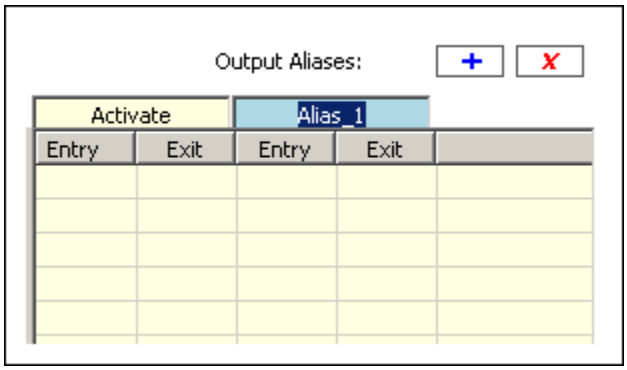
Note:
- You cannot map an Alias to an attribute from the Step Program tab. Do this on the Aliases tab. For more information, see Assign an attribute reference to an alias.
- You can create up to 1000 Aliases in a single Sequencer object.

To create an alias from the Step Program panel

1. Click the **Step Program** tab. The **Step Program** panel appears.
2. Click on the **Add** button near the Outputs entry. An additional column is added in the **Outputs** area with a

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

default name as column header.



3. Enter a name for the new Alias.

   Note: You cannot add an Alias if the Alias Configuration (PrgAliasConfig) is locked, even if the Step Program (PrgStepProgram) is unlocked.

Assign an output value to a step

You can assign output values to each step in your step program. These values are written to the attributes that are mapped to the Aliases.

Note: You can create up to 250 outputs per step and up to 10,000 outputs across all steps.

Use:

- **On Entry outputs** to write outputs as soon as the step becomes active.
- **On Exit outputs** to write outputs as soon as the step or jump condition is fulfilled.

Output values can either be literals (numerical values, True/False, date time values, elapsed time values and strings) or Aliases.

Note: Hexadecimal values are not supported.

To assign output values to Aliases in steps, you can either use:

- Output matrix
- **Edit Outputs** dialog box - is especially useful if you have many outputs as it simplifies how they are shown.

  Note: Unlike the output matrix, you cannot use the **Edit Outputs** dialog box to add or rename Aliases.

Assign a literal output value

You can either use the output matrix or the **Edit Outputs** dialog box to assign a literal output value in a step.

Literal values can be:

- numerical values, such as 3.141, 1000 or -141.22
- boolean values, such as TRUE or FALSE
- string values (enclosed by double quotes), such as "Start" or "Machine 1" or "John Smith"
- time values (enclosed by double quotes), such as "5/30/2006 8:37:43.504 PM"

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

- elapsed time values (enclosed by double quotes), such as "14:32:55.7328740"

Assign a literal output value in the output matrix

You can write literal values to On Entry or On Exit outputs for any step in the step program. These values are specified in the output matrix.

To assign a literal value to a step as output value in the output matrix

1. Click the **Step Program** tab. The **Step Program** panel appears.
2. Locate the cell you need to place the output value at, which is a combination of:
   a. the step for which you want to assign the output value
   b. the Alias which references the attribute you want to write to
   c. On Entry or On Exit, depending if you want the output value to be written when the step becomes active or when the step or jump condition is fulfilled
3. Click the located cell, enter the output value and press Enter.

Output1		Output2
Entry	Exit	Entry
1	11	123.45
		"Hello World"
		"5/16/2006 10:40:34.109 AM"
		"00:10:30.000"
		Alias_3

Important: String, date time or elapsed time data type values need to be enclosed with double quotes ("). Otherwise, they are interpreted as Aliases.

Assign a literal output value in the edit outputs dialog box

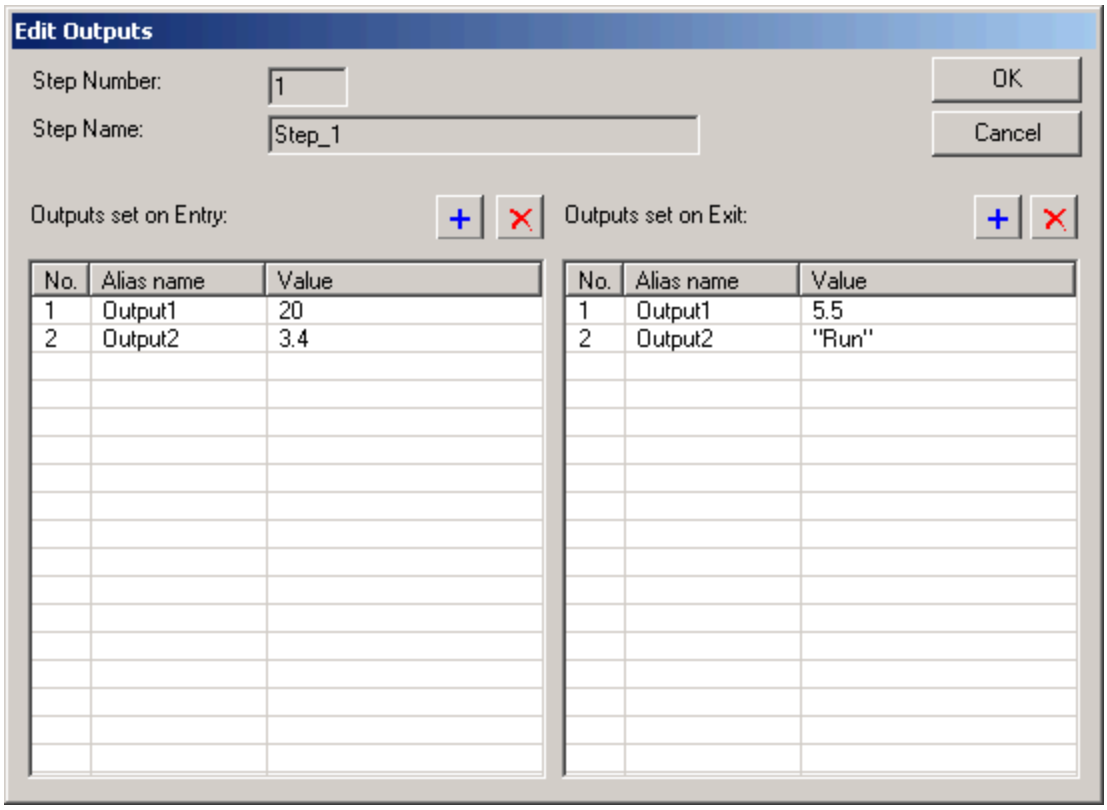
You can write literal values to On Entry or On Exit outputs for any step in the step program. These can be entered in the **Edit Outputs** dialog box.

Note: The **Edit Outputs** dialog box only lists Aliases that have output values assigned to them for the step the dialog is opened from.

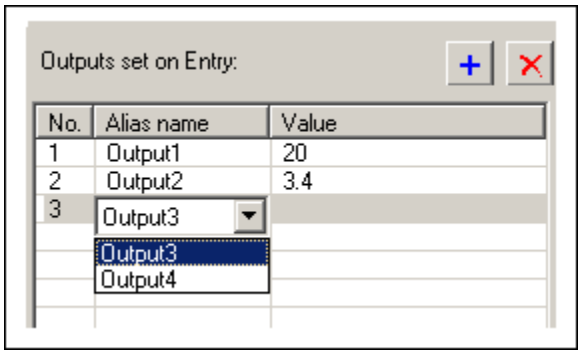
To assign a literal value to a step as output value in the Edit Outputs dialog box

1. Click the **Step Program** tab. The **Step Program** panel appears.
2. Right-click the step you want to assign output values to and select **Edit Outputs**. The **Edit Outputs** dialog box appears.

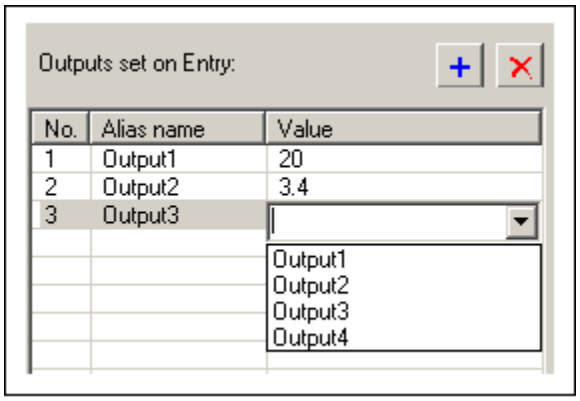
© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.



3. If the Alias you want to use does not appear in the list, click on either add (**+**) button depending if you want to assign an OnEntry or an OnExit value. Then select the Alias from the pulldown list that appears in the list.



4. Select the Alias you want to assign a value to as output and then click the value cell. A pulldown menu appears.



© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

5. Enter a value in the field and press Enter.
6. When you are done, click **OK** to close the **Edit Outputs** dialog box. All values are updated in the output matrix.

Note: When you include an Alias in the **Edit Outputs** dialog box it may appear at the beginning, middle or the end of the list. The order of Aliases in the **Edit Outputs** dialog box depends on the order of Aliases on the **Aliases** panel.

Assign an alias value as the output value

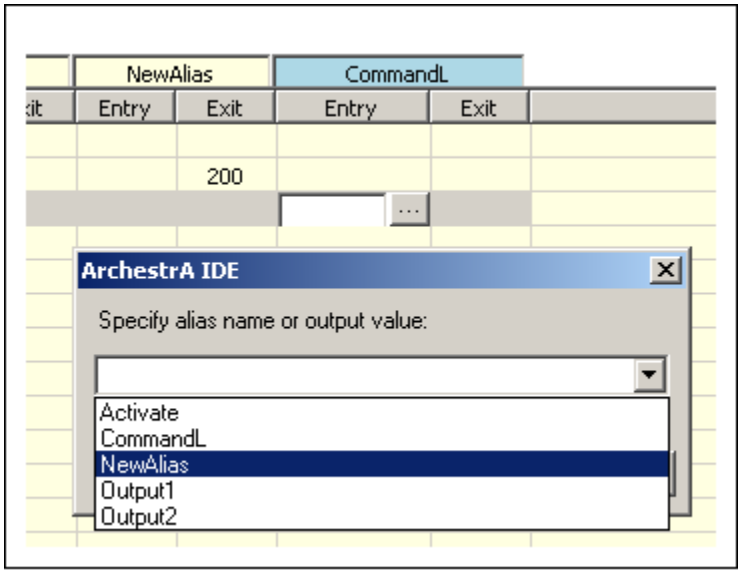
You can either use the output matrix or the **Edit Outputs** dialog box to assign the value of another Alias as output value in a step.

Assign an alias value in the output matrix

You can configure a step so that the current value of one Alias is written to another Alias when that step becomes active or the step or jump condition is fulfilled.

To assign the value of another Alias as output value

1. Click the **Step Program** tab. The **Step Program** panel appears.
2. Locate the cell you need to place the output value at, which is a combination of:
   a. the step for which you want to assign the output value
   b. the Alias which references the ArchestrA attribute you want to write to
   c. On Entry or On Exit, depending if you want the output value to be written when the step becomes active or when the step or jump condition is fulfilled
3. Click the located cell.
4. Select the Alias from the pulldown list or type the name in manually and press Enter.



5. If you enter an Alias name that does not exist, a confirmation dialog appears asking you to create the Alias.
6. Click **Create** to create the Alias.

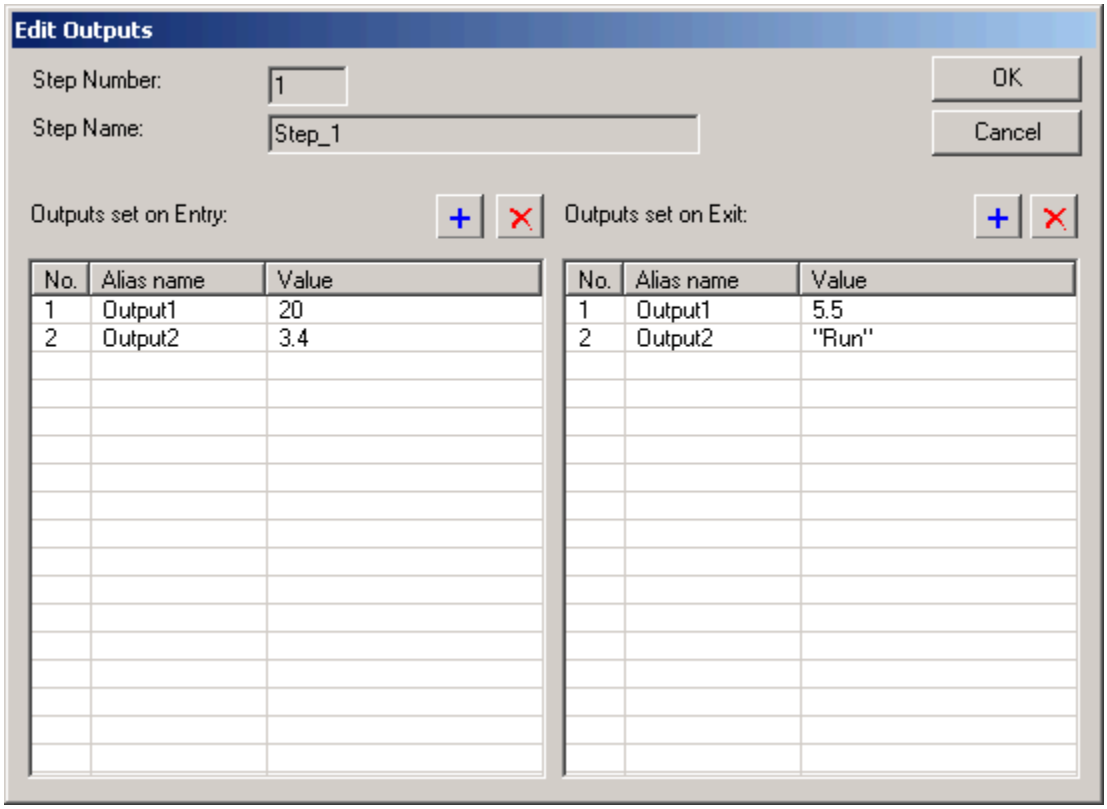
© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Assign an alias in the edit outputs dialog box

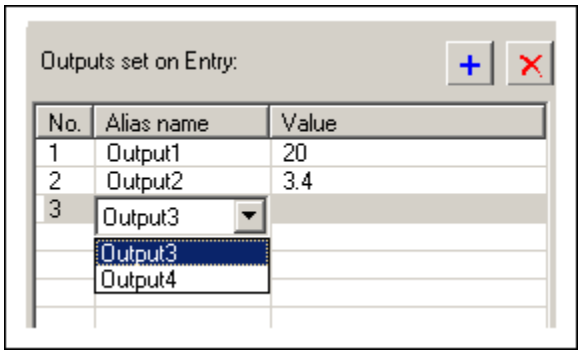
You can write literal values to On Entry or On Exit outputs for any step in the step program. These can be entered in the **Edit Outputs** dialog box.

To assign a literal value to a step as output value in the Edit Outputs dialog box

1. Click the **Step Program** tab. The **Step Program** panel appears.
2. Right-click the step you want to assign values of Aliases to and select **Edit Outputs**. The **Edit Outputs** dialog box appears.

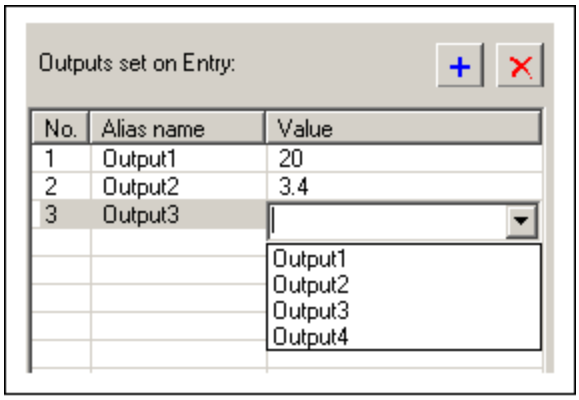


3. If the Alias you want to use does not appear in the list, click on either add (+) button depending if you want to assign an OnEntry or an OnExit Alias. Then select the Alias from the pulldown list that appears in the list.



4. Select the Alias you want to assign a value to as output and then click the value cell. A pulldown menu appears.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.



5. Select an Alias from the pulldown menu and press Enter.
6. When you are done, click **OK** to close the **Edit Outputs** dialog box. All Aliases are updated in the output matrix.

Remove an output value

You can either use the output matrix or the **Edit Outputs** dialog box to remove output values from a step.

Remove values from the output matrix

If you do not want a step to write to an Alias you can blank out the corresponding cell in the output matrix.

To remove output values from the output matrix

1. Click the **Step Program** tab. The **Step Program** panel appears.
2. Locate the cell you need to want to remove the output value from. This cell is at the intersection of:
   a. the step for which you want to remove the output value
   b. the Alias which references the ArchestrA attribute you want to remove the output value for
   c. On Entry or On Exit
3. Click the located cell, delete the contents and press Enter.

Remove values from the edit outputs dialog box

If you do not want a step to write to an Alias you can remove it from the **Edit Output** dialog box.

To remove output values from the Edit Outputs dialog box

1. Click the **Step Program** tab. The **Step Program** panel appears.
2. Right-click the step you want to assign values of Aliases to and select **Edit Outputs**. The **Edit Outputs** dialog box appears.
3. Select the Alias assigned to the value that you want to remove.
4. Click the delete (**X**) button of the corresponding list heading. The Alias and value are removed from the list.
5. Click **OK** to dismiss the **Edit Outputs** dialog box. The value is also removed from the output matrix.

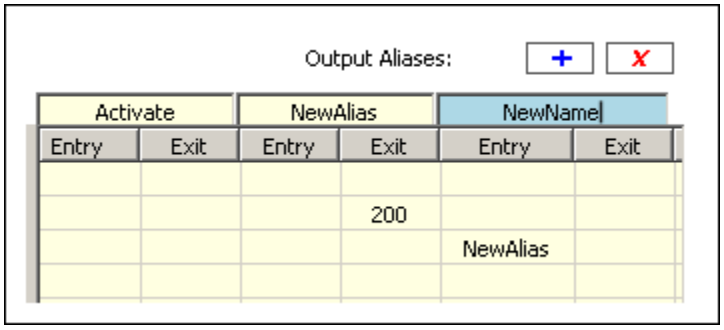
© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Rename an alias

You can rename Aliases at any time. When you rename an Alias, the Step Program Editor updates all references in the Sequencer Object.

To rename an Alias (from the Step Program panel)

1. Click the **Step Program** tab. The **Step Program** panel appears.
2. Locate the Alias from the Output Matrix and select its header.
3. Enter a new name for the Alias.
   The name of the Alias is updated by the Step Program Editor in the Output Matrix, in the **Jump To** list and on the **Aliases** panel.



Note: You cannot rename an Alias if the Alias Configuration (PrgAliasConfig) is locked, even if the Step Program (PrgStepProgram) is unlocked.

Delete an alias

You can delete an Alias either from the Step Program panel or from the Alias panel. You cannot delete an Alias if it is used in a step condition or a jump condition.

When you delete the Alias, it is removed from

- the Step Program panel (as column in the output matrix)
- the Aliases panel Alias definition list
- in pulldown menus that use Aliases (such as triggers)

Note: When you delete an Alias, you cannot undelete it. You must recreate it. If you delete an Alias that has output values associated with it, these values are deleted as well.

To delete an Alias from the Step Program panel

1. Click the **Step Program** panel. The **Step Program** panel appears.
2. Locate the Alias from the **Output Matrix** and select its header.
3. Click the **Delete** (**X**) button at the top of the Output Matrix.

The Alias is deleted from the Output Matrix, the Jump To lists and also from the Aliases panel.

Note: You cannot delete an Alias if the Alias Configuration (PrgAliasConfig) is locked, even if the Step Program (PrgStepProgram) is unlocked.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Alias definitions

This section shows you how to work with Alias definitions. The list of Alias definitions shows you which Alias is mapped to which attribute. You can:

- create an Alias
- assign an attribute reference to an Alias
- rename an Alias
- change the order of appearance and execution of Aliases
- delete an Alias
- delete attribute references of Aliases
- lock the Alias definition list
- view Alias usage, such as references and cross references

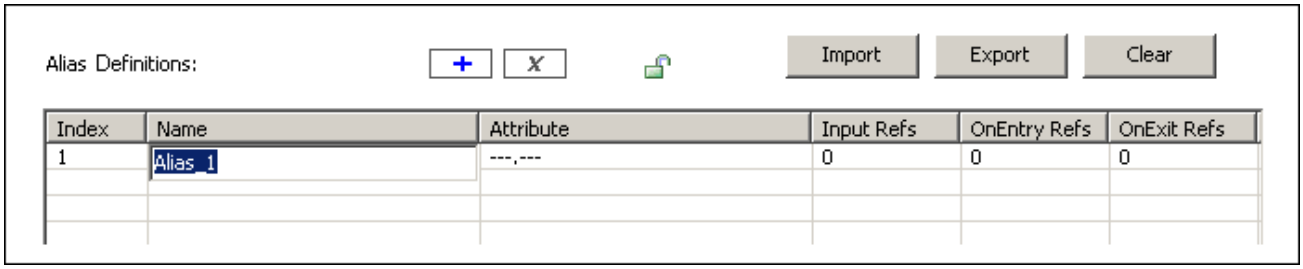
Create an alias

You can create a new Alias either from the Step Program panel when you are creating and configuring steps, or you can create it from the Aliases panel directly. When you create an Alias, it appears:

- on the **Step Program** panel as a new column in the output matrix
- on the **Aliases** panel as a new item in the Alias definition list
- in the menus where Aliases can be selected, such as Trigger columns.

To create an Alias from the Aliases panel

1. Click the **Aliases** tab. The **Aliases** panel appears.
2. Click the **Add** button next to **Alias Definitions**. A template line is added to the **Alias definition** list.



3. Enter a name for the created Alias and press Enter. The Alias is created and appears in the Alias definition list and also as a new column of the output matrix on the **Step Program** panel.

Assign an attribute reference to an alias

You can assign an attribute reference for every Alias you create. Assigning an attribute to an Alias means that

- any value associated with the Alias is written to the assigned attribute
- when the step program uses the Alias as a trigger, the value of the assigned attribute acts as trigger value

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

To assign an attribute reference to an Alias

1. Click the **Aliases** tab. The **Aliases** panel appears.
2. Locate the Alias to which you want to assign the attribute reference.
3. Click on the attribute cell of the located row.
4. Either enter an attribute reference manually and press Enter, or click the ellipsis button and use the Attribute browser to select the attribute:

Rename an alias

You can rename an Alias at any time. You can rename an Alias either from the Step Program panel or from the Aliases panel. When you rename the Alias, it is updated

- on the Step Program panel
- on the Aliases panel
- in menus that use Aliases, such as Triggers

To rename an Alias from the Aliases panel

1. Click the **Aliases** tab. The **Aliases** panel appears.
2. Locate the Alias you want to rename and click on the Name cell of that row.
3. Enter a new name and press Enter. The Alias is renamed.

Change the appearance order of aliases

You can change the order in which Aliases appear on the Aliases panel according to your own preference. Changing the order of the Aliases on the Aliases panel also changes the order of the Aliases columns in the output matrix on the Step Program panel.

To change the order of Aliases

1. Click the **Aliases** tab. The **Aliases** panel appears.
2. Locate the Alias that you want to move to a different position and click on the Index cell of that row.
3. Drag and drop the Alias to a row in the list where you want it to appear. The Alias moves there, the other Aliases are arranged accordingly.

Remove aliases

You can either delete a single Alias, all Aliases or clear their references.

Delete an alias

You can delete an Alias either from the Step Program panel or from the Aliases panel. You cannot delete an Alias if it is used in a step condition or in a jump condition.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

When you delete the Alias, it is removed from

- the Step Program panel (as a column in the output matrix)
- the Aliases panel Alias definition list
- pulldown menus that use Aliases (such as triggers and output matrix cells)
- step output values associated with the Alias are also deleted

If the Alias is referenced to an attribute, you cannot delete it.

Note: When you delete an Alias, make sure you select the correct one. After you delete an Alias, you cannot undelete it. You must recreate it.

To delete an Alias from the Aliases panel

1. Click the **Aliases** tab. The **Aliases** panel appears.
2. Locate the Alias you want to delete and select it.
3. Click the **Delete** (**X**) button at the top of the panel. The Alias is deleted from the list.
   You are asked to confirm this action as no undo is available.

Clear aliases

You can clear Alias from either the Step Program, Aliases or Settings panel. When you clear all Aliases

- all unused Aliases are deleted
- the attribute references for used Aliases are reset to "---.---"

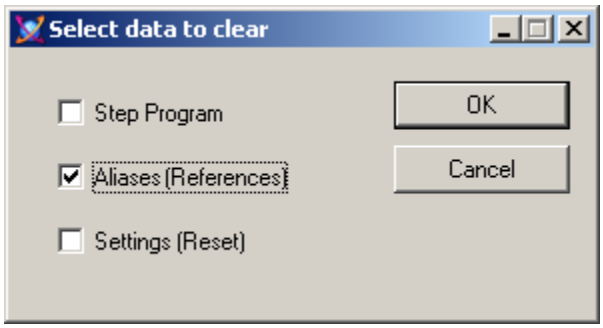
When unused Aliases are cleared, they are removed from

- the Step Program panel (as a column in the output matrix)
- the Aliases panel Alias definition list
- in pulldown menus that use Aliases (such as triggers and output matrix cells)

Note: Clearing all Aliases cannot be undone.

To clear all Aliases

1. From either the Step Program, Aliases or Settings panel, click **Clear**. The **Select data to clear** dialog box appears.



2. Make sure **Aliases (References)** is selected and click **OK**.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

You are asked to confirm this action as no undo is available.

Lock the alias definition list

You can lock the Alias definition list so that:

- the Aliases in derived instances and templates cannot be edited
- users cannot change the Alias configuration at run time
- changes are propagated from templates to instances

The following actions are not allowed if the Alias definition list is locked:

- Adding, renaming or deleting Aliases
- Assigning attribute references to Aliases
- Importing Aliases

Note: When either the Step Program or the Alias configuration is locked, certain constraints apply. When the Step Program template is locked, an Alias referenced by the Step Program cannot be deleted or renamed. Importing an Alias configuration that does not include all of the Aliases referenced by the Step Program fails.

View alias references

You can view the following usage data for each Alias:

- **Input Refs**: Shows the number of times the Alias is used in the Step Program either as a trigger or for passing output values by reference.
- **OnEntry Refs**: Shows how many steps the Alias is assigned an On Entry value.
- **OnExit Refs**: Shows in how many steps the Alias is assigned an On Exit value.

To view Alias usage

1. Click the **Aliases** tab. The **Aliases** panel appears.
2. View the 3 columns on the right hand side of the Alias list.

Index	Name	Attribute	Input Refs	OnEntry Refs	OnExit Refs
1	Activate	MyContainer.Activate	1	1	0
2	InputValve	MyContainer.InputValve	0	3	1
3	Agitator	MyContainer.Agitator	0	3	1
4	OutputValve	MyContainer.OutputValve	0	3	1
5	Contaminated	MyContainer.Contaminated	4	0	0
6	DiscardValve	MyContainer.DiscardValve	0	2	1
7	TankEmpty	MyContainer.TankEmpty	2	0	0
8	TankFull	MyContainer.TankFull	1	0	0

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

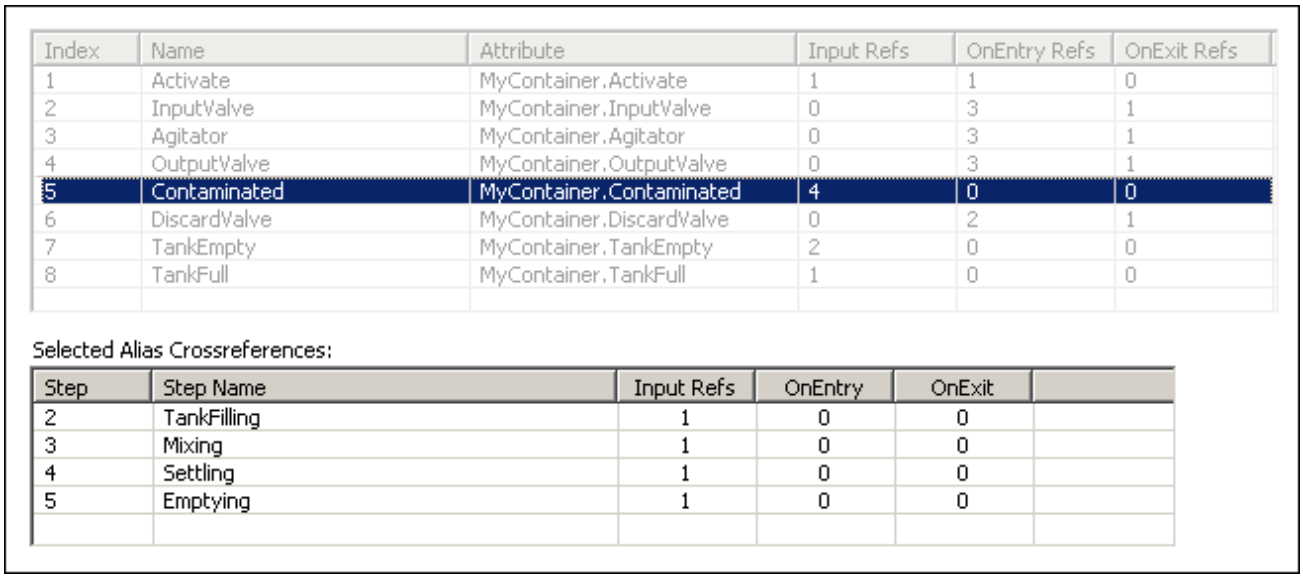
View Alias Crossreferences

You can see Crossreference usage information on an Alias such as:

- **Step**: Shows the step numbers where the Alias is used.
- **Step Name**: Shows the step name of all steps where the Alias is used.
- **Input Refs**: Shows the number of times the selected Alias is used as a condition trigger (step and jump) and referenced in the output matrix for a particular step.
- **OnEntry**: Shows 1 if the selected Alias is used in an On Entry value assignment, otherwise 0, for a particular step.
- **OnExit**: Shows 1 if the selected Alias is used in an On Exit value assignment, otherwise 0, for a particular step.

To view Alias Crossreference usage

1. Click the **Aliases** tab. The **Aliases** panel appears.
2. Select an Alias for which you want to see Crossreference usage. The usage information appears at the bottom of the panel.



Index	Name	Attribute	Input Refs	OnEntry Refs	OnExit Refs
1	Activate	MyContainer.Activate	1	1	0
2	InputValve	MyContainer.InputValve	0	3	1
3	Agitator	MyContainer.Agitator	0	3	1
4	OutputValve	MyContainer.OutputValve	0	3	1
5	Contaminated	MyContainer.Contaminated	4	0	0
6	DiscardValve	MyContainer.DiscardValve	0	2	1
7	TankEmpty	MyContainer.TankEmpty	2	0	0
8	TankFull	MyContainer.TankFull	1	0	0

Selected Alias Crossreferences:

Step	Step Name	Input Refs	OnEntry	OnExit
2	TankFilling	1	0	0
3	Mixing	1	0	0
4	Settling	1	0	0
5	Emptying	1	0	0

Define Sequencer program settings

This section shows you how to define general Sequencer Program Settings, such as

- Initialization Timeout
- Initial Command
- Alarms on errors and warnings
- Security settings

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Configure the initial command

You can control the way the step program is run after any of the following:

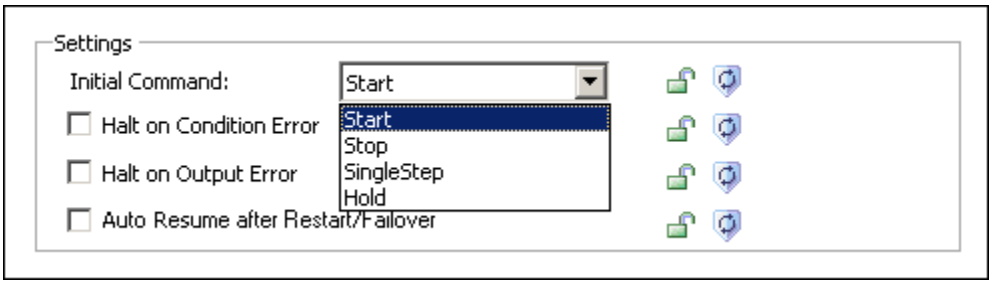
- the Sequencer object is deployed
- the attribute ExecutionStateCmd is set to **Reset**
- the attribute PrgSeqConfigCmd is set to **Load** and completes successfully
- the attribute PrgSeqConfigCmd is set to **Upload** and completes successfully

After deployment, resetting, loading or uploading, Sequencer can either

- run the program: **Start**
- not run the program: **Stop**
- run the program in RunningSingleStep mode: **Single Step**
- start the program but initially hold processing: **Hold**

Note: At run time, the Program flow can be stopped and controlled in other ways. For more information, see Set initial and final steps.

The Initial Command can be set on the Settings panel.



To configure the Step Program to start immediately after deployment, PrgSeqConfigCmd is set to Load or Unload and it is completed successfully or ExecutionStateCmd = Reset.

1. Click the **Settings** tab. The **Settings** panel appears.
2. From the **Initial Command** list, select **Start**.

To configure the Step Program to not start immediately after deployment, PrgSeqConfigCmd is set to Load or Unload and it is completed successfully or ExecutionStateCmd = Reset.

1. Click the **Settings** tab. The **Settings** panel appears.
2. From the **Initial Command** list, select **Stop**.

To configure the Step Program to run one step at a time after deployment, PrgSeqConfigCmd is set to Load or Unload and it is completed successfully or ExecutionStateCmd = Reset.

1. Click the **Settings** tab. The **Settings** panel appears.
2. From the **Initial Command** list, select **SingleStep**.

To configure the Step Program to hold step processing after deployment, PrgSeqConfigCmd is set to Load or

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Unload and it is completed successfully or ExecutionStateCmd = Reset.

1. Click the **Settings** tab. The **Settings** panel appears.
2. From the **Initial Command** list, select **Hold**.

Note: The Initial Command can be changed at run time by setting the PrgInitialCommand attribute.

Configure errors and alarms

You can specify how the Sequencer object handles irregular configuration and values at run time. You can:

- **halt program execution** if a condition trigger or Alias in the output matrix has bad quality or cannot be read, or if an Output OnEntry/OnExit Alias cannot be written or if the Sequencer Object is hanging in Initializing state.
- **report alarms** if an output or condition has a syntax error. You can log separate alarms for On Entry output errors and On Exit output errors.

Two types of errors can occur when the Sequencer object is deployed and interacts with ArchestrA attributes:

- **Condition Errors** - a condition error occurs when
  - the object failed to resolve a trigger reference
  - the quality of the trigger reference is not good
  - the data type is mismatched and cannot be handled by Message Exchange
- **Output Errors** - an output error occurs when
  - an output reference cannot be resolved or has bad quality
  - the reference of a pass by value Alias cannot be resolved or has bad quality

    Note: This affects the attributes ConditionTriggerFailure, OnEntryOutputFailure and OnExitOutputFailure.

Examples:

- configuring a **non-existent** attribute as an output in a step program causes an Output Error
- configuring a **non-numeric** attribute as an input (trigger) in a step program causes a Condition Error
- the quality of the attribute belonging to an input (trigger) or an output is bad.

Configure the initialization timeout

When the sequencer starts up, fails over, or loads or updates a new sequence program, its execution state is set to **Initializing** to allow the sequencer to connect to its input and output references.

In certain cases, for example when the Sequencer Object is put offscan during initialization, the Sequencer Object may be stuck in this state. You can configure the Initialization Timeout value to tell the object to go into a StoppedError state after a certain time period has elapsed.

To configure the Initialization Timeout

1. Click the **Settings** tab. The **Settings** panel appears.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

2. In the **Initialization Timeout** text field, enter a value for the time period in milliseconds you want to pass before the Sequencer object goes into StoppedError state after initializing.



Note: The default value for the initialization timeout is 30000 ms (30 seconds).

Halt program execution on error

The step program can be configured to report an alarm when either an Output Error or a Condition Error occurs. You can configure the Sequencer object to either continue executing or to halt program execution when such errors occur.

Note: After an error occurs and the step program is configured to "Halt Program Execution on Error", the program execution is in StoppedError state. You must reset Sequencer or set the **ExecutionStateCmd = Reset** to start processing again.

You can configure the Program Halting on the Settings panel.



To halt program execution on Condition Error:

1. Click the **Settings** tab. The **Settings** panel appears.
2. In the **Settings** area, select the check box **Halt On Condition Error**.

To halt program execution on Output Error:

1. Click the **Settings** tab. The **Settings** panel appears.
2. In the **Settings** area, select the check box **Halt on Output Error**.

To ignore Condition Errors and Output Errors:

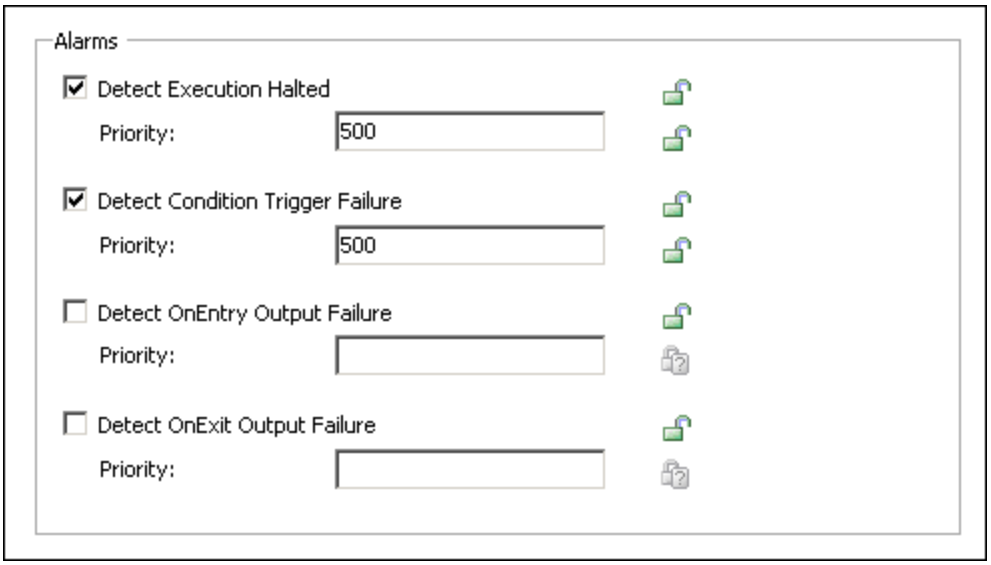
1. Click the **Settings** tab. The **Settings** panel appears.
2. In the **Settings** area, clear the check boxes **Halt On Condition Error** and **Halt on Output Error**.

Use alarms to report errors

You can configure the Sequencer Object to report an alarm if a Condition or Output error occurs, or if program execution halts for any reason.

This can be configured on the Settings panel.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.



To report Program Halting as an alarm:

1. Click the **Settings** tab. The **Settings** panel appears.
2. In the **Alarms** area, select the check box **Detect Execution Halted**.
3. Enter a valid priority in the corresponding text box. The range of valid priorities is from 0 to 999.

To report Condition Errors as alarms:

1. Click the **Settings** tab. The **Settings** panel appears.
2. In the **Alarms** area, select the check box **Detect Condition Trigger Failure**.
3. Enter a valid priority in the corresponding text box. The range of valid priorities is from 0 to 999.

To report On Entry Output Errors as alarms:

1. Click the **Settings** tab. The **Settings** panel appears.
2. In the **Alarms** area, select the check box **Detect OnEntry Output Failure**.
3. Enter a valid priority in the corresponding text box. The range of valid priorities is from 0 to 999.

To report On Exit Output Errors as alarms:

1. Click the **Settings** tab. The **Settings** panel appears.
2. In the **Alarms** area, select the check box **Detect OnExit Output Failure**.
3. Enter a valid priority in the corresponding text box. The range of valid priorities is from 0 to 999.

Configure security

You can restrict access to the following attributes of the Sequencer object on the Settings panel:

- **Program Execution** - prevent the user from changing the step program execution
- **Step Setting** - prevent the user from forcing a step jump or from setting the initial step and final step

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

- **Sequencer Configuration** - prevent the user from changing the Sequencer configuration.
- **Output Array Access** - prevent the user from changing the output values of a selected step
- **Timer Preset Access** - prevent the user from changing the timer preset values of a selected step
- **Jump Destination Access** - prevent the user from changing the jump destination of a selected step

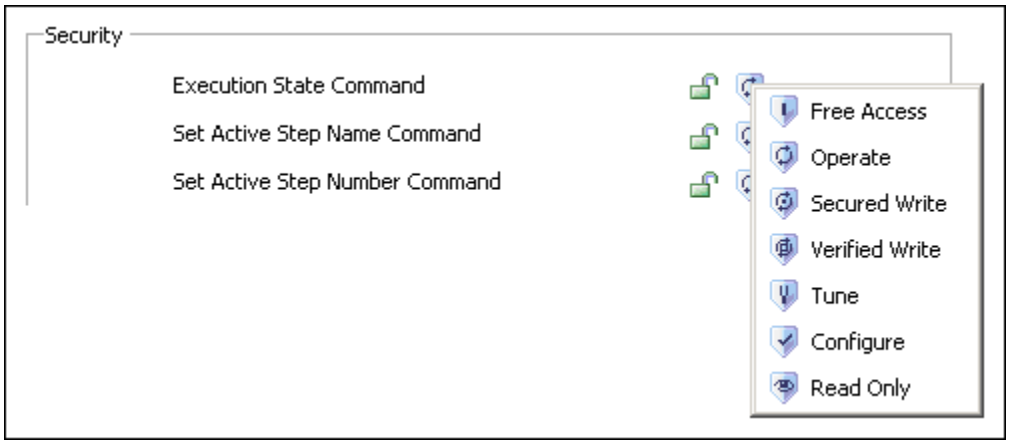
Note: You must enable security before security classification can be enforced.

Modify access rights to the program execution command

The Execution State setting lets you control the execution of the step program, such as stopping, starting, resetting, holding the step program, or running it in Single Step mode. You can prevent the user from changing this in derived templates, instances and at run time by restricting access to it.

To change access rights to the Execution State Command attribute:

1. Click the **Settings** tab. The **Settings** panel appears.
2. Select the appropriate **Execution State Command** security settings using the security classification shield. This changes your access rights to the ExecutionStateCmd attribute.



Modify access rights to step settings

You can prevent the user from instructing the step program to:

- jump to a specific step (at run time) by specifying a Step Number or a Step Name.
- change the Initial Step setting (at run time and in derived templates and instances)
- change the Final Step setting (at run time and in derived templates and instances)

Modify access rights to the active step command settings

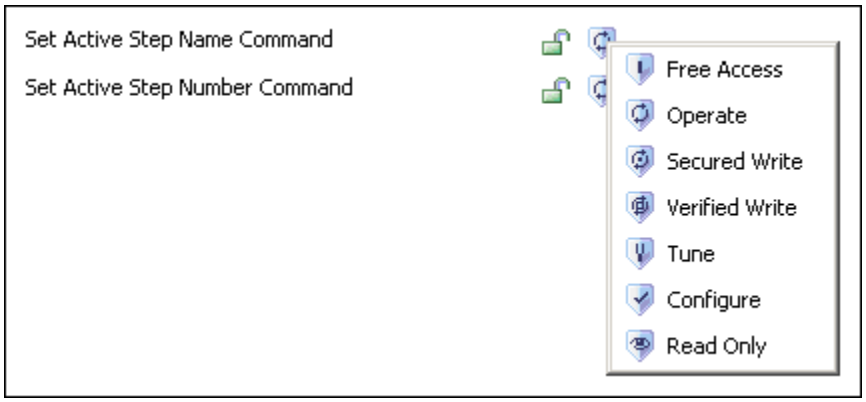
This section shows you how to change access to the Active Step Command settings.

To change the access rights to the active step command settings:

1. Click the **Settings** tab. The **Settings** panel appears.
2. Select the appropriate security settings for either **Set Active Step Name Command** or **Set Active Step Name**

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Command using the security classification shield.



This changes your access rights to the **StepNameCmd** or **StepNumCmd** attributes.

Modify access rights to the initial step command setting

This section shows you how to change access rights to the Initial Step command setting.

To change the access rights to the initial step command setting

1. Click the **Settings** tab. The **Settings** panel appears.
2. Select the appropriate **Set Initial Step Command** security settings using the security classification shield.



This changes your access rights to the **PrgStepInitialCmd** attribute.

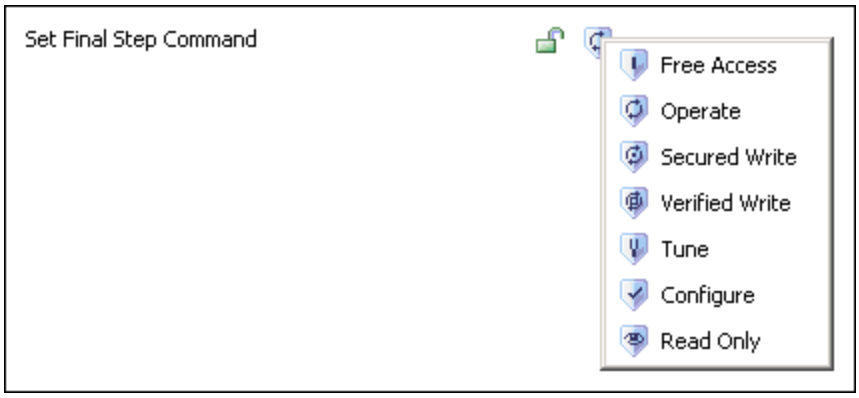
Modify access rights to the final step command setting

This section shows you how to change access rights to the Final Step command setting.

To change the access rights to the final step command setting:

1. Click the **Settings** tab. The **Settings** panel appears.
2. Select the appropriate **Final Step Command** security settings using the security classification shield.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.



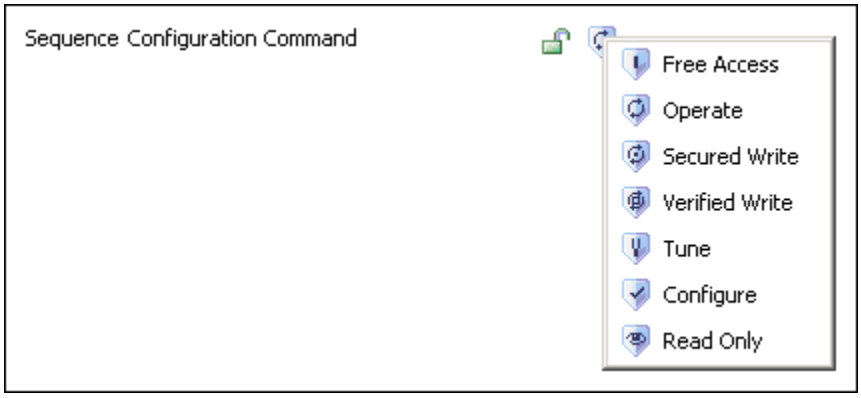
This changes your access rights to the **PrgStepFinalCmd** attribute.

Modify access rights to the sequence configuration command

You can prevent the user from using the Sequence Configuration command at run time. This is done by restricting access to the attribute **PrgSeqConfigCmd**.

To change the access rights to the Sequence Configuration command:

1. Click the **Settings** tab. The **Settings** panel appears.
2. Select the appropriate **Sequencer Configuration Command** security settings using the security classification shield.



Modify access rights to the output value arrays of a selected step

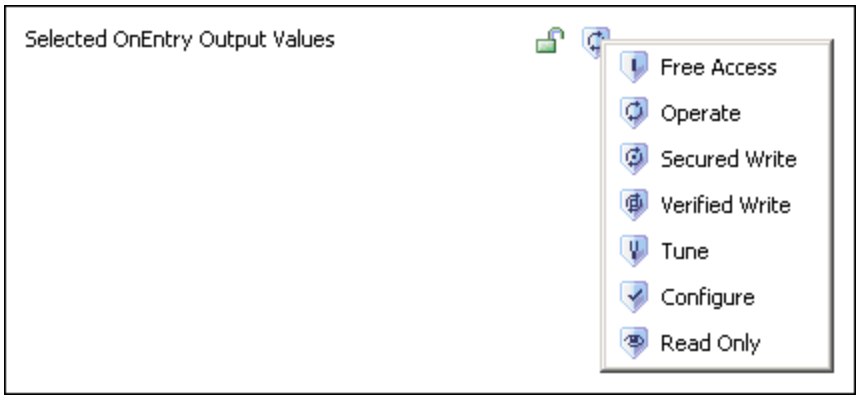
You can prevent the user from instructing the step program to change the on Entry and on Exit output values of the selected step. This is done by restricting access to the following two attributes:

- **Selected.OnEntryOutputValues**
- **Selected.OnExitOutputValues**

To change the access rights to the On Entry output values of the selected step:

1. Click the **Settings** tab. The **Settings** panel appears.
2. Select the appropriate **OnEntryOutputValues** security classification by using the security classification shield.

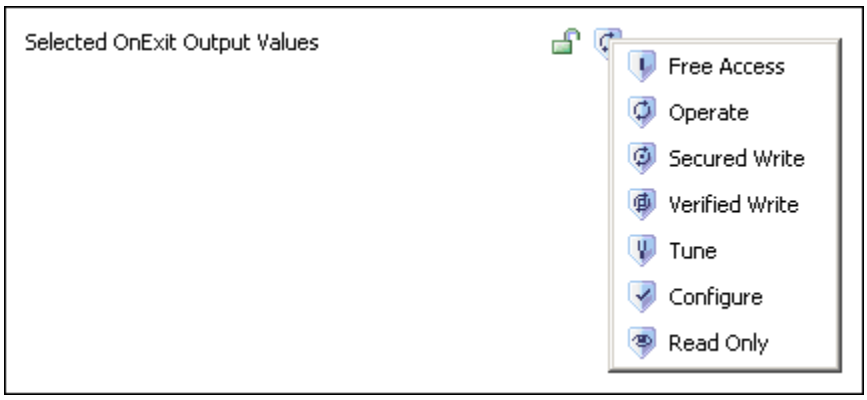
© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.



This changes your access rights to the **Selected.OnEntryOutputValues** attribute.

To change the access rights to the On Exit output values of the selected step:

1. Click the **Settings** tab. The **Settings** panel appears.
2. Select the appropriate **OnExitOutputValues** security classification by using the security classification shield.



This changes your access rights to the **Selected.OnExitOutputValues** attribute.

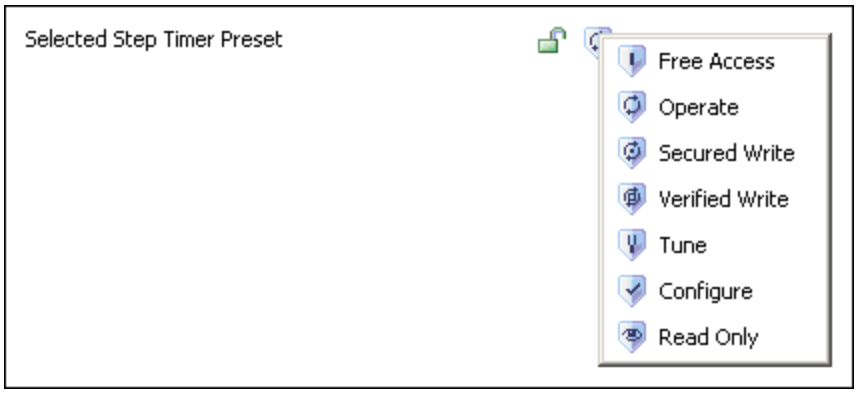
Modify access rights to the timer presets of a selected step

You can prevent the user from instructing the step program to change the timer presets for the step condition and jump condition at run time. This is done by restricting access to the following two attributes; **Selected.StepTimerPreset** and **Selected.JumpTimerPreset**.

To change the access rights to the step timer preset of the selected step:

1. Click the **Settings** tab. The **Settings** panel appears.
2. Select the appropriate **StepTimerPreset** security classification by using the security classification shield.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.



This changes your access rights to the **Selected.StepTimerPreset** attribute.

To change the access rights to the jump timer preset of the selected step:

1. Click the **Settings** tab. The **Settings** panel appears.
2. Select the appropriate **JumpTimerPreset** security classification by using the security classification shield.



This changes your access rights to the **Selected.JumpTimerPreset** attribute.

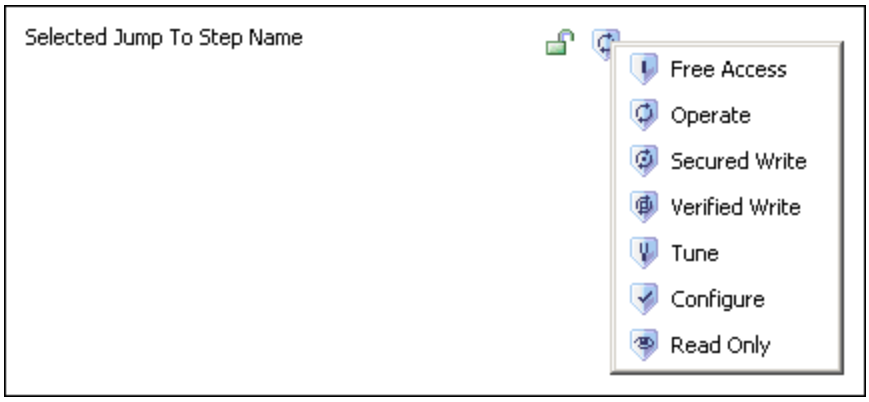
Modify access rights to the jump destination setting of a selected step

You can prevent the user from changing the jump destination of any selected step in the step program. This is done by restricting access to the attribute **Selected.JumpToStepName**.

To change the access rights to the selected jump to Step Name of the selected step:

1. Click the **Settings** tab. The **Settings** panel appears.
2. Select the appropriate **JumpToStepName** security classification by using the security classification shield.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.



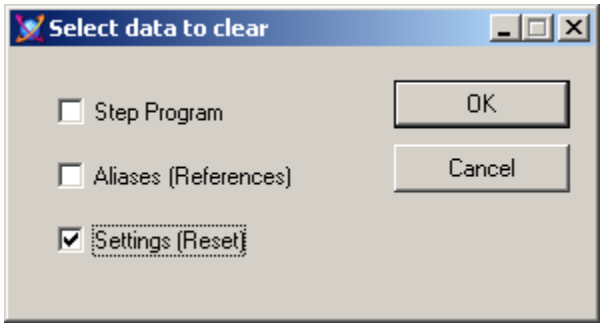
This changes your access rights to the **Selected.JumpToStepName** attribute.

Clear settings

You can clear the settings on the settings panel with the **Clear** button. This resets the attributes in the **Settings** section of the **Settings** panel, except those that are locked.

To clear all settings:

1. Click the **Settings** tab. The **Settings** panel appears.
2. Click the **Clear** button. The **Select data to clear** dialog appears.



3. Make sure the **Settings (Reset)** option is selected and click **OK**. A confirmation dialog appears.
4. Click **Yes**. The settings are reset to their defaults if they are not locked.

Import and export the Sequencer program configuration

This section shows you how to import and export the Sequencer program configuration, such as the step program, Alias definitions and general program settings at design time.

Sequencer program configurations can be imported and exported for use with Sequencer Objects or with third party applications that support XML.

Export the step program

You can export the step program from the Sequencer object editor at any time. The exported data is written to an XML file that can also contain Alias configuration and settings for the Sequencer object.

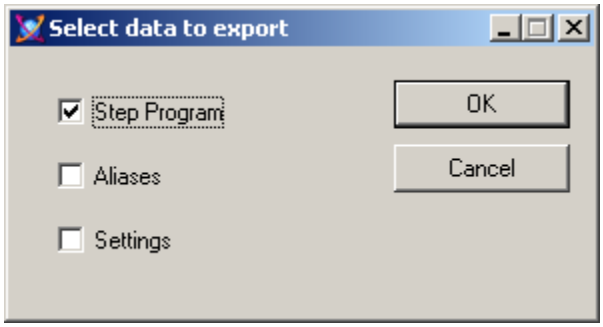
© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

The following Step Program configuration data is exported.

- Step Program Name (attribute: PrgName)
- Step Program Comment (attribute: PrgComment)
- Initial Step (attribute: PrgStepInitial)
- Final Step Sequence (attribute: PrgStepFinal)
- Step Sequence (attribute: PrgStepProgram)

To export the Step Program:

1. Click the **Step Program** tab. The **Step Program** panel appears.
2. Click **Export**. The **Select data to export** dialog box appears.



3. Select the **Step Program** check box. Click **OK**. The **Save As** dialog box appears. Browse to a location and enter a file name in the **File Name** field for the XML file.
4. Click **Save**. An XML file at the specified location is saved.

Export the alias definitions

You can export the Alias definitions from the Sequencer object editor at any time. The exported data is written to an XML file that can also contain the step program and settings for the Sequencer object.

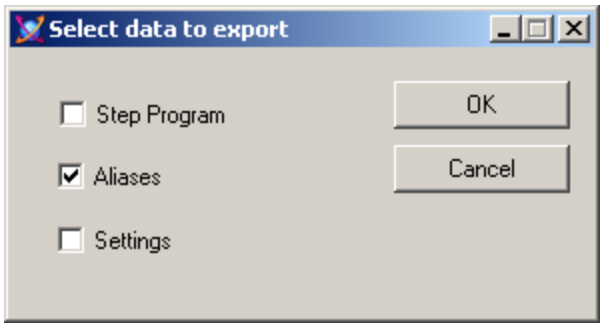
The following Alias configuration data is exported.

- Alias Name
- Attribute Name

To export all Alias definitions:

1. Click the **Aliases** tab. The **Aliases** panel appears.
2. Click the **Export** button. The **Select data to export** dialog box appears.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.



Note: You can export Aliases from either the Step Program panel or from the Settings panel. Depending from which panel you use, the **Select data to export** dialog box shows different default settings. Make sure the **Aliases** check box is selected to export the Aliases.

3. Select the **Aliases** check box. Click **OK**. Browse to a location and enter a file name in the **File Name** box for the XML file.
4. Click **Save**. An XML file is saved to the specified location.

Export program settings

You can save the Sequencer program settings which contain the settings for following items:

- Initial Command
- Halt On Condition Error
- Halt On Output Error
- Auto Resume after Restart/Failover

To export the program settings:

1. Click on the **Settings** tab. The **Settings** panel appears.
2. Click the **Export** button. The **Select data to export** dialog box appears.



3. Select the **Settings** check box and click **OK**. Browse to a location and enter a file name in the **File Name** box for the XML file.
4. Click **Save**. An XML file is saved to the specified location.

Import a step program

You can import the step program from an XML file at any time. The XML file may contain also Alias configuration

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

and settings for the Sequencer object.

Note: Before you import a step program, export any existing step program that you want to keep.

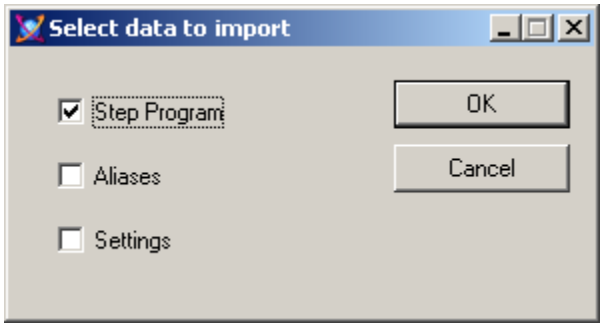
The following Step Program configuration data is imported.

- Step Program Name (attribute: PrgName)
- Step Program Comment (attribute: PrgComment)
- Initial Step (attribute: PrgStepInitial)
- Final Step Sequence (attribute: PrgStepFinal)
- Step Sequence (attribute: PrgStepProgram)

  Caution: You cannot import a step program if the Step Program (PrgStepProgram) is locked in the parent template.

To import a Step Program:

1. Click the **Step Program** tab. The **Step Program** panel appears.
2. Click the **Import** button. The **Open** dialog box appears.
3. Browse to the location of the XML file to load. Select it and click **Open**. The **Select data to import** dialog box appears.



4. Select the **Step Program** check box. Click **OK**. The existing Step Program is overwritten by the configuration from the XML file. Also the Validation window opens and validates the imported configuration.

   Note: If the **Step Program** option is disabled in the **Select data to import** dialog box, the corresponding attribute cannot be updated because it is locked in its parent template. When the **PrgAliasConfig** attribute is locked, importing a new step program with Aliases not defined in the **PrgAliasConfig** attribute terminates the import and the user is notified with an error message. If there are duplicate StepNames in the new Sequencer program, the import is terminated. If the new Sequencer program is missing information such as triggers, timers, jump to step names and/or attribute references, the import succeeds.

Import alias definitions

You can import Alias definitions from an XML file at any time. The XML file may contain also the step program and settings for the Sequencer object.

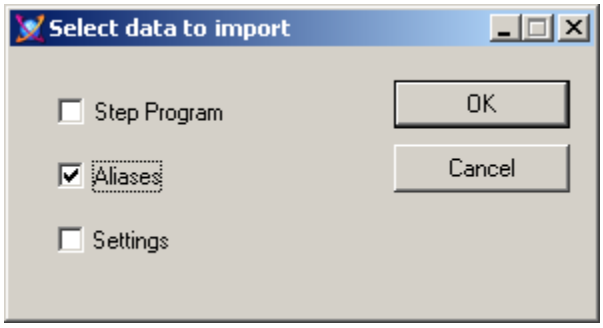
Caution: You cannot import the Alias definitions if the Alias definitions (PrgAliasConfig) are locked in the parent.

To import alias definitions:

1. Click the **Aliases** tab. The **Aliases** panel appears.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

2. Click the **Import** button. The **Open** dialog box appears.
3. Browse to the location of the XML file to load and select it.
4. Click the **Open** button. The **Select data to import** dialog box appears.



5. Click **OK**. The existing Aliases are overwritten by the configuration from the XML file. Also the Validation window opens and validates the imported configuration.

Note: If the **Aliases** option is disabled in the **Select data to import** dialog box, the corresponding attribute cannot be updated because it is locked in its parent template. When the **PrgAliasConfig** attribute is locked, importing a new step program with Aliases not defined in the **PrgAliasConfig** attribute terminates the import and the user is notified with an error message. If there are duplicate StepNames in the new Sequencer program, the import is terminated. If the new Sequencer program is missing information such as trigger, timer, jump to step name and/or attribute reference, the import succeeds.

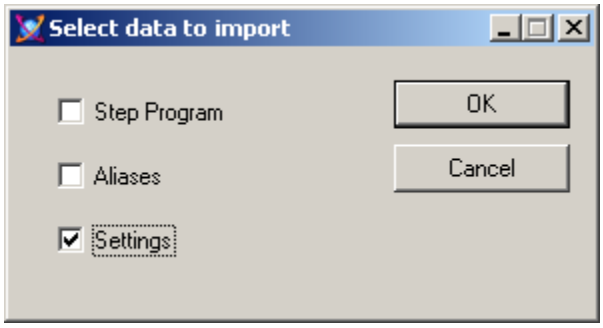
Import program settings

You can import program settings from an XML file at any time. The XML file may contain also the step program and Alias definitions for the Sequencer object.

Caution: You cannot update settings that are locked in the parent.

To import the program settings:

1. Click the **Settings** tab. The **Settings** panel appears.
2. Click the **Import** button. The **Open** dialog box appears.
3. Browse to the location of the XML file to load and select it.
4. Click the **Open** button. The **Select data to import** dialog box appears.



Note: If the **Settings** option is disabled in the **Select data to import** dialog box, the corresponding attribute cannot be updated. This is due either to a security classification of read-only or to being locked in its parent template.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

5. Click **OK**. The existing program settings are overwritten by the configuration from the XML file.

Run the Sequencer

After you configure the Sequencer Object, you can deploy it. This section shows you important information

- on the regular deployment operation
- when restart/failover conditions occur that need your attention

It covers

- what the restart/failover condition is
- how the running sequence program is affected by them
- what you must do to make sure operations can continue smoothly and safely

Deploy the Sequencer object

When the Sequencer object is deployed, the Sequencer object starts up. It parses the sequence program that is stored as XML and invokes the code. The following attributes are affected:

- the PrgSeqConfigStatus attribute is set to the value Updating for a moment before it switches to Ready
- the ExecutionState attribute is set to Initializing while the sequencer acquires first update from all configured aliases. If the binding with the configured attributes takes longer time or the quality of the referenced attributes is initializing, the sequencer may remain in Initializing state until it times out.

  Note: If the Sequencer object is deployed offscan, its execution state is set to "Initializing". The initializing state does not time out.

Restart and failover conditions

This section shows you how the Sequencer Object handles restart/failover conditions.

What are restart/failover conditions?

The following events can be restart/failover events:

- the machine performs a failover
- the Sequencer object is switched offscan, then onscan again
- the hosting AppEngine is shutdown and restarted

How does the running sequence program react to a restart/failover event?

The Sequencer object retains its configuration after a restart/failover event. By default, after a restart/failover event, the sequencer sets itself to a safe execution state where no further writes are automatically performed. If the execution state was Running or RunningSingleStep before the event, the sequencer sets itself to

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

RunningHeld after the event.

Note: During failover, it is not recommended to load, save or update the configuration file through the attributes PrgSeqConfigCmd, PrgSeqConfig and PrgFileNameToLoadSave. If a failover occurs during loading, saving or updating, the PrgSeqConfigCmd is aborted and the sequencer does not raise an alarm or indicate the failed action after the failover completes.

The following table shows you how the execution state changes after a restart/failover event:

Execution state before event	Execution state after event
Running	RunningHeld
RunningSingleStep	RunningHeld
SingleStepTransitionReady	SingleStepTransitionReady
RunningHeld	RunningHeld
Stopped	Stopped
StoppedComplete	StoppedComplete
StoppedError	StoppedError

Detect restart/failover events

You can detect restart/failover events by configuring the Program Execution Halted alarm.

Note: The alarm is raised only if prior to the restart/failover event the Execution State was Running or RunningSingleStep and AutoResume is false. Halting the program execution by setting the ExecutionStateCmd attribute to Hold does not raise this alarm.

Configure alarms to detect if program execution halts

You can configure the Sequencer to generate an alarm if the execution is halted after a restart/failover event. The alarm message includes the name and state of the Sequencer and the cause of the halting. The alarm returns to normal when an execution command is issued.

To configure Sequencer to generate an alarm when program execution is halted:

1. Click the **Settings** tab. The **Settings** panel appears.
2. Select the check box **Detect Execution Halted**.
3. If necessary, enter an alarm priority in the corresponding text field. The range of valid alarm priorities is from 0 to 999. The default is 500.

Use attributes to detect if program execution halts

You can detect a restart/failover event by looking at the values of the attributes ExecutionHalted.Condition and ExecutionHaltedDesc.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

To detect that execution has halted:

1. View the value of the ExecutionHalted.Condition attribute
2. View the value of the ExecutionHaltedDesc - when ExecutionHalted.Condition is true, this attribute provides the following description: Sequencer Program:<Program Name> was halted due to <reason>, where the reason can be either offscan, shutdown or failover.

Note: For more information on the attributes ExecutionHalted.Condition and ExecutionHaltedDesc, see Sequencer object help.

Resume operation after a restart/failover event

After a restart/failover event occurs, the Sequencer object that was previously in Running or RunningSingleStep execution state is now in RunningHeld state. There are two possibilities of what may happen next, depending on the attribute AutoResume that you can configure on the **Settings** panel:

- if AutoResume is false, the operator needs to check the operation status and then to resume the sequence program by setting the attribute ExecutionStateCmd to Resume. This is the default setting.
- if AutoResume is true, the sequence program resumes automatically. No intervention of the operator is required.

Automatically resume execution after a restart/failover event

You can instruct the Sequencer object to resume execution after a restart/failover event by setting the **Auto Resume after Restart/Failover** option in the **Settings** panel.

To set the Auto Resume after Restart/Failover option:

1. Click the **Settings** tab. The **Settings** panel appears.
2. Select the check box **Auto Resume after Restart/Failover** in the Settings group.



What happens to the execution state after the sequence program resumes?

When the sequence program resumes and the event is caused by:

- failover or by AppEngine shutting down and starting up, the execution state goes from Initializing to the execution state it was in before the event occurred.
- switching the Sequencer object offscan and then onscan, the execution state goes directly to the execution state it was in before the event occurred.

The following table shows how attributes and features behave after the sequence program is resumed:

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Attribute or feature	Behavior
On Entry Outputs	are not written again when AutoResume is true
Timers	are frozen (excluding the time during failover, object offscan or object shutdown) and continues if AutoResume is set and the execution state before the failover event was Running or RunningSingleStep
Step Execution	is in the step body of the last active step **Note:** After switching offscan and then onscan, the active step is the next step if the restart/failover happened after the step condition was met (in case of a large scan period) .
Current.StepTimerRunning	persists the state of the timer so conditions can be evaluated accurately
Current.JumpTimerRunning	persists the state of the timer so conditions can be evaluated accurately

Note: After a failover event, the object goes to the last (current) step and is put in RunningHeld state. If it happens that the failover event occurs after the condition is met and after the OnExit outputs are written then OnExit outputs are written again when the Resume or Advanced command is issued.

Sequencer program commands and states

This section shows you how to work with the Sequencer Object after it is deployed. At run time, you can monitor and control the step program by viewing and changing the attributes of the Sequencer Object.

View and monitor the current active step

You can view the configuration or the execution of the step that is being currently processed (active step) by viewing the attributes that are prefixed with "Current".

To view the number and name of the currently active step refer to the table below:

Tasks	Attributes
View step number	`.Current.StepNum`
View step name	`.Current.StepName`

For a detailed overview of the attributes and their properties, see Sequencer object help.

View the configuration of the current active step

You can view the configuration of the step that is being currently processed.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

The following table is an overview of the attributes for monitoring the active step configuration. You can view them in InTouch or Object Viewer:

Tasks	Attributes
View Step Configuration	`.Current.StepConditionCfg`
View Write On Exit Setting for the Step Transition	`.Current.StepWriteOnExit`
View Jump Configuration	`.Current.JumpConditionCfg`
View Write On Exit Setting for the Jump Transition	`.Current.JumpWriteOnExit`
View Jump destination step	`.Current.JumpToStepName`
View number of On Entry outputs	`.Current.OnEntryOutputCnt`
View names of Aliases associated with On Entry outputs	`.Current.OnEntryAliasNames[-1]`
View values that are written to On Entry outputs	`.Current.OnEntryOutputValues[-1]`
View number of On Exit outputs	`.Current.OnExitOutputCnt`
View names of Aliases associated with On Exit outputs	`.Current.OnExitAliasNames[-1]`
View values that are written to On Exit outputs	`.Current.OnExitOutputValues[-1]`

For a detailed overview of the attributes and their properties, see Sequencer object help.

Monitor execution of the current active step

You can view the processing state of the current step.

The following table is an overview of the attributes for monitoring the active step execution. You can view them in InTouch or Object Viewer:

Tasks	Attributes
View when the step became active	`.Current.StepStartTime`
View how long the step has been active	`.Current.StepDuration`
Identify if the step condition is fulfilled	`.Current.StepConditionState`
Identify if the step timer is running	`.Current.StepTimerRunning`

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Tasks	Attributes
View how much time is remaining for the step timer	`.Current.StepTimeRemaining`
Identify if the jump condition is fulfilled	`.Current.JumpConditionState`
Identify if the jump timer is running	`.Current.JumpTimerRunning`
View how much time is remaining for the jump timer	`.Current.JumpTimeRemaining`

For a detailed overview of the attributes and their properties, see Sequencer object help.

Control the program flow at run time

You can control step program flow by telling Sequencer to

- start or stop program execution
- resetting program execution
- hold or resume program execution
- advance a step
- run in Single Step mode
- confirm a step transition when running in Single Step mode
- jump to a specific step

  Note: This is done by writing execution commands to the attribute ExecutionStateCmd. You can query the current state of execution by reading the value of the attribute ExecutionState.

The following terminology used in the tables of this section:

- Current State - the current execution of the step program. You can find out the current state by reading the attribute ExecutionState. Only certain combinations of current state and execution commands produce a meaningful result.
- Target State - the state of the step program execution after you have written the execution command to the attribute ExecutionStateCmd. This is your desired execution state.
- Target State Timer - after you write the execution command to the attribute ExecutionStateCmd, the timers act accordingly. They may restart, continue, or stop.
- Target Step - the step that becomes active after you write the execution command to the attribute ExecutionStateCmd.
- Write OnExit - informs you whether Sequencer writes values to the On Exit outputs, if configured, of the currently active step after you write the execution command.
- Write OnEntry - informs you whether Sequencer writes values to the On Entry outputs of the step that is active after the currently active step.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Start or stop program execution

At run time, you can start or stop program processing by writing to the Execution State Command (ExecutionStateCmd) attribute.

Start program execution

To start program execution write Start to the attribute ExecutionStateCmd.

You can start program execution if the current program execution state is one of the following;

- Stopped
- RunningHeld
- RunningSingleStep
- SingleStepTransitionReady
- StoppedError

  If the command completes successfully the Execution State (attribute ExecutionState) changes to Running.

The following table shows you how the target state and target state timer change depending on the current execution state when the **ExecutionStateCmd** is set to **Start**. It also shows you whether the On Exit and On Entry values are written:

Current state	**Target state** **Target state timer** **Target step**	**Execute OnExit** **Execute OnEntry**
Stopped	State: Running Timer: Restart Step: Current	OnExit: No OnEntry: Yes
RunningHeld	State: Running Timer: Continue Step: Current	OnExit: No OnEntry: No
RunningSingleStep	State: Running Timer: Continue Step: Current	OnExit: No OnEntry: No
SingleStepTransitionReady	State: Running (StopComplete if in the Final step) Timer: Restart Step: Next Step (Current Step if it is the final step)	OnExit: No OnEntry: Yes (next step) No (if final step)
StoppedError	State: Running Timer: Restart Step: Current	OnExit: No OnEntry: Yes

For a detailed overview of the execution state attribute and its properties, see Sequencer object help.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

For a detailed overview of the state transition and information on quality, timers and target steps that are effected by this execution command, see Sequencer state transition tables.

Stop program execution

To stop program execution write Stop to the attribute ExecutionStateCmd.

You can stop program execution if the current program execution state is one of the following;

- Running
- RunningHeld
- RunningSingleStep
- SingleStepTransitionReady.

  If the command completes successfully the Execution State (attribute ExecutionState) changes to Stopped.

  Note: If you are using sufficiently large scan periods in the hosting AppEngine (such as 5 seconds) and you stop program execution after the current step completes (the conditions are satisfied and the OnExit outputs are written) but before the next execution cycle (in the middle between two execution cycles) then and only then the Current.xxx attributes correspond to the next step. If the step is still evaluating conditions and you stop program execution then Current.xxx attributes point to the current step.

The following table shows you how the target state and target state timer change depending on the current execution state when the **ExecutionStateCmd** is set to **Stop**. It also shows you whether the On Exit and On Entry values are written:

Current state	**Target state** **Target state Timer** **Target step**	**Execute OnExit** **Execute OnEntry**
Running	State: Stopped Timer: Stopped Step: Current	OnExit: No OnEntry: No
RunningHeld	State: Stopped Timer: Stopped Step: Current	OnExit: No OnEntry: No
RunningSingleStep	State: Stopped Timer: Stopped Step: Current	OnExit: No OnEntry: No
SingleStepTransitionReady	State: Stopped Timer: Stopped Step: Current	OnExit: No OnEntry: No

For a detailed overview of the execution state attribute and its properties, see Sequencer object help.

For a detailed overview of the state transition and information on quality, timers and target steps that are effected by this execution command, see Sequencer state transition tables.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Reset program execution

At run time, you can reset program execution by writing to the Execution State Command (ExecutionStateCmd) attribute. Program processing is set to Initial Command and Initial Step. The first step is used if no Initial Step is specified.

Reset program execution

To reset program execution write Reset to the attribute ExecutionStateCmd.

You can reset program execution if the current program execution state is one of the following;

- Running
- RunningHeld
- RunningSingleStep
- SingleStepTransitionReady
- Stopped
- StoppedComplete
- StoppedError

  If the command completes successfully the Execution State (attribute ExecutionState) changes to the initial command setting.

The following table shows you how the target state and target state timer change depending on the current execution state. It also shows you whether the On Exit and On Entry values are written:

Current state	Target state Target state timer Target step	Execute OnExit Execute OnEntry
Running	State: InitialCommand Timer: Restart Step: Initial step or first step	OnExit: No OnEntry: Yes (if InitialCommand <> Stop)
RunningHeld	State: InitialCommand Timer: Restart Step: Initial step or first step	OnExit: No OnEntry: Yes (if InitialCommand <> Stop)
RunningSingleStep	State: InitialCommand Timer: Restart Step: Initial step or first step	OnExit: No OnEntry: Yes (if InitialCommand <> Stop)
SingleStepTransitionReady	State: InitialCommand Timer: Restart Step: Initial step or first step	OnExit: No OnEntry: Yes (if InitialCommand <> Stop)

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Current state	Target state Target state timer Target step	Execute OnExit Execute OnEntry
Stopped	State: InitialCommand Timer: Restart Step: Initial step or first step	OnExit: No OnEntry: Yes (if InitialCommand <> Stop)
StoppedComplete	State: InitialCommand Timer: Restart Step: Initial step or first step	OnExit: No OnEntry: Yes (if InitialCommand <> Stop)
StoppedError	State: InitialCommand Timer: Restart Step: Initial step or first step	OnExit: No OnEntry: Yes (if InitialCommand <> Stop)

For a detailed overview of the execution state attribute and its properties, see Sequencer object help.

For a detailed overview of the state transition and information on quality, timers, and target steps that are effected by this execution command, see Sequencer state transition tables.

Hold or resume program execution

You can hold program execution to analyze outputs, debug your program or avert a critical condition. After you hold program execution you can tell it to continue by resuming processing. This is done by writing to the Execution State command (ExecutionStateCmd) attribute.

Note: RunningHeld only occurs in the Step Body.

Hold program execution

To hold program execution write Hold to the attribute ExecutionStateCmd.

You can hold program execution if the current program execution state is one of the following;

- Running
- RunningSingleStep
- SingleStepTransitionReady
- Stopped
- StoppedError.

  If the command completes successfully the Execution State (attribute ExecutionState) changes to RunningHeld.

The following table shows you how the target state and target state timer change depending on the current execution state. It also shows you whether the On Exit and On Entry values are written:

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Current state	Target state Target state timer Target step	Execute OnExit Execute OnEntry
Running	State: Running -> RunningHeld Timer: Freeze at current value Step: Current (or next step body)	OnExit: NO. (Yes, if target step is Next Step body) OnEntry: NO. (Yes, if target step is Next Step body)
RunningSingleStep	State: Running -> RunningHeld Timer: Freeze at current value Step: Current (or next step body)	OnExit: NO. (Yes, if target step is Next Step body) OnEntry: NO. (Yes, if target step is Next Step body)
SingleStepTransitionReady	State:Running -> RunningHeld Timer: Restart and freeze when reaching RunningHeld Step: Next step	OnExit: No OnEntry: Yes
Stopped	State: Running -> RunningHeld Timer: Restart and freeze when reaching RunningHeld Step: Current	OnExit: No OnEntry: Yes
StoppedError	State: Running -> RunningHeld Timer: Restart and freeze when reaching RunningHeld Step: Current	OnExit: No OnEntry: Yes

For a detailed overview of the execution state attribute and its properties, see Sequencer object help.

For a detailed overview of the state transition and information on quality, timers and target steps that are effected by this execution command, see Sequencer state transition tables.

Resume program execution

To resume program execution write Resume to the attribute ExecutionStateCmd.

You can resume program execution only if the current program execution state is RunningHeld. If the command completes successfully the Execution State (attribute ExecutionState) changes to the same state as before the hold command was written.

Note: If the execution state was Stopped (or StoppedError) before holding program execution, resuming program execution puts the execution state back in Stopped (or StoppedError) state again, and not Running state.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Resuming always puts the Sequencer Object into the state it was before its execution was held.

The following table shows you how the target state and target state timer change depending on the current execution state. It also shows you whether the On Exit and On Entry values are written:

Current State	Target state Target state timer Target step	Execute OnExit Execute OnEntry
RunningHeld	State: State before Hold command Timer: Continue in Timer state before Hold command Step: Current	OnExit: No OnEntry: No

For a detailed overview of the execution state attribute and its properties, see Sequencer object help.

For a detailed overview of the state transition and information on quality, timers and target steps that are effected by this execution command, see Sequencer state transition tables.

Advance a step

You can tell the step program to advance to the next step without waiting for the step condition to become true.

Sequencer then

- writes the On Exit values of the current step if the step Write OnExit is true
- transitions to the next step
- writes the On Entry values of the next step
- continuously evaluates the step and jump condition of the next step.

Advance a step

To advance program execution to the next step write Advance to the attribute ExecutionStateCmd.

You can advance program execution if the current program execution state is one of the following;

- Running
- RunningHeld
- RunningSingleStep
- SingleStepTransitionReady
- Stopped
- StoppedError.

  If the command completes successfully the Execution State (attribute ExecutionState) changes to the values as shown in the table below.

  Note: If you instruct the Sequencer Object to advance a step and the current execution state is either Running, RunningSingleStep or RunningHeld and the current step is configured to WriteOnExit it takes two execution cycles to finally advance to the next step.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

The following table shows you how the target state and target state timer change depending on the current execution state. It also shows you whether the On Exit and On Entry values are written:

Current state	Target state Target state timer Target step	Execute OnExit Execute OnEntry
Running	State: Running (StoppedComplete if last step) Timer: Restart Step: Next step (or current if final step)	OnExit: Yes (if Step write on exit is true) OnEntry: Yes (No if last step)
RunningHeld	State: RunningHeld (StoppedComplete if last step) Timer:Restart Step: Next step (or current if final step)	OnExit: Yes, if Step write on exit is true No, if last step OnEntry: Yes (No if last step)
RunningSingleStep	State: RunningSingleStep (StoppedComplete if last step) Timer: Restart Step: Next step (or current if final step)	OnExit:Yes (if Step write on exit is true) OnEntry: Yes (No if last step)
SingleStepTransitionReady	State: RunningSingleStep (StoppedComplete if in the Final step) Timer: Restart Step: Next step (or current if final step)	OnExit: No (OnExit has already been executed in the previous state) OnEntry: Yes (No when this was the final step)
Stopped	State: Stopped Timer: Stopped Step: Next step (or current if final step)	OnExit: No OnEntry: No
StoppedError	State:Stopped Timer: Stopped Step: Next step (or current if final step)	OnExit: No OnEntry: No

For a detailed overview of the execution state attribute and its properties, see Sequencer object help.

For a detailed overview of the state transition and information on quality, timers and target steps that are effected by this execution command, see Sequencer state transition tables.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Run in single step mode

You can instruct Sequencer to execute the program in Single Step mode. In Single Step mode the program executes as follows:

- Step is active
- On Entry values are written to the outputs
- Step condition or jump condition is fulfilled
- On Exit values are written to the outputs
- Program waits for your confirmation
- After your confirmation, processing continues at the next step or a specified step to jump to

Initiate single step mode

To initiate the program in Single Step mode write SingleStep to the attribute ExecutionStateCmd.

You can run the program in Single Step mode if the current program execution state is one of the following;

- Running
- RunningHeld
- Stopped
- StoppedError

  If the command completes successfully the Execution State (attribute ExecutionState) changes to RunningSingleStep.

The following table shows you how the target state and target state timer change depending on the current execution state. It also shows you whether the On Exit and On Entry values are written:

Current State	**Target state** **Target state timer** **Target step**	**Execute OnExit** **Execute OnEntry**
Running	State: RunningSingleStep Timer: Continue Step: Current	OnExit: No OnEntry: No
RunningHeld	State: RunningSingleStep Timer: Continue Step: Current	OnExit: No OnEntry: No
Stopped	State: RunningSingleStep Timer: Restart Step: Current	OnExit: No OnEntry: Yes
StoppedError	State: RunningSingleStep Timer: Restart Step: Current	OnExit: No OnEntry: Yes

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

For a detailed overview of the execution state attribute and its properties, see Sequencer object help.

For a detailed overview of the state transition and information on quality, timers and target steps that are effected by this execution command, see Sequencer state transition tables.

Confirm a transition when running in single step mode

To confirm a transition when the program is running in Single Step mode, write Confirm to the attribute **ExecutionStateCmd**.

You can confirm a transition only if the current program execution state is **SingleStepTransitionReady**. If the command completes successfully the Execution State (attribute ExecutionState) changes to **RunningSingleStep** or **StopComplete** (if the currently active step is the final step).

The following table shows you how the target state, target state timer change depending on the current execution state. It also shows you whether the On Exit and On Entry values are written:

Current state	Target state Target state timer Target step	Execute OnExit Execute OnEntry
SingleStepTransitionReady	State: RunningSingleStep (StopComplete if in the Final step) Timer:Restart Step: Next Step (Current step if it was the final step)	OnExit: No (OnExit has already been executed in the previous state) OnEntry: Yes (No when this was the final step)

For a detailed overview of the execution state attribute and its properties, see Sequencer object help.

For a detailed overview of the state transition and information on quality, timers and target steps that are effected by this execution command, see Sequencer state transition tables.

Jump to a specific step

You can force Sequencer to immediately jump to any step of the program. The On Exit values are written if the Jump Write OnExit setting is true.

Jump to a specific step

To jump to a specific step write the step number to the attribute StepNumCmd or the step name to the attribute StepNameCmd.

You can jump to a specific step if the current program execution state is one of the following;

- Running
- RunningHeld
- RunningSingleStep
- SingleStepTransitionReady
- Stopped

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

- StoppedComplete
- StoppedError

If the command completes successfully the Execution State (attribute ExecutionState) changes to to the values as shown in the table below.

Note: If you instruct the Sequencer Object to jump to a specific step and the current execution state is either Running, RunningSingleStep or RunningHeld and the current step is configured to WriteOnExit it takes two execution cycles to finally jump to the destination step.

The following table shows you how the target state and target state timer change depending on the current execution state. It also shows you whether the On Exit and On Entry values are written:

Current state	Target state Target state timer Target step	Execute OnExit Execute OnEntry
Running	State: Running Timer: Restart Step: StepNum	OnExit: Yes (if Jump write on exit is true) OnEntry: Yes
RunningHeld	State: RunningHeld Timer: Restart Step: StepNum	OnExit: Yes (if Jump write on exit is true) OnEntry: Yes
RunningSingleStep	State: RunningSingleStep Timer: Restart Step: StepNum	OnExit: Yes (if Jump write on exit is true) OnEntry: Yes
SingleStepTransitionRewady	State: RunningSingleStep Timer: Restart Step: StepNum	OnExit: No (OnExit has already been executed in the previous state) OnEntry: Yes
Stopped	State: Stopped Timer: Stopped Step: StepNum	OnExit: No OnEntry: No
StoppedComplete	State: Stopped Timer: Stopped Step: StepNum	OnExit: No OnEntry: No
StoppedError	State: Stopped Timer: Stopped Step: StepNum	OnExit: No OnEntry: No

For a detailed overview of the StepNumCmd and StepNameCmd attributes and their properties, see Sequencer object help

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

For a detailed overview of the state transition and information on quality, timers and target steps that are effected by this execution command, see Sequencer state transition tables

Set initial command at run time

You can set the Initial Command at run time by writing the initial command to the attribute PrgInitialCommand. When the execution state is reset, Sequencer continues processing at the Initial Step, or first step if no Initial Step is defined, and changes its execution state to the value of the attribute PrgInitialCommand.

To set the Initial Command at run time

- Write the execution state that Sequencer should go to when it is reset ("Start", "Stop", "Hold", "SingleStep") to the attribute PrgInitialCommand.

For more information on these attributes, see Sequencer object help

Modify the Sequencer program at run time

This section shows you how to modify some of the Sequencer Object parameters at run time, such as

- configuring a selected step
- setting initial step and final step
- configuring the step program or the Aliases by using XML
- loading and saving Sequencer object configuration data

View and change the configuration of a selected step

You can view and change certain configuration parameters of any step by reading or writing to the attributes that are prefixed with the word "Selected". You must select a step before using any of the attributes in the Selected group.

View configuration of a selected step

You can view the following configuration for any selected step in your step program:

- Step name
- Configuration of the Step Condition
- Step Timer Preset
- Write On Exit setting for the Step Transition
- Configuration of the Jump Condition
- Jump timer preset
- Write On Exit setting for the Jump Transition
- Jump destination
- Number of On Entry outputs

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

- Names of Aliases associated with On Entry outputs
- Values that are written to the On Entry outputs
- Number of On Exit outputs
- Names of Aliases associated with On Exit outputs
- Values that are written to the On Exit outputs

To view the configuration of a specific step

Write the step number of the step you want to select to the attribute YourSequencerObject.Selected.StepNum or Selected.StepName.

1. Refer to the following attribute table to retrieve the attribute reference:

Tasks	Attributes
View step name	`.Selected.StepName`
View Step Condition Configuration	`.Selected.StepConditionCfg`
View Step Timer Preset	`.Selected.StepTimerPreset`
View Write On Exit Setting for the Step Transition	`.Selected.StepWriteOnExit`
View Jump Condition Configuration	`.Selected.JumpConditionCfg`
View Jump Timer Preset	`.Selected.JumpTimerPreset`
View Write On Exit Setting for the Jump Transition	`.Selected.JumpWriteOnExit`
View Jump destination step	`.Selected.JumpToStepName`
View number of On Entry outputs	`.Selected.OnEntryOutputCnt`
View names of Aliases associated with On Entry outputs	`.Selected.OnEntryAliasNames[-1]`
View values that are written to On Entry outputs	`.Selected.OnEntryOutputValues[-1]`
View number of On Exit outputs	`.Selected.OnExitOutputCnt`
View names of Aliases associated with On Exit outputs	`.Selected.OnExitAliasNames[-1]`
View values that are written to On Exit outputs	`.Selected.OnExitOutputValues[-1]`

For a detailed overview of the attributes and their properties, see Sequencer object help.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Change configuration of a selected step

You can change the following configuration for any selected step in your step program:

- Values that are written to On Entry outputs
- Values that are written to On Exit outputs
- Step Timer Preset
- Jump Timer Preset
- Write On Exit setting for the Step Transition
- Write On Exit setting for the Jump Transition
- Jump Destination of the Jump Transition

Note: You can change the configuration of the current step by selecting it. When you use the Selected attributes to modify the Timer Preset and On Exit output values of the current step, these changes take effect immediately.

Note: When the PrgStepProgram attribute is locked, run-time configuration changes via the Selected attributes are rejected with Operational Errors.

To change the configuration of a specific step:

1. Write the step number of the step you want to select to the attribute YourSequencerObject.Selected.StepNum
2. Refer to the following attribute table to retrieve the attribute reference:

Tasks	Attributes
Change values that are written to On Entry outputs	`.Selected.OnEntryOutputValues[-1]`
Change Step Timer Preset	`.Selected.StepTimerPreset`
Change Jump Timer Preset	`.Selected.JumpTimerPreset`
Change values that are written to On Exit outputs	`.Selected.OnExitOutputValues[-1]`
Set or clear Write On Exit setting for the Step Transition	`.Selected.StepWriteOnExit`
Set or clear Write On Exit setting for the Jump Transition	`.Selected.JumpWriteOnExit`
Change the Jump Destination of the Jump Transition	`.Selected.JumpToStepName`

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

For a detailed overview of the attributes and their properties, see Sequencer object help.

Note: When PrgStepProgram is locked, the set is rejected with Operational Error and the description is "Step Program Locked".

View the execution order of aliases

The execution order of aliases determines in which order the output values are written to the OnEntry and OnExit aliases. You can view the writing execution order of aliases by reading the attribute PrgStepProgram.

To view the execution order of Aliases:

1. Read the value of the attribute PrgStepProgram.
2. Locate the <ONENTRY> and <ONEXIT> section for any selected step. The order in which the aliases appear here is the order in which the outputs are written.

Change step program and alias configuration at run time

You can update the step program or the Alias configuration or both of them at the same time by using the PrgSeqConfig and PrgSeqConfigCmd attributes. The advantage of this is that you can add extra security to this attribute.

To update these attributes:

- Stop the step program execution so that it is either in Stopped, StoppedComplete or StoppedError state.
- Write the step program configuration and/or Alias configuration and/or settings to the PrgSeqConfig attribute. At this point there is no update to the run-time configuration.
  - Tell the Sequencer object to update its run-time configuration. The contents of the PrgSeqConfig attribute are written to the PrgStepProgram and PrgAliasConfig attributes and the run-time configuration is updated. Also the corresponding attributes from the Settings XML section are updated.

To update Sequencer object configuration at run time:

1. Verify that the step program execution is stopped by reading the attribute ExecutionState. If it is not in one of the stopped states, write the value Stop to the enum attribute ExecutionStateCmd.
2. Write the step program configuration and/or Alias configuration as XML string to the PrgSeqConfig attribute.
3. Write the value Update to the enum attribute PrgSeqConfigCmd.
4. Check the attribute PrgSeqConfigStatus for details on the update:

Value	Description
Updating	The update is in progress.
Ready	The update has completed successfully.
Error	The update encountered a problem. This could be due to a syntax error in the XML data you tried to update the object with.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Additional feedback and results from the update are reflected in the following attributes:

- PrgSeqConfigStatus.Error
- PrgSeqConfigStatus.Desc
- PrgSeqConfigStatus.Code

For a detailed overview of these attributes and their properties, see Sequencer object help.

After the configuration data is successfully updated with the value from PrgSeqConfig, the step program starts running automatically. The attribute PrgSeqConfig is reset to an empty string after a successful update. When you upload configuration data that doesn't contain all parts of a Sequencer object, for example, if there are no Aliases, the configuration from the file is merged with the currently active run time configuration.

Load and save Sequencer configuration

At run time, you can load and save the Sequencer object configuration (step program, Aliases, and settings) from and to an XML file. Saving configuration at run time and loading it at design time is one way of uploading run-time changes.

Important: When you perform a Load or Save at run time, your user credentials may not be the same as when you Import and Export at configuration time. Make sure that the shared locations of XML files are accessible by the ArchestrA account under which the Sequencer object is running.

Save Sequencer configuration at run time

At run time, you can save the Sequencer object configuration (step program, Aliases, and settings) by

- specifying a file name
- writing the save command

To save the run-time configuration to an XML file:

1. Write the file name with path to the string attribute PrgFileNameToLoadSave.
2. Write the value Save to the enum attribute PrgSeqConfigCmd.
3. Check the attribute PrgSeqConfigStatus for details on the save:

Value	Description
Saving	The save is in progress.
Ready	The save has completed successfully and an XML file with the specified name is created at the specified location.
Error	The save encountered a problem. This could be due to insufficient privileges to create the XML file in the specified location. A file error does not affect the error status of the

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Value	Description
	Sequencer object.

Additional feedback and results from the save are reflected in the following attributes:

- PrgSeqConfigStatus.Error
- PrgSeqConfigStatus.Desc
- PrgSeqConfigStatus.Code

For a detailed overview of these attributes and their properties, see Sequencer object help.

Load Sequencer configuration at run time

At run time, you can load Sequencer object configuration (step program, Aliases, and settings) by

- stopping the current step program execution
- specifying a file name
- writing the load command

To save the run-time configuration to an XML file:

1. Verify that the step program execution is stopped by reading the attribute ExecutionState. If it is not stopped, write the value Stop to the enum attribute ExecutionStateCmd.

   For more information on these attributes, see Sequencer object help.
2. Write the file name with path to the string attribute PrgFileNameToLoadSave.
3. Write the value Load to the enum attribute PrgSeqConfigCmd.
4. Check the attribute PrgSeqConfigStatus for details on the load:

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Value	Description
Loading	The load is in progress.
Ready	The load has completed successfully and the run-time configuration now contains the configuration of the specified XML file. **Note:** If a failover occurs during the load, the value of PrgSeqConfigStatus remains Ready. **Caution:** If a failover occurs while a Load, Save or Update operation is in progress, the operation is aborted. PrgSeqConfigStatus returns to Ready. The existing sequence program is retained. Error is not raised.
Error	The load encountered a problem. This could be due to invalid file location or invalid data contained within the file. A file error does not affect the error status of the Sequencer object.

Additional feedback and results from the load are reflected in the following attributes:

- PrgSeqConfigStatus.Error
- PrgSeqConfigStatus.Desc
- PrgSeqConfigStatus.Code

For a detailed overview of these attributes and their properties, see Sequencer object help.

After configuration data is successfully loaded from the XML file

- the step program starts execution based on the value of the InitialCommand specified in the loaded file
- all attributes from the Settings node are updated per the configuration in the XML file.

  Caution: If any of these attributes such as InitialCommand are missing or are blank in the XML file then they are set to their default values. If the Settings node is not specified in the XML file, the corresponding object attributes are not updated.

After a successful load or update, the sequencer starts executing InitialCommand that it is explicitly specified in the sequence program that was just loaded or updated. If an Initial Command is not explicitly specified, the default InitialCommand "Stop" is used.

When you load configuration data from a file that does not contain all parts of a Sequencer object (for example, the Aliases section is missing), the configuration from the file is merged with the currently active run-time configuration.

Upload run-time changes

In Sequencer you cannot upload run-time changes to the step program or Alias configuration by using the

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Upload Runtime Changes option on the context menu of the Sequencer instance. This is due to a current limitation of ArchestrA.

To upload run-time changes:

1. At run time, save the run-time configuration as an XML file. For more information on how to do this, see Save Sequencer configuration at run time.
2. At design time import the XML file. The run-time data that you previously saved is imported. For more information on how to import data, see Load Sequencer configuration at run time.

Set initial and final steps

After deployment or a reset to **ExecutionStateCmd**, the Sequencer starts from the Initial Step. If an Initial Step is not specified, Sequencer starts from the first step. At run time, you can set the Initial Step and the Final Step. The changes take immediate effect.

If the Initial Step and Final Step are set so that:

- the current step is before the Initial Step, Sequencer continues until the final step is reached and continues at the Initial Step
- the current step is after the Final Step, Sequencer continues until the last step of the step program is reached, continues with the Initial Step (or first step) and then cycles between the InitialStep (or first step) and Final Step
- if the final step is specified and the transition conditions of the final step are met, the Sequencer writes OnExit outputs and moves to the StoppedComplete state
- if the final step is not specified and the transition conditions of the last step are met, the Sequencer writes OnExit outputs and returns to the first step.

To set the initial step at run time:

- Write the step name to the attribute PrgStepInitialCmd. The update is reflected in the attribute PrgStepInitial.

Tip: To clear the Final Step, set PrgStepFinalCmd to an empty string. When Final Step is blank, Sequencer does not stop at any step and returns back to the Initial or first step.

For more information on these attributes, see Sequencer object help.

To set the final step at run time:

- Write the step name to the attribute PrgStepFinalCmd. The update is reflected in the attribute PrgStepInitial.

Tip: To set the final step as the last step in the Step Program, pass an empty string to the attribute PrgStepFinalCmd.

Detect errors at run time

At run time, you can use various attributes to detect errors and alarms caused by:

- program execution halting

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

- invalid condition triggers or bad quality values
- failure while writing values to On Entry outputs
- failure writing values to On Exit outputs

Detect if program execution halts

You can detect if the program execution is abnormally halted by the Sequencer by reading the values of the ExecutionHalted.Condition attribute and the ExecutionHaltedDesc attribute.

The program execution is halted and an alarm raised if, for example:

- you configure the **Halt on Condition Error** or **Halt on Output Error** options in the Sequencer editor
- a failover occurs and **AutoResume** is false (and the execution state is Running or SingleStep before the failover)
- the Sequencer object is taken Off Scan and then brought On Scan again and **AutoResume** is false (and the execution state is Running or SingleStep before taking it Off Scan)
- the hosting AppEngine object is restarted On Scan (and the execution state is Running or SingleStep before restarted the hosting AppEngine)
- the computer is rebooted (and the execution state is Running or SingleStep before restarted the hosting AppEngine).

To detect program execution halting:

- Read the values from the following attributes:

Attribute	Description
ExecutionHalted.Condition	TRUE - The program execution was halted FALSE - The program execution is running as controlled by the user
ExecutionHaltedDesc	A description indicating possible causes

For more information on these attributes, see Sequencer object help.

Detect condition trigger failure

You can detect if the Alias used as a condition trigger is invalid, inaccessible or has bad data quality by reading the values of the ConditionTriggerFailure.Condition attribute and the ConditionTriggerFailureDesc attribute.

To detect condition trigger failures:

- Read the values from the following attributes:

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Attribute	Description
ConditionTriggerFailure.Condition	TRUE - The Alias associated with a condition trigger is invalid or contains bad quality values. FALSE - The Alias associated with a condition trigger is valid and contains good quality values.
ConditionTriggerFailureDesc	A description indicating possible causes

For more information on these attributes, see Sequencer object help.

Detect OnEntry output failure

You can detect if the write to one or more On Entry Outputs failed by reading the values of the OnEntryOutputFailure.Condition attribute or the OnEntryOutputFailureDesc attribute.

To detect OnEntry Output failures:

Read the values from the following attributes:

Attribute	Description
OnEntryOutputFailure.Condition	TRUE - The write to an On Entry Output failed FALSE - The write to an On Entry Output did not fail
OnEntryOutputFailureDesc	A description indicating possible causes

For more information on these attributes, see Sequencer object help.

Detect OnExit output failure

You can detect if the write to one or more On Exit Outputs failed by reading the values of the OnExitOutputFailure.Condition attribute or the OnExitOutputFailureDesc attribute.

To detect On Exit Output failures:

- Read the values from the following attributes:

Attribute	Description
OnExitOutputFailure.Condition	TRUE - The write to an On Exit Output failed FALSE - The write to an On Exit Output did not fail
OnExitOutputFailureDesc	A description indicating possible causes

For more information on these attributes, see Sequencer object help.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Detect errors during Sequencer configuration change

You can detect if the Sequencer Configuration change failed during a load or update procedure. For this you can use the values of the ProgramError and ProgramErrorDesc attributes.

To detect program errors:

- Read the values from the following attributes:

Attribute	Description
ProgramError	TRUE - A Program Error has occurred FALSE - No Program Error has occurred
ProgramErrorDesc	A description indicating possible causes

You can monitor the values of the following attributes for more information:

- PrgSeqConfigStatus.Error
- PrgSeqConfigStatus.Desc
- PrgSeqConfigStatus.Code

For more on these attributes, see Sequencer object help.

Redundancy and the Sequencer object

There are a few things to keep in mind when using the Sequencer object with redundancy:

- When the Sequencer object is hosted by a redundant application engine, avoid performing Load and Save operations using a path that refers to a local drive (i.e., C:\Sequence.xml). The Load and Save operations occur relative to the local engine.
- If a failover occurs during a Load, Save or Update operation, the operation is terminated and must be restarted manually. You are not notified of the failure of the operation.

Sequencer object help

This section includes the following:

- Configuration object attributes
- Run-time object attributes
- Alarm attributes
- History attributes

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Configuration object attributes

The following section lists the Sequencer object attributes associated with configuration panels.

- Step program
- Aliases
- Settings

Step program

The following table lists the Sequencer object attributes associated with options on the **Step Program** panel:

Editor option	Associated attribute	Description	Run-time access
Step Program	.PrgStepProgram	Stores the Step Program in an XML format. The step program consists of all the steps and corresponding output values StepProgramName, Step Program Comment, Initial Step Name and Final Step Name. The step program does not include information about the Alias References or settings.	Read-Only
Step Program Name	.PrgStepProgram	Exposed at run time via the PrgName attribute. For more details, see attribute .PrgName in Program attributes.	Read-Only
Initial Step Name	.PrgStepProgram	Exposed at run time via the .PrgStepInitial. Modifiable at run time via PrgStepInitialCmd. For more details, see attribute .PrgStepInitial in Program attributes.	Read-Only

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Editor option	Associated attribute	Description	Run-time access
Final Step Name	.PrgStepProgram	Exposed at run time via the .PrgStepFinal. Modifiable at run time via PrgStepFinalCmd. For more details, see attribute .PrgStepFinal in Program attributes.	Read-Only
Step Program Comment	.PrgStepProgram	For run-time behavior, see attribute .PrgComment in Program attributes.	Read-Only

Aliases

The following table lists the Sequencer object attributes associated with options on the **Aliases** panel:

Editor option	Associated attribute	Description	Run-time access
Alias Definitions	.PrgAliasConfig	Stores the Sequencer Alias/attribute reference configuration in an XML format string.	Read-Only

Settings

The following table lists Sequencer object attributes associated with options on the **Settings** panel.

Editor option	Associated attribute	Description	Run-time access
Initialization Timeout	.InitializationTimeout	Allows you to specify timeout period (ms) for the Initializing state. If the timeout period had elapsed and not all of the references returned valid updates, the following attributes are set: • ExecutionState = StoppedError • ExecutionHalted.Condition = True	Supervisory, User

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Editor option	Associated attribute	Description	Run-time access
		• ExecutionHaltedDesc= "Failed to acquire valid initial updates from input/output(s), program execution halted.”	

Settings

The following table lists Sequencer object attributes associated with options on the **Settings** panel in the **Settings** group box. These four attributes can be imported/exported and load/saved as XML.

Editor option	Associated attribute	Description	Run-time access
Initial Command	.PrgInitialCommand	Sets the initial command the Sequencer starts after deployment or reset "Start". Start the Sequencer (switch to running mode) "Stop". Stop the Sequencer (switch to stopped mode) "SingleStep". Switch to Single Step Mode "Hold". Switch to hold	Supervisory, User
Halt on Condition Error	.PrgHaltOnConditionError	If TRUE (default) then when a Condition Trigger Failure is detected, the Sequencer is halted and ExecutionState is set to StoppedError. If FALSE then when a Condition Trigger Failure is detected, the sequencer does not halt.	Supervisory, User

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Editor option	Associated attribute	Description	Run-time access
Halt on Output Error	.PrgHaltOnOutputError	If TRUE (default) then when an OnEntry or OnExit Output failure is detected, the Sequencer is halted and ExecutionState is set to StoppedError. If FALSE then Sequencer continues.	Supervisory, User
Auto Resume after Restart/Failover	.AutoResume	If the ExecutionState is Running or SingleStep before restart/failover, the Sequencer returns to the RunningHeld state following restart/failover. Set this attribute to True to set the Sequencer to auto resume. Allows user to configure whether the object should auto resume on restart/ failover. If false and ExecutionHalted alarm is enabled, an ExecutionHalted alarm is raised on restart/failover after the ExecutionState is set to RunningHeld. Restart refers to the following: • going back to Onscan after toggling from OnScan to Offscan • AppEngine Startup for checkpoint	Supervisory, User

Alarms

The following table lists Sequencer object attributes associated with options on the Settings panel in the **Alarms** group box:

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Editor option	Associated attribute	Description
Detect Execution Halted	.ExecutionHalted.Alarmed	Enable or disable ExecutionHalted alarm. The ExecutionHalted alarm is raised when the sequencer is abnormally halted.
Priority	.ExecutionHalted.Priority	See Alarm attributes.
Detect Condition Trigger Failure	.ConditionTriggerFailure.Alarmed	Enable or disable ConditionTriggerFailure alarm. The ConditionTriggerFailure alarm is raised when the sequencer failed to evaluate condition trigger of the current step.
Priority	.ConditionTriggerFailure.Priority	See Alarm attributes.
Detect OnEntry Output Failure	.OnEntryOutputFailure.Alarmed	Enable or disable OnEntryOutputFailure alarm. OnEntryOutputFailure alarm is raised if there is error writing to any of the OnEntry outputs.
Priority	.OnEntryOutputFailure.Priority	See Alarm attributes.
Detect OnExit Output Failure	.OnExitOutputFailure.Alarmed	Enable or disable OnExitOutputFailure alarm. OnExitOutputFailure alarm is raised if there is error writing to any of the OnExit outputs.
Priority	.OnExitOutputFailure.Priority	See Alarm attributes.

Locking and security

The following table lists Sequencer object attributes associated with options on the **Settings** panel in the **Security** group box:

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Editor option	Associated attribute	Description
Execution State Command	.ExecutionStateCmd	For run-time behavior, see attribute .ExecutionStateCmd in Execution attributes.
Set Active Step Name Command	.StepNameCmd	For run-time behavior, see attribute .StepNameCmd in Execution attributes.
Set Active Step Number Command	.StepNumCmd	For run-time behavior, see attribute .StepNumCmd in Execution attributes.
Set Initial Step Command	.PrgStepInitialCmd	For run-time behavior, see attribute .PrgStepInitialCmd in Program attributes.
Set Final Step Command	.PrgStepFinalCmd	For run-time behavior, see attribute .PrgStepFinalCmd in Program attributes.
Sequence Configuration Command	.PrgSeqConfigCmd	For run-time behavior, see attribute .PrgSeqConfigCmd in Program attributes.
Selected OnEntry Output Values	.Selected.OnEntryOutputValues	For run-time behavior, see attribute .Selected.OnEntryOutputValues in Selected attributes.
Selected OnExit Output Values	.Selected.OnExitOutputValues	For run-time behavior, see attribute .Selected.OnExitOutputValues in Selected attributes.
Selected Step Timer Preset	.Selected.StepTimerPreset	For run-time behavior, see attribute .Selected.StepTimerPreset in Selected attributes.
Selected Jump Timer Preset	.Selected.JumpTimerPreset	For run-time behavior, see attribute .Selected.JumpTimerPreset in Selected attributes.
Selected Jump To Step Name	.Selected.JumpToStepName	For run-time behavior, see attribute .Selected.JumpToStepName in Selected attributes.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Editor option	Associated attribute	Description
Selected Step Write On Exit	.Selected.StepWriteOnExit	For run-time behavior, see attribute .Selected.StepWriteOnExit in Selected attributes.
Selected Jump Write On Exit	.Selected.JumpWriteOnExit	For run-time behavior, see attribute .Selected.JumpWriteOnExit in Selected attributes.

Run-time object attributes

The following section lists the Sequencer object attributes associated with the run-time environment.

- Execution attributes
- Current attributes
- Program attributes
- Selected attributes

Execution attributes

The following table lists the Sequencer object attributes associated with the Sequencer execution at run time.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Attribute	Description	Run-time access
.ExecutionState	The execution state of the Sequencer can be: • "Running" (1) • "Stopped" (2) • "RunningHeld" (3) • "RunningSingleStep" (4) • "SingleStepTransitionReady" (5) • "StoppedComplete" (6) • "StoppedError" (7) • "Initializing" (8) **Note**: Depending on the current ExecutionState, certain ExecutionStateCmds are allowed while others are not.	Read-Only

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Attribute	Description	Run-time access
.ExecutionStateCmd	Set the mode at run time to control the operation mode of the Sequencer Object: "Start". Start the Sequencer (switch to running mode) "Stop". Stop the Sequencer (switch to stopped mode) "Reset". Start the Sequencer from the InitialStep in the state specified in by the InitialCommand. If InitialStep is blank, start from Step Number 1. "Advance". Advance unconditionally to the next step (manually forced step transition) Set at run time to force the value of StepNum to the next step without evaluating step conditions or timers or jump conditions or jump timers. If StepNum = StepCnt and StepNum < > FinalStep then StepNum advances to the first step (1). If the OnExit condition have not been set yet (step current step is in the step body) the OnExit outputs of the step transition is written if configured in the step condition. "SingleStep". Run Sequencer in Single Step Mode.	Supervisory, User

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Attribute	Description	Run-time access
.ExecutionStateCmd (continued)	"Confirm". Confirm a pending single step transition. Set at run time to confirm the transition to the next step or jump destination in the state: "SingleStepTransitionReady" "Hold" Switch to RunningHeld "Resume" Resume to running from HOLD state **Note:** When in "SingleStepTransitionReady" or "RunningHeld" state, the operator can still move to new steps using the "Advance", "Reset" commands or the StepNumCmd or StepNameCmd attributes.	Supervisory, User
.ExecutionHalted.Condition	Shows that the Sequencer is abnormally halted when: • a condition error occurs and HaltOnConditionError is true • an output error occurs and HaltOnConditionError is true • a failover occurs (without resume) and AutoResume is false • the object is switched OffScan and then OnScan and AutoResume is false • the hosting AppEnging is shutdown and restarted OnScan and AutoResume is false TRUE: The program execution was abnormally halted by the Sequencer. FALSE: The program execution is running as controlled by the user.	Read-Only

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Attribute	Description	Run-time access
.ExecutionHaltedDesc	Warning description to indicate why the execution is halted.	Read-Only
.ConditionTriggerFailure.Condition	Monitors the ConditionTrigger of the CurrentStep and is set to True when a problem is detected. TRUE: Failed to evaluate condition trigger of the current step because • either the trigger attribute quality is not Good • or the reference to the trigger cannot be resolved, coerced or made available. FALSE: No Condition Error detected in the current step.	Read-Only
.ConditionTriggerFailureDesc	Warning description that indicates a possible reason for the ConditionTriggerFailure.	Read-Only
.OnEntryOutputFailure.Condition	TRUE: Failed to evaluate condition trigger of the current step because • the output Alias reference cannot be resolved • or the output value cannot be coerced or the output Alias reference is unavailable. TRUE: In case of an On Entry Output error detected in the current step. FALSE: No OnEntry Output error detected in the current step.	Read-Only
.OnEntryOutputFailureDesc	Warning description for the OnEntryOutputFailure attribute.	Read-Only

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Attribute	Description	Run-time access
.OnExitOutputFailure.Condition	TRUE: In case of an On Exit Output error detected in the current step because • the output Alias reference cannot be resolved • or the output value cannot be coerced or the output Alias reference is unavailable. FALSE: No OnExit Output error detected in the current step.	Read-Only
.OnExitOutputFailureDesc	Warning description for the OnExitOutputFailure attribute.	Read-Only
.StepNameCmd	Set to desired step name at run time to force the Sequencer Object to go unconditionally to this step without evaluating step conditions or timers or jump conditions or jump timers (manually forced jump transition). If the OnExit outputs of the current step have not been set yet, the OnExit values are written if JumpWriteOnExit is true before going to the new step location where the OnEntry output values are set. As soon as the new step position is reached all Current.xxx values are updated. Run time: Forces the Sequencer Object to set the target step as the current step executing all OnExit and OnEntry output value settings.	Supervisory, User

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Attribute	Description	Run-time access
.StepNumCmd	Set to desired step number at run time to force the Sequencer Object to go unconditionally to this step without evaluating step conditions or timers or jump conditions or jump timers (manually forced jump transition). If the OnExit outputs of the current step have not been set yet, the OnExit values are written if the JumpWriteOnExit is true before going to the new step location where the OnEntry output values are set. As soon as the new step position is reached all Current.xxx values are updated. Run time: Forces Sequencer Object to set the target step as the current step executing all OnExit and OnEntry output value settings.	Supervisory, User

Current attributes

The following table lists all Sequencer object attributes associated with the currently active step at run time:

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Attribute	Description	Run-time access
.Current.JumpConditionConfig	The string representation of the step condition configuration. For a full description, read the section XML roots and sub-roots. <condition type>|<timer preset>|<trigger expression> Where: <condition type> consists of 3 characters indicating the condition, trigger, and timer type: <Timer preset> always has the format: <dd>:<hh>:<mm>:<ss> where the delimiter is always: <trigger expression> is the only variable length section of this syntax. It can have the form: <Alias name> for Boolean or data change evaluation (trigger T,F,t,f,c) Where Alias name is any symbolic Alias name used by the Sequencer.	Read-Only
.Current.JumpConditionState	Shows the jump trigger condition evaluation result. TRUE if Jump condition is TRUE else FALSE.	Read-Only
.Current.JumpTimeRemaining	Remaining time of the jump timer in seconds.	Read-Only
.Current.JumpTimerRunning	Shows if the jump timer is running.	Read-Only
.Current.JumpToStepName	Represents the name of the destination step that the Sequencer Object jumps to on jump transition in the current step.	Read-Only
.Current.JumpWriteOnExit	Shows if WriteOnExit is set for the Jump Condition of the current step. TRUE: Outputs are written before jump transition. FALSE: Outputs are not written before jump transition.	Read-Only

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Attribute	Description	Run-time access
.Current.OnEntryAliasNames	Shows the string array list of the current step OnEntry output names.	Read-Only
.Current.OnEntryOutputCnt	Shows the number of outputs in the OnEntry Alias and value arrays.	Read-Only
.Current.OnEntryOutputValues	Shows the string array list of the current step OnEntry output set values.	Read-Only
.Current.OnExitAliasNames	Shows the string array list of the current step OnExit output Alias names.	Read-Only
.Current.OnExitOutputCnt	Shows the number of outputs in the OnExit Alias and value arrays.	Read-Only
.Current.OnExitOutputValues	Shows the string array list of the current step OnExit output set values.	Read-Only
.Current.StepConditionConfig	Shows the string representation of the step condition configuration. For a full description, read the section XML roots and sub-roots. <condition type>|<timer preset>|<trigger expression> Where: <condition type> consists of 3 characters indicating the condition, trigger and timer type: <Timer preset> always has the format: <dd>:<hh>:<mm>:<ss> where the delimiter is always: <trigger expression> is the only variable length section of this syntax. It can have the form: <Alias name> for Boolean or data change evaluation (trigger T,F,t,f,c) Where Alias name is any symbolic Alias name used by the Sequencer.	Read-Only

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Attribute	Description	Run-time access
.Current.StepConditionState	Shows the step condition evaluation result. TRUE if the step condition is TRUE else FALSE.	Read-Only
.Current.StepDuration	Shows the elapsed time since the start of the current step.	Read-Only
.Current.StepName	Shows the name of the currently active step. Empty string if no step is active.	Read-Only
.Current.StepStartTime	Shows the time that the Sequencer Object entered the current step. Resets to the current time when step number changes or the current step is restarted.	Read-Only
.Current.StepTimeRemaining	Shows the remaining time of the step timer.	Read-Only
.Current.StepTimerRunning	Shows if the Step timer is running.	Read-Only
.Current.StepWriteOnExit	Shows if WriteOnExit is set for the Step Condition of the current step TRUE: Outputs are written before step transition. FALSE: Outputs are not written before step transition.	Read-Only

Program attributes

The following table lists the Sequencer object attributes associated with the Sequencer program at run time:

Attribute	Description	Run-time access
.PrgAliasCnt	The total number of Aliases contained in the Sequencer program.	Read-Only
.PrgAliasNames	The string array list of the Sequencer Object Alias names.	Read-Only
.PrgComment	Stores an optional Sequencer program comment.	Read-Only

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Attribute	Description	Run-time access
.PrgFileMatch	Set to False when the user has made a change to the program (step program, Alias configuration or settings) at run time but the program has not been saved to a file. This flag is also set to False when loading an XML file that does not contain all 3 sections (step program, Alias configuration, settings) so that existing sections are preserved or when loading an XML file into sections that have locked attributes.	Read-Only
.PrgFileName	The file name and path of the most-recently (successfully) loaded or saved file name.	Read-Only
.PrgFileNameToLoadSave	Stores the file name that the user specifies for loading and saving.	Supervisory, User
.PrgName	Displays the optional Sequencer program name.	Read-Only
.PrgReferenceNames	The string array list of the attribute references that are mapped to the Sequencer Object Aliases. The array is in the same sequence as the Alias names array.	Read-Only
.PrgSeqConfig	1. Allows user to update PrgStepProgram, PrgAliasConfig or Settings or any combination of them by specifying the corresponding section in the XML data set to this attribute. 2. The actual update takes place once PrgSeqConfigCmd is set to Update. 3. After Update is processed successfully, the value in the PrgSeqConfig is cleared.	Supervisory, User

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Attribute	Description	Run-time access
.PrgSeqConfigCmd	Set at run time to perform file IO or update the attribute PrgStepProgram, PrgAliasConfig or Settings or any combination of them. Feedback is provided by the PrgSeqConfigStatus attribute. "Save" - save the Sequencer program XML configuration to the file which is specified in PrgFileNameToLoadSave (the filename from where it was loaded or where it was successfully saved to in a previous save as operation). This operation fails if PrgFileNameToLoadSave is empty or specifies an invalid or read only file. "Load" - load the Sequencer XML data from the file specified in PrgFileNameToLoadSave. The Sequencer object loads whatever data is found in the XML file: Step Program data (steps and output values), Alias data (Alias names and attribute references) and Settings. If any of these sections are missing, the corresponding currently active data section remains unchanged. This allows replacing individual sections only (like loading a step program without modifying the existing IO configuration). "Update" - similar to except that the data is coming from PrgSeqConfig attribute instead of a file. Run time: Performs configuration operations and reflects the result in the attribute PrgSeqConfigStatus. Rejects with an Operational Error any attempt to issue a PrgSeqConfigCmd while PrgSeqConfigStatus is other than Ready or Error.	Supervisory, User

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Attribute	Description	Run-time access
.PrgSeqConfigStatus	Shows the result of the Save, Load or Update initiated by the PrgSeqConfigCmd. • 1 - "Ready" • 2 - "Error" • 3 - "Loading" • 4 - "Saving" • 5 - "Updating"	Read-Only
.PrgSeqConfigStatus.Code	Shows the result of the Save/Load/ Update initiated by the PrgSeqConfigCmd Possible values: • 0 - The action completed successfully, no problem was detected • Severity 1000 (Warning) - reflects skipping update on some attributes due to lock during Load or Update • Severity 2000 (Critical) - means error and the whole Load, Update or Save is rejected	Read-Only
.PrgSeqConfigStatus.Desc	Corresponds to PrgSeqConfigStatus.Code Includes Step Name and Step # when relevant. Includes Alias Name and # when relevant. It is empty when PrgSeqConfigStatus.Code = 0	Read-Only
.PrgSeqConfigStatus.Error	Shows the result of the Save/Load/ Update initiated by the PrgSeqConfigCmd It is set to True when PrgSeqConfigStatus.Code>=2000	Read-Only

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Attribute	Description	Run-time access
.PrgStepCnt	Shows the total number of steps contained in the Sequencer object.	Read-Only
.PrgStepInitial	Shows the starting step name of the Sequencer Object when deploying to run time or when reset.	Read-Only
.PrgStepInitialCmd	Allows run-time modification of the initial step name. Run time: Replaces initial step name in the Sequencer step program attribute and updates PrgStepInitial. The set is rejected with Operational error if PrgStepProgram attribute is locked. Validates the string value. A valid initial step string value is either an empty string or any existing step name in the step program. If this initial step name is empty the Sequencer starts on the first step in the sequence.	Supervisory, User
.PrgStepFinal	Allows run-time modification of the final step name.	Read-Only
.PrgStepFinalCmd	Allows run-time modification of the final step name. Run time: Replaces final step name in the Sequencer step program attribute and updates PrgStepFinal. The set is rejected with Operational error if PrgStepProgram attribute is locked. Validates the string value. A valid final step string value is either an empty string or any existing step name in the step program.	Supervisory, User
.PrgStepNames	Shows the string array list of the Sequencer Object step names.	Read-Only
.ProgramError	TRUE: In case of an Program Error. FALSE: No Program Error detected.	Read-Only

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Attribute	Description	Run-time access
.ProgramErrorDesc	Description of the Program Error.	Read-Only

Selected attributes

The following table lists all Sequencer object attributes associated with the selected step at run time:

Attribute	Description	Run-time access
.Selected.JumpConditionConfig	Represents the string representation of the jump condition configuration. For a full description, read the section XML roots and sub-roots. <condition type>|<timer preset>|<trigger expression> Where: <condition type> consists of 3 characters indicating the condition, trigger, and timer type: <Timer preset> always has the format: <dd>:<hh>:<mm>:<ss> <trigger expression> is the only variable length section of this syntax. It can have the form: <Alias> for Boolean or data change evaluation (trigger T,F,t,f,c) Where Alias is any symbolic Alias used by the Sequencer.	Read-Only
.Selected.JumpTimerPreset	Allows run-time modification of the jump timer preset. Run time: Replaces timer preset of the selected step with poked value if valid, updates Selected.JumpConditionCfg attribute. If Current = Selected, the new value is reflected in the calculation of the Current.JumpTimeRemaining attribute.	Supervisory, User

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Attribute	Description	Run-time access
.Selected.JumpToStepName	Allows run-time modification of the destination step that the Sequencer Object jumps to on jump transition in the selected step. If Current = Selected the new value should be reflected in the Current.JumpToStepName attribute. The set is rejected with a Config Error if the new value does not match an existing step of the step program.	Supervisory, User
.Selected.JumpWriteOnExit	Shows if WriteOnExit is set for the Jump Condition of the selected step. TRUE: Outputs are processed before jump transition. FALSE: Outputs are not processed before jump transition. Run time: Replaces jump WriteOnExit of the selected step with poked value, updates Selected.JumpConditionCfg attribute. If Current = Selected then Current.JumpWriteOnExit and Current.JumpConditionCfg are also updated.	Supervisory, User
.Selected.OnEntryAliasNames	OnEntry Output names corresponding to the Selected.OnExitOutputCnt and Selected.StepNum	Read-Only
.Selected.OnEntryOutputCnt	The number of outputs in the OnEntry Alias and value arrays of the selected step.	Read-Only

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Attribute	Description	Run-time access
.Selected.OnEntryOutputValues	Shows and allows modification of OnEntry Output values that correspond to the selected step. Run time: Invalid values lead to an output error later when writing to the output.	Supervisory, User
.Selected.OnExitAliasNames	OnExit Output names corresponding to the selected step.	Read-Only
.Selected.OnExitOutputCnt	The number of outputs in the OnExit Alias and value arrays of the selected step.	Read-Only
.Selected.OnExitOutputValues	Shows and allows modification of OnExit Output values corresponding to the selected step. If Current = Selected then the new value(s) are reflected in the Current.OnExitOutput values. **Note:** The size of the array cannot be changed, only the existing values. Run time: Invalid values lead to an output error later when writing to the output.	Supervisory, User

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Attribute	Description	Run-time access
.Selected.StepConditionConfig	Represents the string representation of the step condition configuration. For a full description, read the section XML roots and sub-roots. <condition type>|<timer preset>|<trigger expression> Where: <condition type> consists of 3 characters indicating the condition, trigger and timer type: <Timer preset> always has the format: <dd>:<hh>:<mm>:<ss> <trigger expression> is the only variable length section of this syntax. It can have the form: <Alias> for Boolean or data change evaluation (trigger T,F,t,f,c) Where Alias is any symbolic Alias used by the Sequencer.	Read-Only
.Selected.StepName	Represents the Step Name of the selected step. This attribute can be set at run time and behaves like .Selected.StepNum.	Supervisory, User
.Selected.StepNum	Run-time configurable Step Number value. Selected.StepNum is used for run-time (esp. HMI) editing of the selected step number parameters. Run time: Value cannot be less than one and cannot exceed the StepCnt. RTError is generated when a set attempt is made on an out of range value. When a value is set to Selected.StepNum update all Selected.* attribute values.	Supervisory, User

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Attribute	Description	Run-time access
.Selected.StepTimerPreset	Allows run-time modification of the step timer preset. Run time: Replaces timer preset of the selected step with poked value if valid, updates Selected.StepConditionCfg attribute. If Current = Selected, the new value is reflected in the calculation of the Current.StepRemaining attribute.	Supervisory, User
.Selected.StepWriteOnExit	Shows if WriteOnExit is set for the Step Condition of the selected step. TRUE: Outputs are evaluated before step transition. FALSE: Outputs are not evaluated before step transition. Run time: Replaces step WriteOnExit of the selected step with poked value, updates Selected.StepConditionCfg attribute. If Current = Selected then Current.StepWriteOnExit and Current.StepConditionCfg are also updated.	Supervisory, User

Alarm attributes

If an alarm attribute is enabled, then the following additional alarm attributes become available. None of these attributes can be alarmed.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Attribute	Description
<Attribute>.Acked	Used to specify whether an alarm has been acknowledged. This attribute is updated when the AckMsg attribute is set. This attribute is always set to FALSE when a new alarm condition is detected (when the InAlarm attribute changes from FALSE to TRUE).
<Attribute>.AckMsg	The operator comment at the time the alarm is acknowledged. Any received text is stored, and the Acked attribute is set to TRUE. Also, the TimeAlarmAcked attribute is set to the current time. The maximum length is 256 characters.
<Attribute>.Category	The category of the alarm. The label of each alarm category is fixed.
<Attribute>.DescAttrName	The name of the attribute (within the same object) to be used as the alarm message. The attribute you specify must be of type String or InternationalizedString, with a maximum string length of 329 characters. If no attribute is specified, the short description of the object is used as the alarm message.
<Attribute>.InAlarm	The alarm state. This is exactly the same as the attribute in the host primitive that represents the alarm condition, except when the alarm state is disabled, in which case, InAlarm is set to Off, regardless of the actual condition state. The quality is set during execute to the quality of the attribute, except when the alarm is disabled, in which case the quality is always GOOD.
<Attribute>.Priority	The value for the urgency of the alarm. Valid values are 1 through 999, with 1 being the most urgent.
<Attribute>.TimeAlarmAcked	The timestamp indicating the last time this alarm was acknowledged. The date format reflects the current locale setting for the operating system.
<Attribute>.TimeAlarmOff	The timestamp indicating the last time this alarm (as represented by the InAlarm attribute) went off. The date format reflects the current locale setting for the operating system.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Attribute	Description
<Attribute>.TimeAlarmOn	The timestamp indicating the last time this alarm (as represented by the InAlarm attribute) went on. The date format reflects the current locale setting for the operating system.

History attributes

If a historization attribute is enabled, then the following additional history-related attributes become available. None of these attributes can be alarmed.

Attribute	Description
<Attribute>.ForceStoragePeriod	The time interval, in milliseconds, at which the value must be stored, even if the value has not changed. A value of 0 disables this feature. As an example, a setting of 3600000 indicates the value must be stored once per hour (measured from the time the object was last put onscan). If this value is less than the scan period of the host object, forced storage occurs every scan period (effectively equivalent to setting a value deadband of 0).
<Attribute>.TrendHi	The default top of the trend scale for clients. This value must be greater than or equal to the low value for the trend. If this value is changed at run time, the maximum engineering unit change is not reflected in the historian until you redeploy the object. This attribute only applies to numeric data types.
<Attribute>.TrendLo	The default bottom of the trend scale for clients. This value must be less than or equal to the high value for the trend. If this value is changed at run time, the minimum engineering unit change is not reflected in the historian until you redeploy the object. This attribute only applies to numeric data types.
<Attribute>.ValueDeadBand	The amount in engineering units by which a value referenced by the attribute must change to force a storage. A value of 0 means it is unused, and thus any value change forces a storage. Also, a quality change in the attribute always forces a storage regardless of this setting. This attribute only applies to numeric data types.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Sequencer program XML schema

This section describes the structure and syntax of the XML strings that are passed to attributes to control the Sequencer program at run time.

General information

The Sequencer XML file is case-sensitive for keywords and parameters. For example, passing the Initial Command value "hold" instead of "Hold" won't work.

XML roots and sub-roots

All Sequencer XML documents have the common root <SEQ_PRG> with the following sub-roots:

<STEPS/>	contains the step program
<ALIASES/>	contains the Alias configuration
<SETTINGS/>	contains global settings like "halt on condition errors", initial step and final step

About steps

Steps are located in the <STEPS/> sub-root each with a <STEP> entry. Each step has following XML attributes:

NUMBER	specifies the step number (integer)
NAME	specifies the step name (string)
STEPCONDITION	specifies the step condition (string) - see below for details
JUMPCONDITION	specifies the jump condition (string) - see below for details
JUMPTOSTEPNAME	specifies the step to jump to (string)

Each step has two sub-sections <ONENTRY> and <ONEXIT> containing <OUT> keys to define output value assignments.

Important: If you are passing literal string (or date, time or elapsed time) values as outputs you need to enclose these with the ampersand notation of double-quotes, which is: "

Examples:

- <OUT name="MachineInstruction" value=""Start""></OUT>
  which passes the literal value Start to the Alias MachineInstruction for a given step.
- <OUT name="MachineInstruction" value="Start"></OUT>
  which passes the value of the Alias named Start to the Alias MachineInstruction for a given step.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

- <OUT name="MachineInstruction" value=""Joe says "Start"""></OUT> which passes the literal value Joe says "Start" to the Alias MachineInstruction for a given step.

Step/jump condition syntax

Conditions are fully described by a string containing the condition type; timer preset, and trigger expression:
<condition type><onexit output flag><timer preset>|<trigger expression>

Condition type

Condition type consists of 3 characters:	
111	Always TRUE
000	Always FALSE
<T>--	Trigger only
--<I>	Timer only
<T>A<I>	Trigger AND Timer
<T>O<I>	Trigger OR Timer
<T>D<D>	Trigger Timer Delay

where the trigger character <T> can be any of the following:

T	Evaluate for TRUE (While TRUE)
F	Evaluate for FALSE (While FALSE)
t	Evaluate for transition to TRUE (On TRUE)
f	Evaluate for transition to FALSE (On FALSE)
c	Evaluate for data change

and the timer character <I> can be any of the following:

S	Simple timer
R	Retentive timer
N	Non-retentive timer
M	Cyclic timer month with offset (days, hours, minutes, seconds)
W	Cyclic timer week with offset (days, hours, minutes, seconds)

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

d	Cyclic timer day with offset (hours, minutes, seconds)
h	Cyclic timer hour with offset (minutes, seconds)
m	Cyclic timer minute with offset (seconds)

OnExit output flag

OnExit Output Flag consists of one character:	
!	all OnExit outputs are set on transition
|	no OnExit outputs are set on transition

Timer preset

<timer preset> always has the format <dd>:<hh>:<mm>:<ss> where:

dd	two digits offset in days (with leading zeros)
hh	two digits offset in hours (with leading zeros)
mm	two digits offset in minutes (with leading zeros)
ss	two digits offset in seconds (with leading zeros)
:	delimiter

Trigger expression

Trigger expression is the only variable length section of this syntax and is the symbolic Alias name.

Examples of conditions

- stepcondition="T--!00:00:00:00|TankFull" - immediate transition to the next step when the attribute associated with the Alias named TankFull is true.
- stepcondition="TDS!00:00:00:05|TankEmpty" - 5 seconds delayed transition to the next step after the attribute associated with the Alias named TankEmpty is true.
- jumpcondition="T--|00:00:00:00|Contaminated" - jump immediately to the step defined in the parameter jumptostepname (not shown here) as soon as the attribute associated with the Alias Contaminated is true.
- stepcondition="111!00:00:00:00|" - transition to the next step regardless of any trigger or timer.

Alias configuration

Aliases (Inputs/Outputs) are located in the <ALIASES> sub-root each with a <ALIAS> entry. Each Output has the

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

following XML attribute:

NAME	logical name (string)
ATTR	attribute name (string)

Settings

Settings are located in the <SETTINGS> sub-root each with a dedicated property:

<InitialCommand>	contains the initial command (Start, Stop, Hold, SingleStep)
<HaltOnConditionError>	specifies if program processing should halt on error condition (1) or not (0)
<HaltOnOutputError>	specifies if program processing should halt on output error (1) or not (0)
<ResumeAfterFailover>	specifies if processing should resume after a failure (1) or not (0).

Caution: The properties are case-sensitive. For example: If you specify <initialcommand> instead of <InitialCommand> the XML property is not recognized.

Example XML

The following shows an example of a Sequencer XML file:

```
<SEQ_PRG>
<STEPS name="Sample Sequencer Program" comment="Sample Sequencer Program demonstrating steps with
different conditions" StepInitial="Step_NOP" StepFinal="TestAndExit">
<STEP name="Step_NOP" stepcondition="111!00:00:00:00|" jumpcondition="000|00:00:00:00|"
jumptostepname="">
<ONENTRY>
<OUT name="Output1" value="111" />
<OUT name="Output2" value="112" />
</ONENTRY>
<ONEXIT>
<OUT name="OutputString" value="&quot;Hello World&quot;" />
</ONEXIT>
</STEP>
<STEP name="Trigger" stepcondition="T--!00:00:00:00|Input1" jumpcondition="000|00:00:00:00|"
jumptostepname="">
<ONENTRY>
<OUT name="Output1" value="0" />
</ONENTRY>
<ONEXIT>
<OUT name="Output2" value="222" />
<OUT name="OutputString" value="&quot;10/24/2006 10:30:00 AM&quot;" />
</ONEXIT>
```

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

```
</STEP>
<STEP name="Timer" stepcondition="--S!00:00:00:15|" jumpcondition="000|00:00:00:00|"
jumptostepname="">
<ONENTRY>
<OUT name="Output1" value="10" />
</ONENTRY>
<ONEXIT>
<OUT name="OutputString" value="&amp;quot;00:01:30.000&amp;quot;" />
</ONEXIT>
</STEP>
<STEP name="TestAndExit" stepcondition="--S!00:00:00:20|" jumpcondition="TDR|00:00:00:12|Input1"
jumptostepname="Step_NOP">
<ONENTRY>
<OUT name="Output1" value="0" />
<OUT name="Output2" value="15" />
</ONENTRY>
</STEP>
</STEPS>
<ALIASES>
<ALIAS name="Output1" attr="Motor.Speed" />
<ALIAS name="Output2" attr="Filler.Test" />
<ALIAS name="OutputString" attr="Object.Description" />
<ALIAS name="Input1" attr="Valve.Open" />
</ALIASES>
<SETTINGS>
<InitialCommand value="Stop" />
<HaltOnConditionError value="0" />
<HaltOnOutputError value="0" />
<ResumeAfterFailover value="0" />
</SETTINGS>
</SEQ_PRG>
```

Sequencer state transition tables

This section shows the relationship between step program execution states and execution commands that you can write. The structure is from a current execution state point of view.

Terminology

The following terminology is used in the State Transition Tables:

- Current State - the current execution state of the step program. You can find out the current state by reading the attribute ExecutionState.
- Current Timer - the timer state of the current step before you change the execution state.
- Command - the command you write to the execution state command attribute (ExecutionStateCmd).
- Target State - the state of the step program execution after you have written the execution command to the attribute ExecutionStateCmd. This is your desired execution state.
- Target State Timer - after you write the execution command to the attribute ExecutionStateCmd, the timers act accordingly. They may restart, continue, or stop.
- Target Step - the step that becomes active after you write the execution command to the attribute

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

ExecutionStateCmd.

- Execution OnExit - informs you whether Sequencer writes values to the OnExit outputs (if configured) of the currently active step after you write the execution command.
- Execution OnEntry - informs you whether Sequencer writes values to the OnEntry outputs of the step that is active after the currently active step.

Current execution state: Running

The following table gives you an overview of how the state, timers, active step, and OnExit/OnEntry execution change depending on the execution state command you issue, when the current execution state is Running:

Current State: Running **Current Timer: Running** **Calculated attributes quality set to BAD: none**			
Command	**Target state** **Target state timer**	**Target step**	**Execute OnExit** **Execute OnEntry**
Stop	State: Stopped Timer: Stop	current	OnExit: NO OnEntry: NO
Reset	State: InitialCommand Timer: Restart	Initial step or first step	OnExit: NO OnEntry: YES (if InitialCommand <> Stop)
Advance	State: Running (StoppedComplete if last step) Timer: Restart	Next step (or current if last step)	OnExit: YES (if Step write on exit is true) OnEntry: YES (NO if last step)
SingleStep	State: RunningSingleStep Timer: Continue	current	OnExit: NO OnEntry: NO
Hold	State: Running -> RunningHeld Timer: Freeze at current value	Current or next step body	OnExit: NO (YES if next step body) OnEntry: NO (YES if next step body)
StepNumCmd/ StepNameCmd	State: Running Timer: Restart	StepNum/StepName	OnExit: YES (if Jump write on exit is true) OnEntry: YES

Current execution state: RunningSingleStep

The following table gives you an overview of how the state, timers, active step, and OnExit/OnEntry execution

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

change depending on the execution state command you issue, when the current execution state is RunningSingleStep:

Current State: RunningSingleStep Current Timer: Running Calculated attributes quality set to BAD: none			
Command	**Target state** **Target state timer**	**Target step**	**Execute OnExit** **Execute OnEntry**
Start	State: Running Timer: Continue	Current	OnExit: NO OnEntry: NO
Stop	State: Stopped Timer: Stopped	Current	OnExit:NO OnEntry: NO
Reset	State: InitialCommand Timer: Restart	Initial step or first step	OnExit: NO OnEntry: YES (if InitialCommand <> Stop)
Advance	State: RunningSingleStep (StoppedComplete if last step) Timer: Restart	Next step (or current if last step)	OnExit: YES (if Step write on exit is true) OnEntry: YES (NO if last step)
Hold	State: Running -> RunningHeld Timer: Freeze at current value	Current or next step body	OnExit: NO (YES, if next step body) OnEntry: NO (YES, if next step body)
StepNumCmd/ StepNameCmd	State: RunningSingleStep Timer: Restart	StepNum/StepName	OnExit: YES (if Jump write on exit is true) OnEntry: YES

Current execution state: SingleStepTransitionReady

The following table gives you an overview of how the state, timers, active step, and OnExit/OnEntry execution change depending on the execution state command you issue, when the current execution state is SingleStepTransitionReady:

Current State: SingleStepTransitionReady Current Timer: Running Calculated attributes quality set to BAD: none			
Command	**Target state** **Target state timer**	**Target step**	**Execute OnExit** **Execute OnEntry**

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Start	State: Running (StopComplete if in the Final step) Timer: Restart	Next Step (Current step if it was the final step)	OnExit: NO OnEntry: YES (next Step) (NO when this was final step)
Stop	State:Stopped Timer: Stopped	Current	OnExit:NO OnEntry: NO
Reset	State: InitialCommand Timer: Restart	Initial step or first step	OnExit: NO OnEntry: YES (if InitialCommand <> Stop)
Advance	State: RunningSingleStep (StopComplete if in the Final step) Timer: Restart	Next step (Current step if it was the final step)	OnExit: NO (OnExit has already been executed) OnEntry: YES (NO when this was the final step)
Confirm	State: RunningSingleStep (StopComplete if in the Final step) Timer: Restart	Next Step (Current step if it was the final step)	OnExit: NO OnEntry: YES (NO when this was the final step)
Hold	State: Running -> RunningHeld Timer: Restart and freeze when reaching RunningHeld	Next step	OnExit: NO OnEntry: YES
StepNumCmd/ StepNameCmd	State: RunningSingleStep Timer: Restart	StepNum/StepName	OnExit: NO (OnExit has already been executed) OnEntry: YES

Current execution state: RunningHeld

The following table gives you an overview of how the state, timers, active step, and OnExit/OnEntry execution change depending on the execution state command you issue, when the current execution state is RunningHeld:

Current State: RunningHeld **Current Timer: Frozen** **Calculated attributes quality set to BAD: none**			
Command	**Target state** **Target state timer**	**Target step**	**Execute OnExit** **Execute OnEntry**

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Start	State: Running Timer: Continue	Current	OnExit: NO OnEntry: NO
Stop	State: Stopped Timer: Stopped	Current	OnExit: NO OnEntry: NO
Reset	State: InitialCommand Timer: Restart	Initial step or first step	OnExit: NO OnEntry: YES (if InitialCommand <> Stop)
Advance	State: Running Held (StoppedComplete if last step) Timer: Restart	Next step (or current if Final step)	OnExit: YES (if Step write on exit is true) (NO, if last step) OnEntry: YES (NO, if last step)
SingleStep	State: RunningSingleStep Timer: Continue	Current	OnExit: NO OnEntry: NO
Resume	State: State before Hold command Timer: Continue in Timer state before Hold command	Current	OnExit: NO OnEntry: NO
StepNumCmd/ StepNameCmd	State: Running Held Timer: Restart	StepNum/StepName	OnExit: YES (if Jump write on exit is true) OnEntry: YES

Current execution state: Stopped

The following table gives you an overview of how the state, timers, active step and OnExit/OnEntry execution changes depending on the execution state command you issue, when the current execution state is Stopped:

Current State: Stopped **Current Timer: Stopped** **Calculated attributes quality set to BAD: all dynamic attributes**			
Command	**Target state** **Target state timer**	**Target step**	**Execute OnExit** **Execute OnEntry**
Start	State: Running Timer: Restart	Current	OnExit: NO OnEntry: YES

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Reset	State: InitialCommand Timer: Restart	Initial step or first step	OnExit: NO OnEntry: YES (if InitialCommand <> Stop)
Advance	State: Stopped Timer: Stopped	Next step (or current if last step)	OnExit: NO OnEntry: NO
SingleStep	State: RunningSingleStep Timer: Restart	Current	OnExit: NO OnEntry: YES
Hold	State: Running -> RunningHeld Timer: Restart and freeze when reaching RunningHeld	Current	OnExit: NO OnEntry: YES
StepNumCmd/ StepNameCmd	State: Stopped Timer: Stopped	StepNum/StepName	OnExit: NO OnEntry: NO

Current execution state: StoppedComplete

The following table gives you an overview of how the state, timers, active step and OnExit/OnEntry execution changes depending on the execution state command you issue, when the current execution state is StoppedComplete:

Current State: StoppedComplete **Current Timer: Stopped** **Calculated attributes quality set to BAD: all dynamic attributes**			
Command	**Target state** **Target state timer**	**Target step**	**Execute OnExit** **Execute OnEntry**
Reset	State: InitialCommand Timer: Restart	Initial step or first step	OnExit: NO OnEntry: YES (if InitialCommand <> Stop)
StepNumCmd/ StepNameCmd	State: Stopped Timer: Stopped	StepNum/StepName	OnExit: NO OnEntry: NO

Current execution state: StoppedError

The following table gives you an overview of how the state, timers, active step, and OnExit/OnEntry execution change depending on the execution state command you issue, when the current execution state is StoppedError:

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Current State: StoppedError Current Timer: Stopped Calculated attributes quality set to BAD: all dynamic attributes			
Command	**Target state** **Target state timer**	**Target step**	**Execute OnExit** **Execute OnEntry**
Start	State: Running Timer: Restart	Current	OnExit: NO OnEntry: YES
Reset	State: InitialCommand Timer: Restart	Initial step or first step	OnExit: NO OnEntry: YES (if InitialCommand <> Stop)
Advance	State: Stopped Timer: Stopped	Next step (or current if last step)	OnExit: NO OnEntry: NO
SingleStep	State:RunningSingleStep Timer: Restart	Current	OnExit: NO OnEntry: YES
Hold	State: Running -> RunningHeld Timer: Restart and freeze when reaching RunningHeld	Current	OnExit: NO OnEntry: YES
StepNumCmd/ StepNameCmd	State: Stopped Timer: Stopped	StepNum	OnExit: NO OnEntry: NO

Error codes for run-time updates

The following table shows an overview of error codes and descriptions that result from updating the step program, alias configuration and/or settings by the PrgSeqConfig attribute at run time. The attribute group PrgSeqConfigStatus contains error information, such as code, error and description.

Code	Key	Description
2000	FailedToParseXML	Failed to parse XML
2001	FailedToSaveXML	Failed to save XML
2002	InvalidXMLFile	Invalid XML file
2003	InvalidXMLFormat	Invalid XML format
2004	InvalidStepProgramXMLData	Invalid step program XML data

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

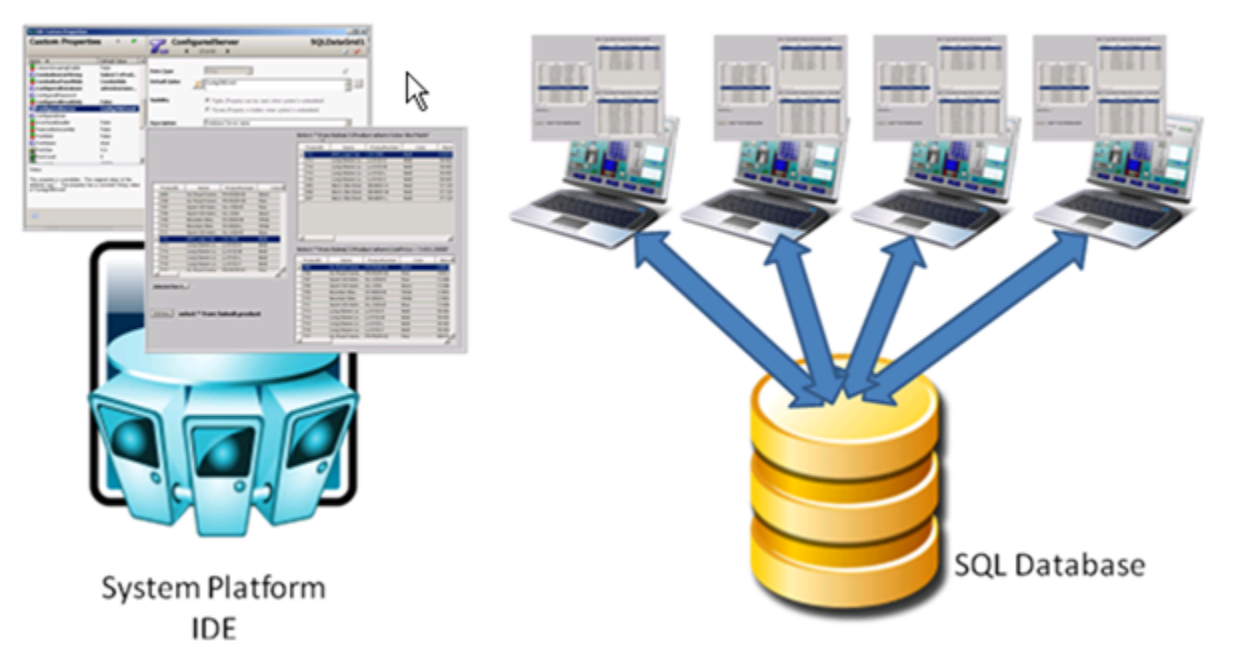
Code	Key	Description
2005	InvalidAliasConfigXMLData	Invalid Alias configuration XML data
2006	InvalidStepConfiguration	Invalid step configuration
2007	InvalidCondition	Invalid condition
2008	MissingStepName	Missing step name
2009	ConditionCodeTooShort	Condition code too short
2010	InvalidStepName	Invalid step name
2011	DuplicateStepName	Duplicate step name
2012	InvalidJumpToStepName	Invalid JumpTo step name
2013	MissingJumpToStepName	Missing JumpTo step name
2014	MissingStepCondition	Missing step condition
2015	MissingStepTrigger	Missing step trigger
2016	MissingTrigger	Missing trigger
2017	TriggerNotConfigured	Trigger not configured
2018	JumpTriggerNotConfigured	Jump trigger not configured
2019	InvalidTimerConfiguration	Invalid timer configuration
2020	InvalidTimerCode	Invalid timer code
2021	InvalidInitialStepName	Invalid initial step name
2022	InvalidFinalStepName	Invalid final step name
2023	InvalidAliasConfiguration	Invalid Alias configuration
2024	InvalidAliasName	Invalid Alias name
2025	DuplicateAliasName	Duplicate Alias name
2026	InvalidIOReference	Invalid Alias Reference
2027	OnEntryExitAliasNotConfigured	OnEntry/OnExit Alias not configured
2028	OnEntryExitValueAliasNotConfigured	OnEntry/OnExit Alias values not configured

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

SQLDataGrid Graphic

The SQLDataGrid is a graphic that enables you to interact with data stored in SQL databases. You can use the SQLDataGrid graphic in an InTouch application that you create in a published or managed mode.

The following picture shows the main components associated with the SQLDataGrid.



You can perform the following activities with the SQLDataGrid graphic:

- Insert or modify data in selected database tables and views.
- Delete rows from a database table (select the row and press the **Delete** key).
- Select data from existing tables and views.
- Run a custom SQL query.

This section provides the following information:

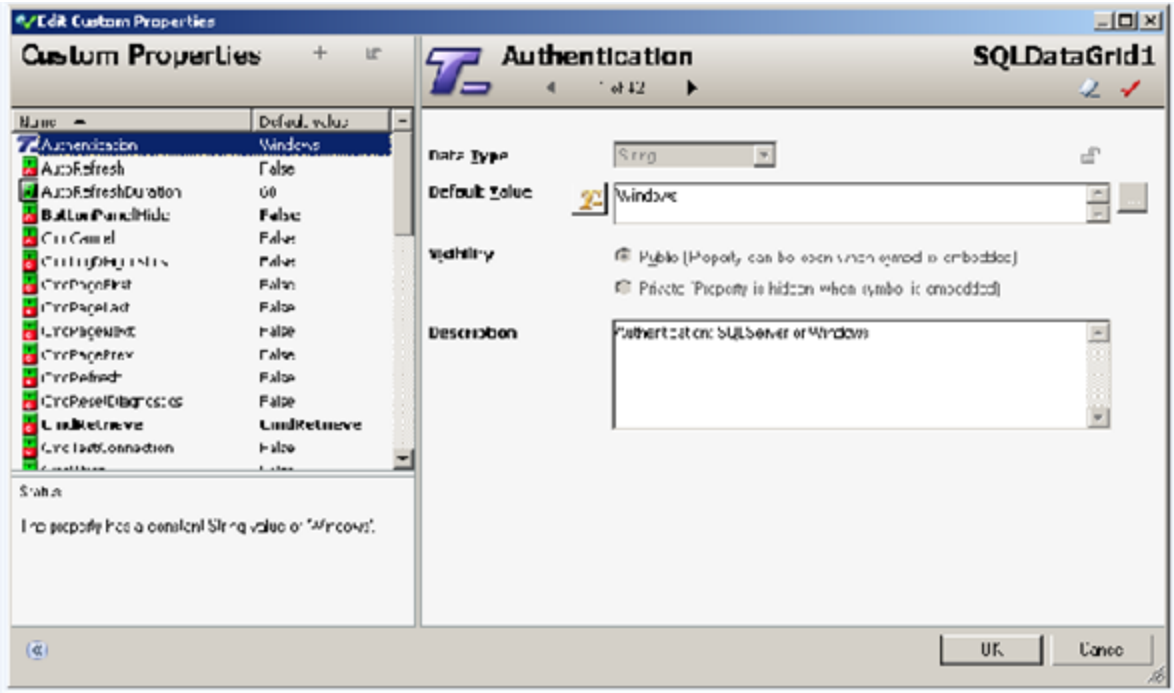
- The minimum configuration required by the SQLDataGrid graphic to retrieve data from a SQL database.
- The set of custom properties that must be configured to authenticate users and connect to the database.

A list of the custom properties that define the parameters associated with a specific database and query to be retrieved.

The SQLDataGrid graphic

The SQLDataGrid graphic appears in the **Visualization** folder of the System Platform IDE. Embed the graphic in an existing window or a new window. When you double-click the graphic, the **Edit Custom Properties** dialog box appears.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.



For detailed information on working with graphics, see the *ArchestrA and InTouch Integration Guide*.

Configure authentication modes

This section describes how to use the custom properties of the SQLDataGrid graphic to authenticate a user to the database. You can assign static values to these custom properties or link them to attributes or tag names in your application.

Note: The value fields are not case sensitive.

To configure an authentication mode for the SQLDataGrid graphic

1. In the **Custom Properties** pane, select **Authentication**.
2. In the **Default Value** box, type the name of a valid security type. Acceptable values are:
   - Windows
   - SQL Server

   A more detailed description of these authentication modes follows this procedure.
3. In the **Custom Properties** pane, select **ConfiguredUser**.
4. In the **Default Value** box, type a user name.
5. In the **Custom Properties** pane, select **ConfiguredPassword**.
6. In the **Default Value** box, type the password associated with the user name.

Windows authentication

If you use Windows as a value for the **Authentication** custom property and do not designate custom properties for a domain, user, or password, the SQLDataGrid graphic uses the credentials of the user who is currently logged on to the node running InTouch.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

If you select Windows as a value for the **Authentication** custom property and designate custom properties for a domain, user, and password, the SQLDataGrid graphic uses the specified Windows domain name, user name, and password to connect to the SQL Server database.

SQL Server

This authentication mode uses the specified SQL Server user name and password to connect to the SQL Server database.

Configure the database connection

This section describes the procedure to configure the custom properties required to establish a connection to a specific database and its associated data. You can assign static values to these custom properties or link them to attributes or tag names in your application.

Note: The value fields are not case sensitive.

If you are using a non-default port for the SQL Server database, see the SQLData Object help file for configuration information.

To configure the database connection

1. In the **Custom Properties** pane, select **ConfiguredServer**.
2. In the **Default Value** box, type the network name of the computer hosting the database.
3. In the **Custom Properties** pane, select **ConfiguredDatabase**.
4. In the **Default Value** box, type the name of the target database.
5. In the **Custom Properties** pane, select **SQLQuery**.
6. In the **Default Value** box, type the name of the target table, view or query.

To populate the SQLDataGrid, click **Runtime**, and then click **Retrieve**.

Configure SQLDataGrid graphic custom properties

This section lists custom properties in functional groups. For a detailed description of each custom property, see Custom Properties for the SQLDataGrid Graphic.

The default run-time window shows the complete graphic. However, you can hide the interface controls and display only the grid.

You can use custom properties to display additional information about the status of the query, the dataset, or to provide interactive buttons that initiate specific actions

In addition to the table, view, and query input, you can specify additional parameters by using custom properties and control them with scripting.

The following table lists the various custom properties you can insert in your scripts by functional groups.

Authentication	Authenticate users to databases	**Authentication**

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

		ConfiguredDomain
		ConfiguredPassword
		ConfiguredUser
Database Connectivity	Designate database connectivity functions	
		AutoRefresh
		AutoRefreshDuration
		CmdCancel
		CmdRefresh
		CmdRetrieve
		CmdTestConnection
		CmdWrite
		ComboBoxListString
		ConfiguredDatabase
		ConfiguredReadOnly
		ConfiguredServer
		RowCount
		RowLimit
		SelectedRowNumber
		SQLQuery
		SQLQueryCurrent
		SQLQueryPassThrough
		SQLQueryPending
Diagnostic	Perform diagnostic functions	
		CmdLogDiagnostics
		CmdResetDiagnostics
		ErrorPanelDisable
		StatusDesc

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

		StatusError
		StatusIsError
Appearance	Modify the appearance of the run-time SQLDataGrid that contains data from the database	
		ButtonPanelHide
		CmdPageFirst
		CmdPageLast
		CmdPageNext
		CmdPagePrev
		ColumnAggregateEnable
		ColumnFiltering
		ColumnGroupingEnable
		ComboBoxPanelHide
		FontBold
		FontName
		FontSize
		SupportThemes
		SuppressAuthenticationWarning
		WriteButtonHide

Scripts with the SQLDataGrid

When you use the **SQLDataGridUserCtl** to create a new graphic, you can script all the properties and events of the control.

Associate grid control actions with InTouch scripts and animations

You can hide the buttons of the SQLDataGrid graphic and trigger data retrieval from outside the grid interface. For example, you can configure a button to generate a dataset from an InTouch application.

To do this, use the **CmdRetrieve** property and associate it to an InTouch tag. The tag is linked to the InTouch button.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

To configure dataset retrieval from InTouch

1. In WindowMaker, open the InTouch window.
2. Double-click the SQLDataGrid graphic. The **Edit Custom Property** dialog box appears.
3. In the **Custom Properties** panel, select the **ButtonPanelHide** property.
4. In the **Default Value** field, type **True**.
5. Select the **CmdRetrieve** property.
6. In the **Default Value** field, select an InTouch tag by one of the following actions:
   a. Type a tag name. If the tag does not exist, you are prompted to define a new tag.
   b. Click the ellipsis to open the Tagname Dictionary and select a tag.

   For this example, type **CmdRetrieve** in the field.
7. Click **OK** and define the tag as a Memory Discrete tag.
8. When you finish defining the tag, close the Tagname Dictionary and click **OK** on the **Custom Properties** panel.
9. Create a button on your InTouch window. For this example, its caption is **Retrieve**.
10. Link the **CmdRetrieve** tag to the button.
11. Switch to run-time mode and click **Retrieve**.

Access data from a selected row in the SQLDataGrid

You can use the SQLDataGrid graphic to read the content of a selected row and associate it to attributes outside the graphic. You can configure scripts to read the data from a selected row and programmatically associate that data to tags in an InTouch application or to attributes in the Galaxy.

The following script example shows how to retrieve data from a selected row. Notice that the **SelectedRowData** property is available in the SQLDataGrid client control and not in the SQLDataGrid graphic.

```
dim rd1 as System.Data.DataRow;
rd1 = SQLDataGridUserCtrl1.SelectedRowData;
if(rd1 <> null) then
InTouch:Text9 = rd1.ItemArray[4].ToString();
'This assigns the 4th column value for the selected row, to the InTouch "Text9" tag.
endif;
```

The SQLDataGrid in runtime

During run time, you can select the database table, view, or query that will populate the SQLDataGrid graphic and change the appearance of the data.

You can use the run-time application to perform the following tasks:

- Move columns
- Group columns
- Aggregate column values
- Filter columns
- Sort columns

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

If you have the appropriate permissions, you can also make the following changes to the database:

- Change the order of columns in an existing row.
- Insert a row into a table.
- Update a row in a table.
- Delete a row from a table.

Note: You cannot update or delete a row in a table unless that table has a primary key defined. You can add new rows to a table without a primary key as long as that is the only action you are performing on the table.

View data

You can use the run-time version of the SQLDataGrid graphic to view data in the following formats:

- Data from a specific table
- Data from a specific view
- Data that is retrieved by a SQL query

You can resize the column width of any table, view, or query result.

To view specific data from a SQL query

1. In the **Table, View or Query** box, type a SQL query.
2. Click **Retrieve**. The appropriate dataset appears.

Populate the Table, View, or Query list

You can also populate the Table, View or Query box with items that you frequently use, so that you do not repeatedly have to type entries.

To populate the Table, View, or Query box

1. In the **Edit Custom Properties** dialog box, click **ComboBoxListString**.
2. In the **Default Value** box, type any of the following items, separated by semicolons (;):
   - Multiple table names
   - Multiple SQL statements
   - A combination of table names and SQL statements
3. Click **OK**.
4. When you go to run time, click the arrow to the right of the **Table, View or Query** box. Each item that you entered is displayed on a separate line.

Hide and show window issues

If you hide the InTouch window that contains the SQLDataGrid and then show it again, the last data values contained in the window might not appear. Use the following procedure to ensure that the data values always appear.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

To ensure that grid values appear when you hide and show the InTouch window

1. In WindowMaker, click **Special > Configure > WindowViewer**.
2. Under **WindowViewer Memory**, clear the check box **Always load windows from disk**.

Move columns

You can rearrange the order of the columns to meet your viewing needs.

To move columns

1. Click the heading of the column that you want to move.
2. Holding your mouse key down, drag the column heading to another position within the table.

Grouping Columns

The group-by function lets you sort data at more specific levels. This example shows a view that groups customer names and their business addresses.

Note: The ColumnGroupingEnable_dg custom property must be True before you can use this function.

To use the Group-by function

1. In the **Table, View or Query** box, type the name of a table or view or click the arrow to display a list of pre-determined items. For details about pre-determined items, see View Data.
2. Click **Group-by**. The **Drag a column header** area appears.

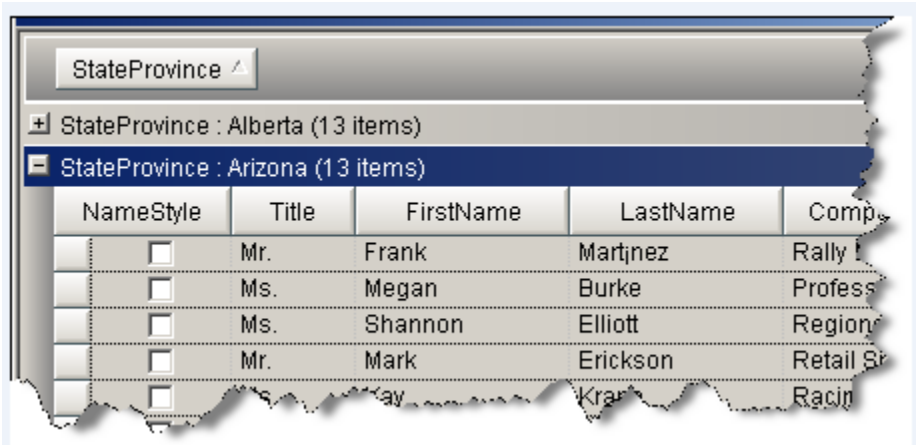


3. Drag a column heading to the upper area.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.



4. To view the data groups, press **Enter** to expand any of the **StateProvince** items.

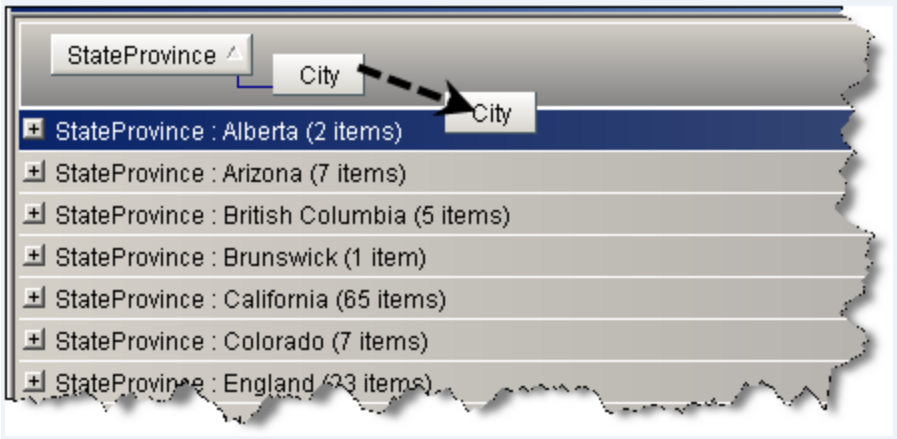


5. To see a more specific sorting of the data, drag another column heading to the upper sorting area.



6. If you want to sort the data at an even finer level, you can repeat steps 3 and 4 and drag another column heading to the sort area.
7. If you want to remove one of the column headings from the sorting area, drag it down to the data area.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.



8. To return the original dataset, click **Group-by**.

View aggregate values

You can perform a numeric operation on data within a single column of the table by clicking the **Aggregate** button.

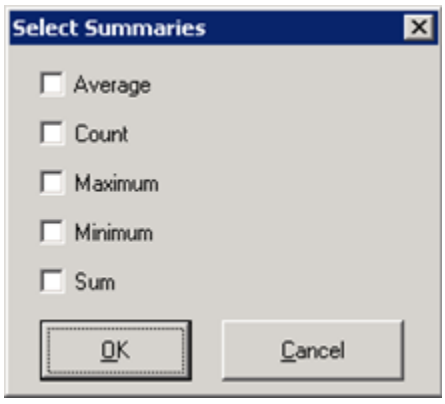
Note: Before you can use this function, the ColumnAggregateEnable custom property must be True.

You can perform the following functions on column data. Not all operations may be available for a column because of the data type.

- Average: Calculates the average number of items in the column.
- Count: Lists the number of rows in the column.
- Maximum: Lists the highest number in the column.
- Minimum: Lists the lowest number in the column.
- Sum: Adds the numbers in each row and provides a total.

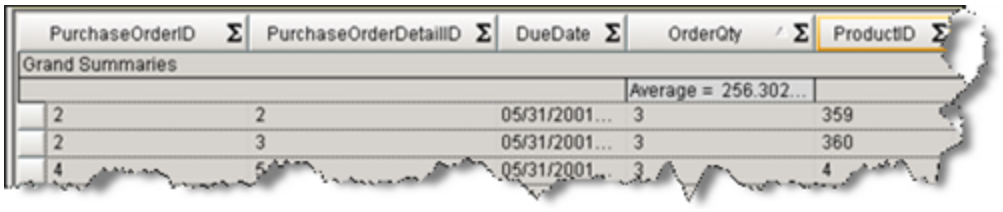
To view aggregate values

1. Click **Aggregate**. The summation sign (S) appears at the head of each column.
2. Click the summation sign at the top of the **OrderQty** column. The **Select Summaries** dialog box appears.



3. Select **Average** and click **OK**. The value appears just below the **Grand Summaries** row. You can re-size the column width to view the entire number This following figure shows the average value for **OrderQty**.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.



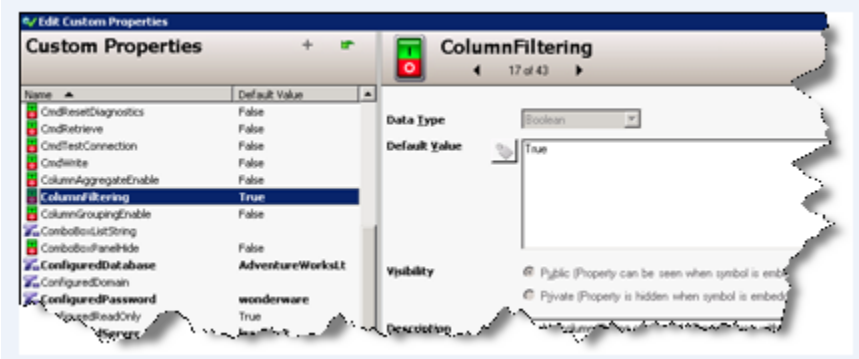
4. To hide the **Grand Summaries** row follow these steps:
   a. Click the summation sign.
   b. Clear the **Average** check box in the **Select Summaries** dialog box
   c. Click **OK**.
5. To hide the summation signs, click **Aggregate**.

Progressive column filtering

You can configure the SQLDataGrid to support progressive column filtering. Type the first letters of your search, and the matching values appear in the grid.

To configure and use progressive filtering

1. Switch your SQLDataGrid window to WindowMaker mode.
2. Double-click the graphic and select the custom property **ColumnFiltering**.
3. Set the **ColumnFiltering** default value to **True**.

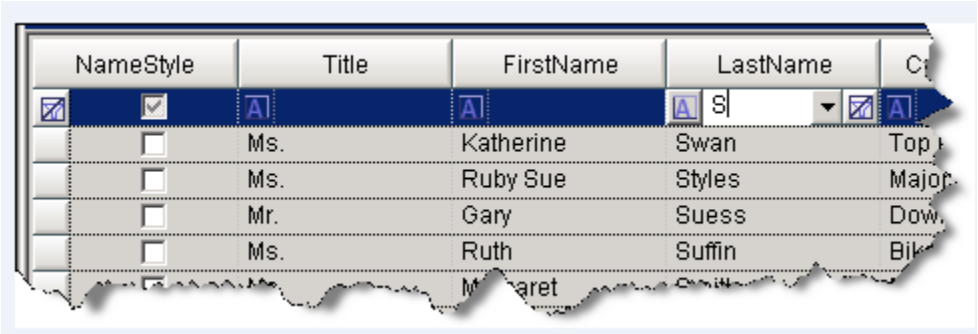


4. Click **OK** and run the InTouch application.
5. To return a dataset, click **Retrieve**. In this example, the dataset contains a table view of customer names and addresses.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

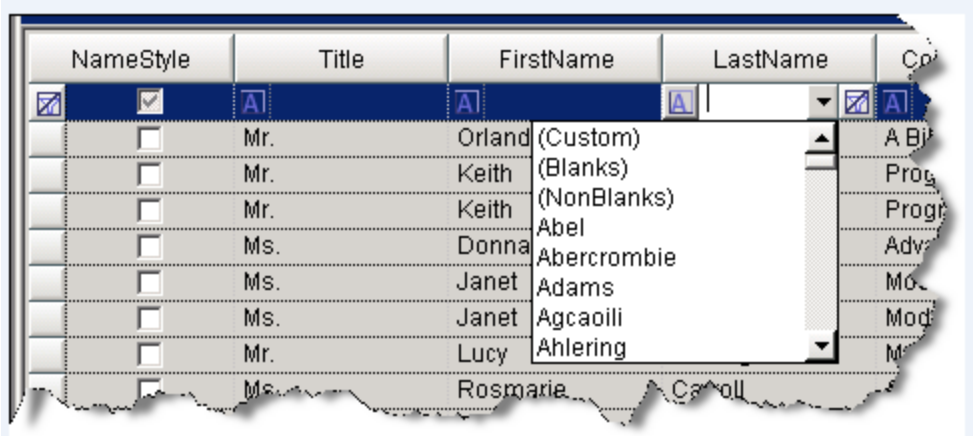


6. Move your cursor over the field below the **LastName** column heading. A list arrow appears.
7. You can filter the values by any of the following actions:
   - Click the list arrow and type the first letter of the value If you type additional letters, your sort is more precise.

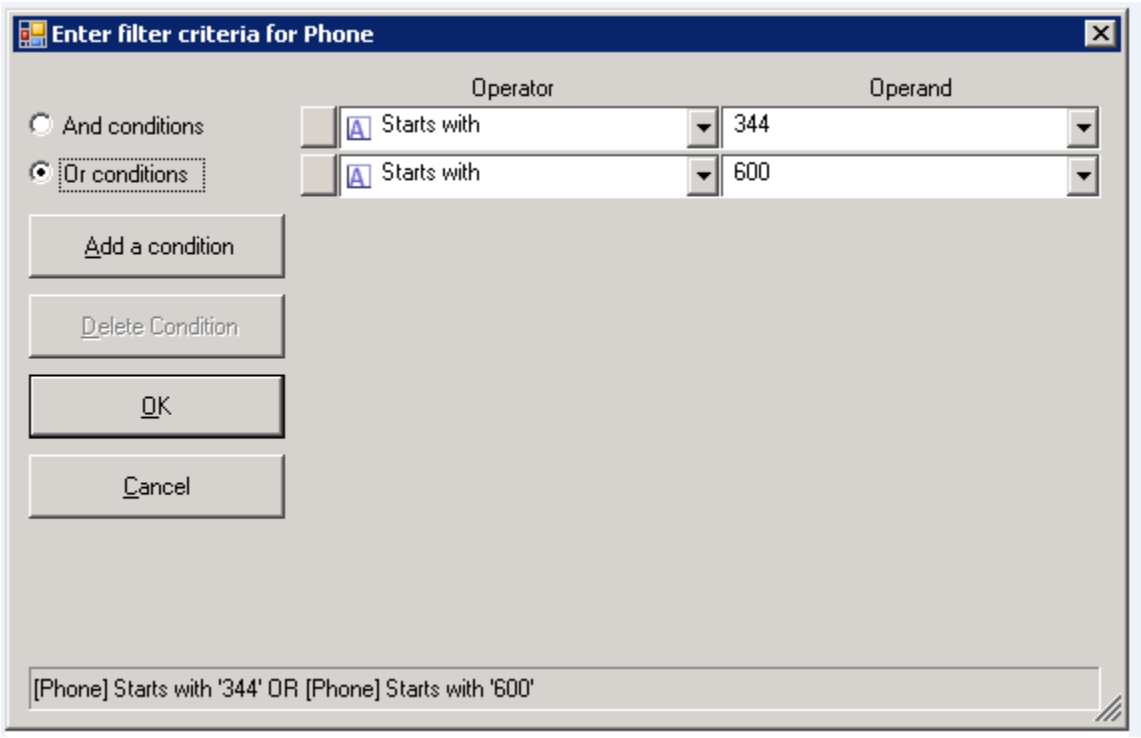


   - Click the list arrow and select an item from the list.

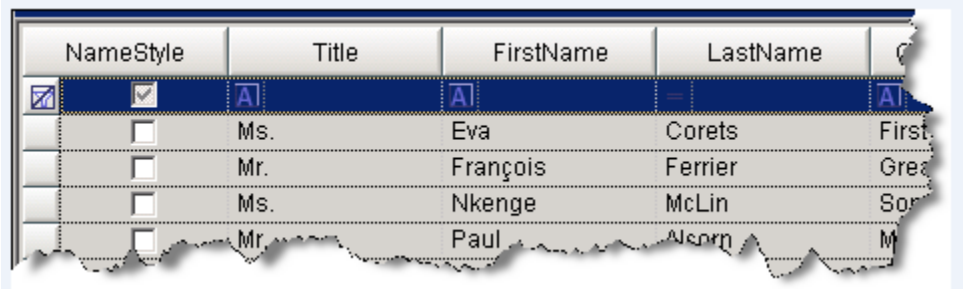
© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.



- If you select the **(Custom)** item from the list, you can further refine your search in the **Custom** dialog box. This example shows a search for two different area codes.



8. To perform a different search, clear your search criteria by clicking the cancel icon at the left of the filtering row.



© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Discard changes

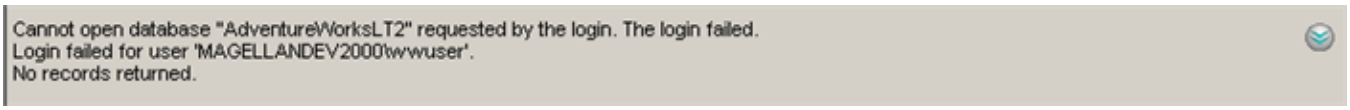
If you need to discard the changes made to grid values, click **Retrieve**. This action reloads the table values from the database and discards any changes. You can also initiate the action by using a script that sets the **CmdRetrieve** property to **True**.

View errors

You can manually display or close the **Status** panel in the lower part of the grid display area. The status panel can also automatically appear when an error occurs. However, after the panel appears, it remains on view.

If a command object is successful, the message "Success" appears in the Status panel. If the panel is closed, "Success" messages do not appear for successful command objects.

To close the panel, click the double-arrow icon at the right of the panel. To open the panel on demand, right-click and select **Show Status**.



Note: The error panel display does not change the overall area occupied by the grid graphic.

Write changes to the database

You can write changes to the database if the following conditions are met:

- The database query (or table) can be updated.
- The **ConfiguredReadOnly** custom property value is **False**.
- SQL Server security allows the security mode configured at the grid to perform an update.
- Changes made in the grid are consistent with the database schema.

Changes made in the grid are written to the database using the Write command. Changes are handled as a single database transaction. If any change fails, the entire set of changes is rolled back.

If you have sufficient privileges, you can change a value and click **Write**. You can also use a script to set the **CmdWrite** property. This action causes the grid contents to be evaluated for changes and for the changes to be written to the database.

Tip: After you modify cells in the grid and write them back to the database, you should also verify that the data is correctly written. In certain cases, such as a database table with an automatically incrementing primary key or interdependent cells, the grid does not show the true contents of the database table after you write back to it. Also, no error message is shown. It is recommended that you execute a retrieve or auto refresh command after writing to the database to make sure it is showing the current data.

Views and updates

Some views cannot be updated for the following reasons:

- The views contain table joins on keys other than primary keys.

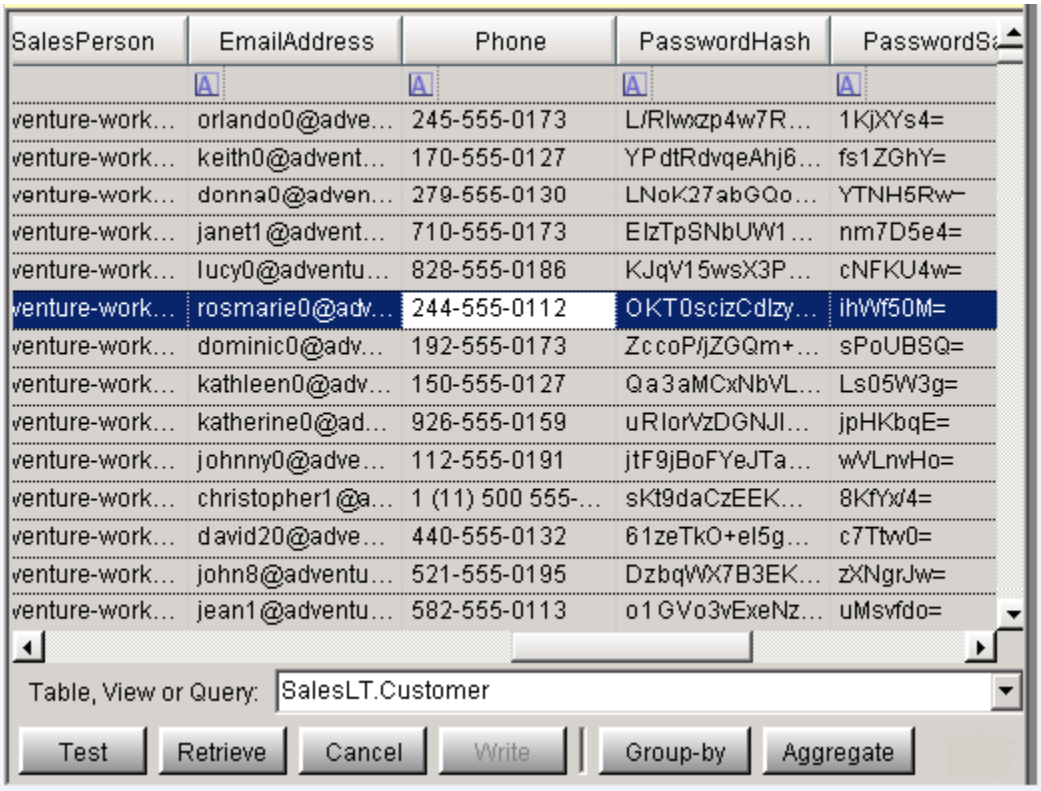
© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

- The views include security settings that prohibit updates.

If the write process fails, you can view the status in the **Error** panel. You can also monitor the **StatusDesc**, and **StatusError** properties by using text fields on the InTouch window for run-time display.

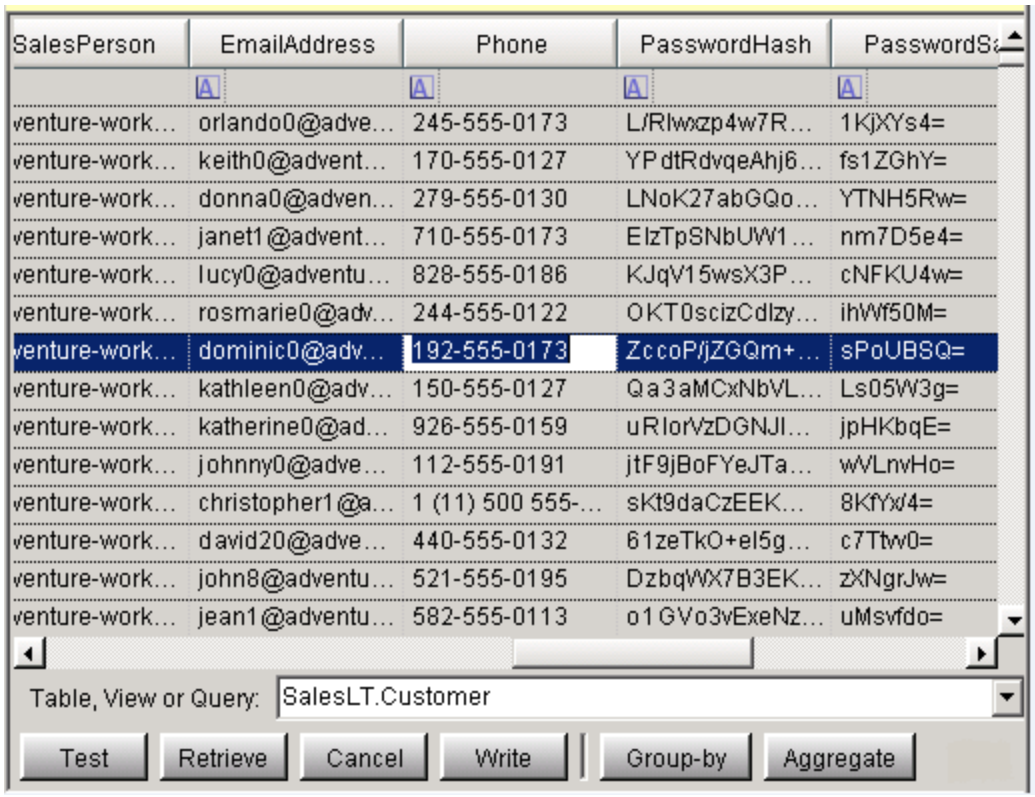
To make a change to data in the data grid

1. Ensure that the **ConfiguredReadOnly** custom property value is **False**.
2. Click **Runtime**.
3. Enter a SQL statement or the name of a database table or view.
4. Click **Retrieve**. This example shows data from the customer table in AdventureWorks. You will modify a phone number.
5. Select the phone number that you must modify. The area becomes editable.



6. Change the last four digits of the phone number to 0122.
7. Click an area outside the edit area. The **Write** button becomes enabled.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.



8. Click **Write**.
9. To verify the change, click **Retrieve**.

Database connection management

An important feature of the SQLDataGrid is the ability to optimize the management of multiple simultaneous connections to the database. This section describes how the infrastructure handles the database connection pool and sharing for the SQLDataGrid.

Pool and shares connections

Multiple instances of the SQLDataGrid configured with identical connection strings use a shared connection pool and SQL command queue when they are active in the same instances of the InTouch WindowViewer. To limit the size of the shared pool, you can use the connection pool property **MaxConnections**. For configuration options, see Custom Properties for the SQLDataGrid Graphic.

Physical connections to the SQL database are opened based on command requests and are closed when no requests are pending. All SQL commands are queued and processed based on availability of connections and processing time.

The SQLDataGrid asynchronously processes all SQL commands. Asynchronous command processing ensures availability of interaction with the InTouch WindowViewer during time-consuming command processing.

Any database connection that is established in a script within a graphic using the SQLData Script Library also takes advantage of connection pooling. Similar connection strings from either the SQLDataGrid or the script in the same InTouchView application share a common pool.

Note: Connection sharing and connection pooling require that the same security mode and user credentials be

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

used for all SQLDataGrid graphics within the InTouchView application that shares the connection or the pool.

Custom properties for the SQLDataGrid graphic

The following table lists all available Custom Properties. Data types are shown in the Custom Properties panel and are not included in this table.

Property Name	Description
Authentication	Designates whether database user authentication is SQL Server or Windows. **Writable at Run Time**: Yes **Default**: Windows
AutoRefresh	Automatically refreshes data appearing in the SQLDataGrid. When True, the query that repopulates the SQLDataGrid runs after the **AutoRefreshDuration** time elapses. **Writable at Run Time**: Yes **Default**: False
AutoRefreshDuration	Designates the requested time interval in seconds for AutoRefresh. The actual refresh rate can be slower than the specified rate. **Writable at Run Time**: Yes **Default**: 60
ButtonPanelHide	Hides grid buttons when the value is True. **Writable at Run Time**: Yes **Default**: False
CmdCancel	Cancels the last database query. **Writable at Run Time**: Yes **Default**: False
CmdClear	Clears the contents of the SQLDataGrid. **Writable at Run Time**: Yes **Default**: False
CmdCopy	When True, copies the current selection to the clipboard. **Writable at Run Time**: Yes **Default**: False

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Property Name	Description
CmdLogDiagnostics	Writes the current DCM diagnostic values to the logger. **Writable at Run Time**: Yes **Default**: False
CmdPageFirst	Moves the SQLDataGrid display to the first row. **Writable at Run Time**: Yes **Default**: False
CmdPageLast	Moves the SQLDataGrid display to the last row. **Writable at Run Time**: Yes **Default**: False
CmdPageNext	Moves the SQLDataGrid display down one page. **Writable at Run Time**: Yes **Default**: False
CmdPagePrev	Moves the SQLDataGrid display up one page. **Writable at Run Time**: Yes **Default**: False
CmdPaste	When True, pastes the current text contained on the clipboard to the grid. **Writable at Run Time**: Yes **Default**: False
CmdRefresh	Refreshes the current SQL query data. Any pending changes are discarded. **Writable at Run Time**: Yes **Default**: False
CmdResetDiagnostics	Resets the current DCM diagnostic values and clears the message and event counters. **Writable at Run Time**: Yes **Default**: False
CmdRetrieve	Loads the selected table, view, or query to the SQLDataGrid. Any changes to a previous dataset are discarded. **Writable at Run Time**: Yes **Default**: False

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Property Name	Description
CmdTestConnection	Tests the database connection for the specified server, database, security type; and if applicable, user name and password. The result is placed in the Error panel. **Writable at Run Time**: Yes **Default**: False
CmdWrite	Submits changes made in the SQLDataGrid to the database. **Writable at Run Time**: Yes **Default**: False
ColumnAggregateEnable	Specifies whether the column aggregate feature is exposed by default. **Writable at Run Time**: Yes **Default**: True.
ColumnFiltering	Enables column filtering and provides a row with the progressive filtering entry field. **Writable at Run Time**: Yes **Default**: False
ColumnGroupingEnable	Specifies whether the column grouping feature of the SQLDataGrid is exposed by default. **Writable at Run Time**: Yes **Default**: False
ComboBoxListReadOnly	If False, you can make changes to the edit box above the combo list box. **Writable at Run Time**: Yes **Default**: False
ComboBoxListString	Enables a list of table or view names, separated by semicolons to be viewable in the **Runtime** combo box. **Writable at Run Time**: Yes **Default**: Blank
ComboBoxPanelHide	Hides the combo box and panel. **Writable at Run Time**: Yes **Default**: False

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Property Name	Description
CommandTimeout	The number of seconds that the SQLDataGrid waits for a pending command to run before timing out. **Writable at Run Time**: Yes **Default**: 30
ConfiguredDatabase	The name of the database that connects to the SQLDataGrid. **Writable at Run Time**: Yes **Default**: Blank
ConfiguredDomain	The domain to use with impersonation if the authentication mode is Windows. **Writable at Run Time**: Yes **Default**: Blank
ConfiguredPassword	SQL Server domain if the security type is Windows. **Writable at Run Time**: Yes **Default**: ********
ConfiguredReadOnly	Prevents the SQLDataGrid from making changes to the database. **Writable at Run Time**: Yes **Default**: True
ConfiguredServer	The name of the computer hosting the database. **Writable at Run Time**: Yes **Default**: Blank
ConfiguredUser	The SQL Server user name if security type is SQL. This property is optional when security type is Windows. **Writable at Run Time**: Yes **Default**: Blank
ConnectionTimeout	The number of seconds that the SQLDataGrid tries to connect to a database before timing out. **Writable at Run Time**: Yes **Default**: 15

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Property Name	Description
ContextMenuEnable	When True and you right-click the grid, the context menu appears. When False and you right-click the grid, the context menu does not appear. **Writable at Run Time**: Yes **Default**: True
ErrorPanelDisable	Disables the error message display. **Writable at Run Time**: Yes **Default**: False
ErrorPanelShow	When True, the error panel is visible. When False, the error panel is hidden. **Writable at Run Time**: Yes **Default**: False
FeatureButtonsHide	Hides the **Aggregate** and **Group-by** buttons. **Writable at Run Time**: Yes **Default**: False
FontBold	Specifies whether to display the SQLDataGrid font in bold text. **Writable at Run Time**: Yes **Default**: False
FontName	Specifies the name of the font. **Writable at Run Time**: Yes **Default**: Arial
FontSize	Specifies the point size of the font. **Writable at Run Time**: Yes **Default**: 9.0
InstanceName	A name that you choose to identify a specific SQLDataGrid instance. The name appears in the **ConnectionName** diagnostic. **Writable at Run Time**: Yes **Default**: Blank

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Property Name	Description
MaxConnections	The number of active connections allowed in the connection pool for a given connection string. **Writable at Run Time**: Yes **Default**: 100
RowCount	The row count returned in the last successful data request. **Writable at Run Time**: No **Default**: 0
RowLimit	Modifies the query to return a specified number of records. To prevent the database query from being modified, set the **SQLQueryPassThrough** custom property to True. **Writable at Run Time**: Yes **Default**: 20,000
RowsChanged	If True, one or more rows have been changed in the grid (added, odified, or deleted). If False, no rows were changed in the grid. **Writable at Run Time**: No **Default**: False
SelectedRowNumber	The number of the selected grid row. **Writable at Run Time**: Yes **Default**: 0
SQLQuery	Specifies the SQL query, table, or view data to be retrieved to the SQLDataGrid. If **SQLQueryPassThrough** is True, you must add the SELECT statement along with the table or view name. For example, SELEC*T * FROM person.contact*, rather than *person.contact*. **Writable at Run Time**: Yes **Default**: Blank
SQLQueryCurrent	Specifies the SQL command that was run to show the current data in the SQLDataGrid. **Writable at Run Time**: No **Default**: Blank

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Property Name	Description
SQLQueryPassThrough	If True, enables the SQL statement specified in the **SQLQuery** custom property to be passed directly to the database without modification. When this property is True and the **SQLQuery** custom property is set only to a table name or view, the grid does not insert "SELECT * FROM" in front of the table or view name. Additionally, if this property is True, the top "n" is not inserted into a SELECT statement (where "n" is the value of the RowLimit custom property). **Writable at Run Time**: Yes **Default**: False
SQLQueryPending	Indicates a pending query. **Writable at Run Time**: No **Default**: False
StatusDesc	Specifies the error description from the last query. If no error exists, the string is empty. **Writable at Run Time**: No **Default**: Blank
StatusError	Specifies the error number from the last request. **Writable at Run Time**: No **Default**: 0
StatusIsError	Indicates whether an error occurred in the last query. True = Error False = no error **Writable at Run Time**: No **Default**: False
SupportThemes	Specifies the SQLDataGrid window is formatted with Windows XP or Vista visual themes. You should always set the value of this custom property to True. This custom property is not yet supported. **Writable at Run Time**: No **Default**: True

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Property Name	Description
SuppressAuthenticationWarning	If user specifies any values in the ConfiguredUser and ConfiguredPassword properties, then a warning textbox is displayed in runtime. The warning informs the user of the possible security issue of storing the database user credentials locally. If True, hides the textbox regardless of configuration. **Writable at Run Time**: No **Default**: True
WriteButtonHide	Hides the **Write** button used to send changes to the database. **Writable at Run Time**: Yes **Default**: False

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Deploy

You can deploy and test your objects at any time during development. When you are ready to test or run the application in production, you deploy the Galaxy.

You can see what your application looks like in the Deployment view or the Model view. Both views show you the structure of your application. For more information, see IDE application views.

A simulation server, derived from the $OPCClient Device Integration system object, is included with Galaxies created from the Default.cab file. The simulator can be imported to or created for other Galaxies. Use the simulator to test a Galaxy prior to deployment.

When you create a Galaxy from the Default.cab file, the Galaxy's assets are automatically assigned to the simulator's Fast scan group. When you have completed testing and created new Device Integration objects, move the assets from the simulator to scan groups within the new DI objects.

Deployment overview

The deployment process copies a Project's assets from the development environment to the run-time environment. Once they are deployed, your assets are "live" and functional.

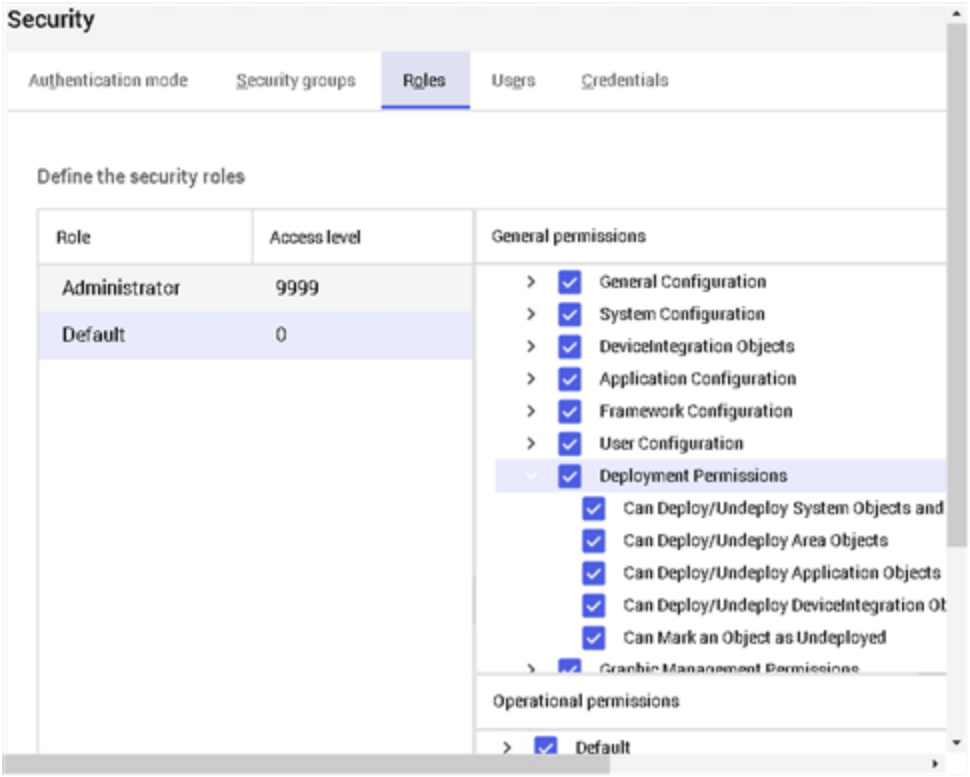
Configuration changes do not appear in the run-time environment until the changes are deployed.

You must have the appropriate level of permissions to deploy assets.

To view your permissions

1. From the ribbon, select **Galaxy,** then select **Configure** and **Security**.
2. Select Roles. To make sure you have the permission to deploy an application, in the **General Permissions** section, expand the **IDE Permissions** item, then expand **Deployment Permissions**. Make sure all necessary deployment-related items are checked.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.



To deploy an asset from the System Platform IDE

1. In the Deployment or Model view, select the asset or assets you want to deploy.
2. On the **Home** ribbon, in the **Deploy** area, select **Deploy**.

For detailed instructions, see Deploy objects.

Deploy objects from the IDE to run time

Objects deploy from the configuration environment to the run-time environment as follows:

IDE **config environment**		**Object Viewer** **run-time environment**
Galaxy database	®	[Does not exist in run-time environment]
Templates	®	[Does not exist in run-time environment]
Instance objects	®	Instance objects [Run-time configuration and behavior]
Security: General permissions	®	[Does not exist in run-time environment]

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

IDE config environment		Object Viewer run-time environment
Security: Operational permissions	®	Run-time permissions to acknowledge alarms and modify attributes
Scripts configuration	®	Scripts execution
Alarms configuration	®	Alarms generate and acknowledge
History configuration	®	History Logs [Historian]

Allocate run-time resources

Distribute components across platforms and engines as you develop your IDE configuration environment. This will help to balance the I/O, memory, and CPU burden across computing resources at run time.

The operating system determines engine assignment to individual CPU cores, and attempts to load balance across CPU and other computing resources. If you need to change how resources are allocated to improve load balancing, undeploy the objects and reassign them in the IDE configuration environment.

- If you are using redundant AppEngines, you can control how objects are loaded at runtime. For more information, see CPU load balancing. For details about implementing redundancy, see Redundancy.
- If you are using Flex licensing, see Allocate AppEngines to platforms for more information.

Allocate galaxy repository node memory

The Galaxy database resides on SQL Server, which runs on the Galaxy Repository (GR) node. In some cases, particularly when a GR node contains less than the recommended RAM amount, system performance may be impacted as SQL Server will use more CPU to compensate for the lower amount of available memory. In some cases, high CPU usage can even occur on systems configured with the recommended RAM amount. A galaxy with many objects, for example, can use more memory and cause high CPU usage.

If you notice CPU usage increasing over time to the point that is affecting system performance, we recommend that you set the SQL Server minimum memory (min server memory) to 1/3 of total physical memory. Application Server automatically configures the max server memory setting, and there is no need to manually configure it.

Example minimum server memory settings:

Application	Level	Installed RAM (GB)	min server mem (MB)
Small (1 - 25,000 I/O per Node)	Minimum	2 GB	682 MB
	Recommended	4 GB	1365 MB
Medium (25,000 - 50,000 I/O per Node)	Minimum	8 GB	2730 MB
	Recommended	12 GB	4096 MB

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

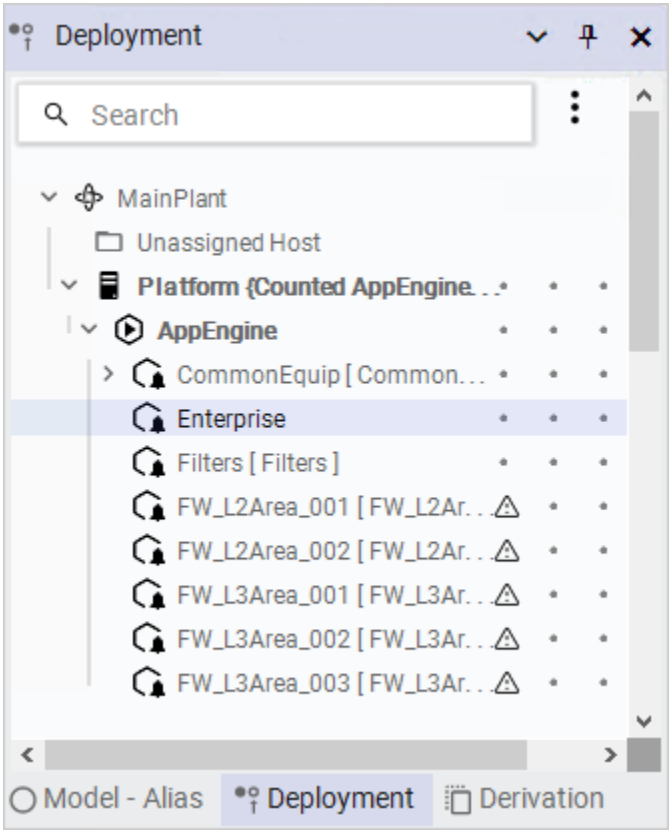
Large (> 50,000 I/O per Node)	Minimum	16 GB	5461 MB
	Recommended	24 GB	8192 MB

Object, area, engine, and platform assignments

Getting Started with the IDE shows the basic Galaxy structure. When you create an application object, such as a User Defined Object, it must be assigned to an area. When you look at your objects in the Deployment view, an area is assigned to an AppEngine, and an AppEngine is assigned to a WinPlatform.

The **Deployment** view uses the WinPlatform as its foundation, and all objects cascade from the WinPlatform object. The WinPlatform object can be thought of as the node (computer) that hosts the rest of the objects. The basic hierarchy of objects in the **Deployment** view is:

	Galaxy >	WinPlatform >	AppEngine >	Area >	Application Object DI Object



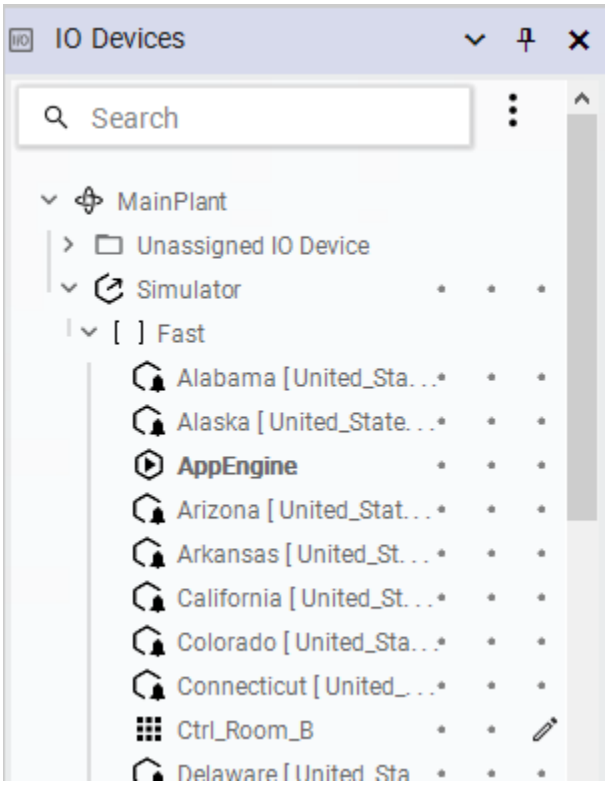
Data Integration (DI) Objects provide the interface between application objects and PLCs in the physical environment. DI Objects contain scan groups, and application objects are assigned to scan groups; this assignment can be done in the **IO Devices** view.

The relationship of other objects in the Galaxy to DI Objects is most clearly viewed in the **IO Devices** view. All IO between objects in the Galaxy and the PLC is through a DI Object. Scan groups are configured in the DI Object, and objects are assigned to a scan group.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

	Galaxy >	DI Object >	Scan Group >	AppEngines Areas Application Objects

While objects can be deployed from any IDE view, the **Deployment** view provides the simplest way to understand the object hierarchy during deployment. The **IO Devices** view provides the simplest view of the relationship of each scan group and its assigned objects.



Allocate AppEngines to platforms

If you are using Flex licensing, you must configure the number of AppEngines allowed on each platform object in the Galaxy. You can set each platform to **No AppEngine**, **Counted AppEngines**, or **Unlimited AppEngines**. For any platform object that you set to **Counted AppEngines**, you then set the number of AppEngine licenses for that platform.

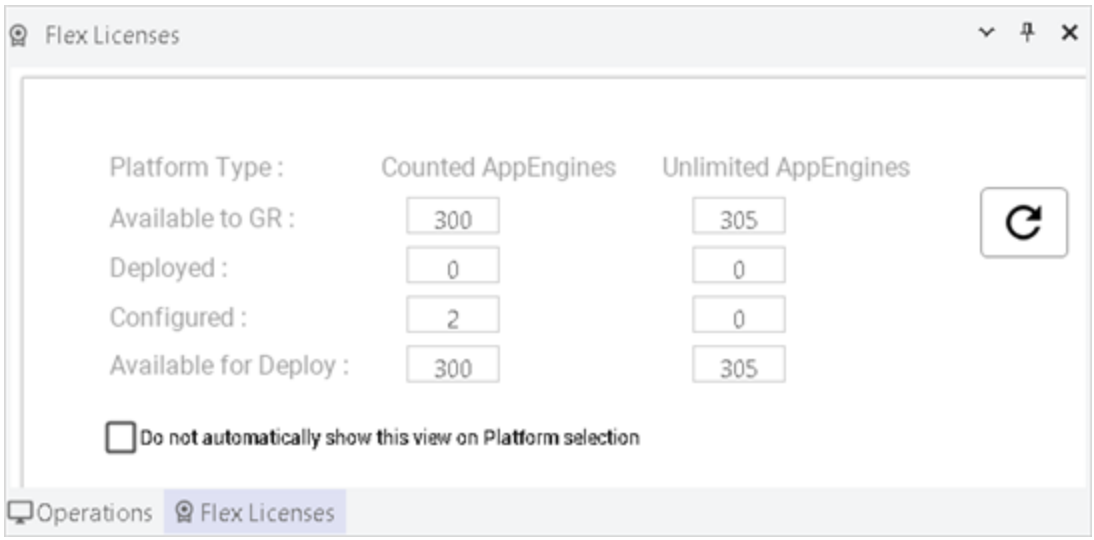
You can configure Flex licensing for a platform only from **Deployment** view.

Note: If you use the **Counted AppEngines** or **Unlimited AppEngines** setting, the actual number of engines that will be usable on a platform may be limited by system performance, which will vary due to hardware, number of IO operations, historization, and other factors such as galaxy complexity and other running software. This limitation also applies when using perpetual licenses.

To configure Flex licensing

1. If the **Flex Licenses** view is not shown, select the **View** ribbon, then, in the **Windows** area, select **Flex licenses** to display the view. Check how many licenses are available.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.



2. Select **Deployment** view.
3. Select the WinPlatform object to configure for licensing.
4. If the platform object is deployed, undeploy it.

   Note: If you are increasing the number of licenses for a platform that is set to **Counted AppEngines**, you do not need to undeploy the platform. You do need to undeploy the platform to decrease the number of licenses or change the type of licensing.
5. Right-click the object and select :**Flex Licensing**.
6. Select the desired number of AppEngine licenses for this platform:
   - **No AppEngine** means this platform is not licensed for any AppEngines.If you assign an AppEngine to the platform, the platform will show a configuration error and you will not be able to deploy it or any objects beneath it in the deployment hierarchy.
   - **Counted AppEngines** lets you assign one or more AppEngines to the platform. The **Configure hosted engine count** dialog box opens. Type the number of licenses to assign to this platform or use the up and down arrow keys to select the number. You can assign AppEngines up to the number of assigned licenses. If you assign too many AppEngines, you will receive a configuration error and the extra AppEngines and objects assigned to them will not be deployable. When too many engines are assigned, licenses are assigned randomly; you will not know until deployment which engines have been licensed and deployed and which have not.

     Note: You can increase or decrease the number of licenses assigned to a platform by selecting **Counted AppEngines** again to return to the **Configure hosted engine count** dialog box.
   - **Unlimited AppEngines** lets you assign any number of AppEngine objects to the platform. As noted above, there is a practical limit to the number of AppEngines based on various software and hardware factors.
7. Deploy the platform.
8. Look at the **Flex Licenses** view to see the updated license availability counts. The count is automatically updated when a platform is deployed or undeployed,after a connection to the galaxy, or after a change to the platform license assignment. You can manually update the license counts shown by clicking the **Refresh** button.



© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Data Integration Objects

Data Integration Objects (DI Objects) provide the interface between PLCs and Galaxy objects, including area objects and application objects. DI Objects contain scan groups (topics); application objects are assigned to individual scan groups to handle IO between the objects and the PLC.

Auto-build

Refer to the *OI Core Communications Driver Help* for information about the Auto-Build process. Key points to remember include:

- Specify a new Galaxy as the target Galaxy until you are familiar with how Auto-Build works. This will help to avoid accidentally overwriting objects.
- Always specify a prefix. This will also help you avoid overwriting objects and lets you differentiate between similarly-named controllers.
- Use pre-check validation to catch illegal object names, such as names that exceed the maximum number of characters and that contain illegal characters. Note that Auto-Build skips names with errors.
- You can only run one Auto-Build process at a time.
- The user who launches the OCMC must be part of the oiAdministrators Windows Local Users and Groups to access the Auto-Build functionality
- Auto-Build add two levels of derivation to accommodate different layers of standardization, for example, corporate standards (global) and local standards.

Auto-Build creates a DI Object for you. The DI Object is derived from the $DDESuiteLinkClient template, and will include the two character prefix that you specify (first character must be alphabetic, the second character can be a letter or number (alphanumeric). The DI Object must include at least one scan group (topic).

The Auto-Build process adds the prefix that you specified to all objects it captures. After Auto-Build completes, you can view, test, and adjust object assignments as needed, using the different IDE views.

- Use the **Deployment** view to modify object assignments.
- Use the **IO Devices** view to modify scan group assignments.
- Use the **IO Device Mappings** view to verify and test IO references.

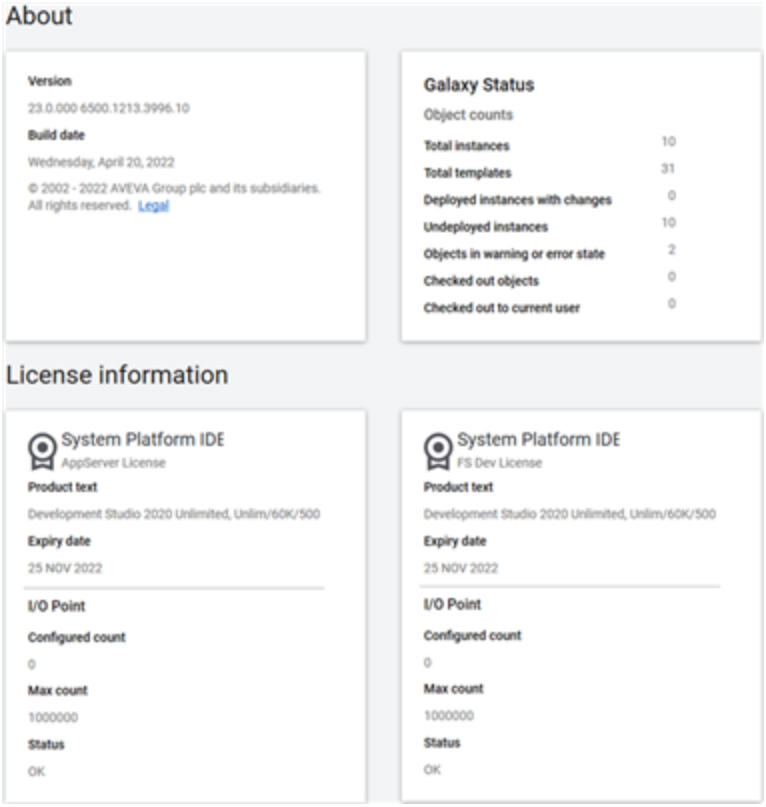
Determine galaxy status

You can see an overview of the condition of your Galaxy before you deploy. This lets you know if you have objects that are in warning or error status.

To determine the status of a Galaxy

1. Connect to the Galaxy.
2. On the ribbon, select **Galaxy**, then select **License** at the lower-left corner of the window. This displays information about your software version, system licensing and your galaxy status information..

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.



The **Galaxy Status** area shows the counts of total instances, total templates, deployed instances with changes, undeployed instances with changes, objects that have an error or warning state, objects that are checked out, and object you have checked out.

3. Click **OK**.

Configure advanced communication management

Advanced Communications Management minimizes network traffic and CPU usage of a DI Object and DAServer when a particular DI attribute is currently subscribed to, but its value is not currently shown. For example, scanning for updated values from a DI attribute representing pump RPM is suspended when an operator minimizes the application window containing a pump graphic containing attribute references that subscribe to the DI object's pump RPM attributes.

Note: This dependence on the attribute being referenced by a running application is only the case when the attribute subscription is not already active due to configuration for history, alarming, or scripting. For example, if an attribute is configured for history, the subscription to the field device will always be active regardless of what window is active in InTouch, and regardless of whether InTouch is running at all.

When the items on the current window are no longer active because the window is minimized, a Suspend operation is performed for each item. Suspending scanning causes data updates to stop flowing to the application without deleting the subscriptions for each item. This reduces the overhead of unsubscribing and subscribing again to items when windows are minimized and restored.

In Advanced Communication Management, applications monitor the number of outstanding references to attributes that are updated with values from an external source object. When an attribute's external reference

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

count reaches zero, the application sends a Suspend message to the source object requesting it to suspend updates. For example, a UserDefined object named PumpRPM has an attribute named RPM, which has an integer data type. The input source of RPM is plc1.n7:11, where plc1 is an ABPLC DI object.

When there are no active subscriptions to ud1.RPM, the subscription between input source and PumpRPM is suspended. When a client like ObjectViewer or a managed application running in WindowViewer subscribes to ud1.RPM, an Activate message is sent to the DI object to start sending data between the plc1 object and the PumpRPM object. This enables ObjectViewer or a managed client application running in WindowViewer to receive current plc1 data.

Unlike defined static attributes, some scan groups contain *dynamic* attributes. Dynamic attributes within a scan group provide the means for a running application's objects to dynamically subscribe to data from an external device. Dynamic attributes are created, activated and added to the scan group using ApplicationObject attributes or InTouch indirect tags and scripts that include the IOSetRemoteReferences function.

When a dynamic attribute is poked, the time stamp is updated to the timestamp passed in with the value if available, or the current time provided by the hosting AppEngine is used. If the data provider passes in a Value, Time, Quality (VTQ) triplet of data in the callback, the time stamp is associated with the value and quality used to update the attribute.

To enable Advanced Communication Management, you must:

- Select Advanced Communication Management for your Galaxy.
- Set the scan mode for each scan group that belongs to your device integration objects within the Galaxy.

Select advanced communication management

Advanced Communication Management is a Galaxy-wide configuration setting that you set from the System Platform IDE. Advanced Communication Management is disabled by default.

When upgrading a Galaxy or restoring a Galaxy from a .cab file, the configuration setting that was previously selected is retained.

For a description of Advanced Communication Management, see Advanced Communication Management.

To configure advanced communication management

1. Connect to the Galaxy.
2. From the ribbon, select **Galaxy**, then select **Configure.**
3. Select **Communication**, then **Advanced Communications Management**. The **Advanced Communications Management** information appears.
4. If necessary, select **Enable advanced communication management** and select **Save**.

   Note: Any deployed platform and application engines must be restarted when the Advanced Communication Management state changes. Restarting should be done by means of the OCMC.

Configure scan modes

A scan group is a collection of object attributes with associated data items. Scan group attributes reflect I/O points in the address space of Data Access Server whose values are polled from devices at a common update interval. The update interval for each scan group is inherited from the scan groups configured in the source DIObjects.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

You can configure scan groups for OPCClient, InTouchProxy, DDESuiteLinkClient, and RedundantDIObject device integration objects. Within each scan group, you can specify data items to use as the object attributes. At run time, scan group items are updated with the latest values from the OPC DAServer according to the update interval.

The OPCClient, RedundantDI, DDESuiteLinkClient, and InTouchProxy objects contain a Scan Mode attribute that can be set for each scan group. The assigned scan mode value determines if objects are continuously updated with current data when an operator minimizes or closes the application window.

You can assign three different scan modes to a scan group:

- **Active**
  - Items are deleted and added to the scan group as requested (referenced) by the clients.
  - Items that exist when a scan mode change occurs are not deleted unless the previous mode was ActiveAll and the items are no longer referenced.
  - The Active scan mode polls all points that are referenced, whether active or inactive.
  - In Active scan mode an attribute is always in the active state. When the last reference to the attribute is unregistered/unadvised, the attribute is deleted.
- **ActiveAll**
  - Scanning is continuous and dynamic attributes are never removed when unsubscribed.
  - Items are activated by client requests, but continue to exist even after the client are not interested in them anymore.
  - Items are not removed when the scan mode changes.
  - The scan group polls the field device for all points irrespective of whether they are currently active, inactive, or even subscribed to by items.
- **ActiveOnDemand**
  - When the scan mode is configured as ActiveOnDemand, attributes that are not actively being referenced by any client or object are made Inactive and are not scanned.
  - New Items are deleted and added to the scan group as requested (referenced) by clients.
  - Items that exist when a scan mode change occurs are not deleted unless the previous mode was ActiveAll and the items are no longer referenced.
  - ActiveOnDemand scan mode polls the field device for currently active items only. Inactive items are not scanned.

The following table shows scan mode scan states for dynamic attributes based on whether items are referenced or not.

Dynamic Attribute State	Scan Mode Active	Scan Mode ActiveAll	Scan Mode ActiveOnDemand
Reference/ Some active	Scan	Scan	Scan
Reference/All Inactive	Scan	Scan	No Scan
Not Referenced	Delete	Scan	Delete

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

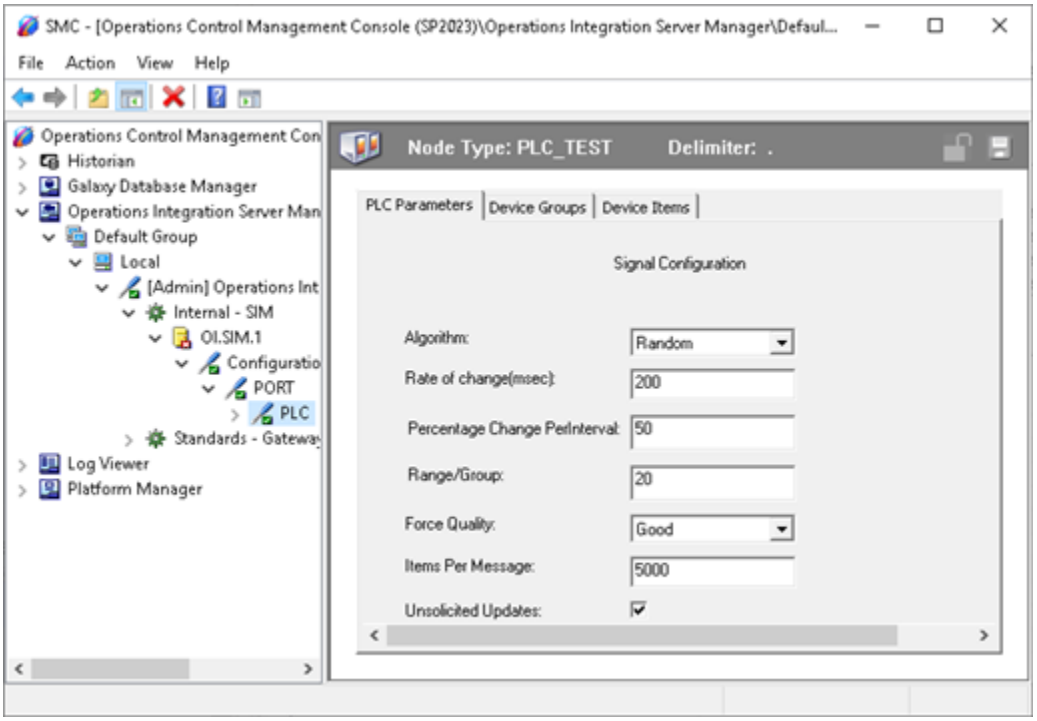
To set a scan mode for a scan group

1. Edit the device integration object.
2. Show the editor page containing scan groups by completing one of the following:
   - Click the **Scan Group** tab if you are editing a RedundantDIObject or OPCClient object.
   - Click the **Topic** tab if your are editing a DDESuiteLinkClient object.
   - Click the **Items Configuration** tab if your are editing an InTouchProxy object.
3. If necessary, add a scan group by clicking the plus sign box above the list of available scan groups.
4. If necessary, set the length of the scan group update interval in milliseconds. The default update interval is 500.
5. Double-click in the **Scan Mode** box of the scan group to show a drop-down list of available scan modes. ActiveOnDemand is the default scan mode.
6. Select a scan mode from the list.
7. Save your changes to the device integration object and exit from the editor.

Simulation server

System Platform includes a simulated data access server that lets you develop and test a Galaxy before deploying it to your production environment.

The simulation server's hierarchy can be viewed in the **Operations Integration Server Manager**, within the **Operations Control Management Console (OCMC)**. To run the OCMC, from the **Start** menu, point to **AVEVA** and select **Operations Control Management Console**.



The simulation server is automatically installed on the Galaxy Repository node for Galaxies derived from the Default.cab file, and is always available. It mimics the data output from a PLC. The simulation server sends data

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

to attributes of instances configured for I/O Automatic Assignment in your Galaxy, the same way that a PLC would.

The simulation server contains all the system objects needed to create a functional test environment for testing I/O references that use I/O Automatic Assignment. However, some manual configuration is required before the simulation server can be used. The simulation server includes an Operations Integration server and a PLC simulator.

The simulation server is based on an OPC client device integration (DI) object. You can add it to Galaxies that do not contain it by default. If you need to add the simulation server, you will configure it the same way as you would an OPC client in a normal production environment. Required configuration steps include choosing the server node and server name, and adding scan groups. This process is described in more detail below.

In its default configuration, the simulation server consists of the following elements:

- A channel for OPC communications between your Galaxy and the simulation server. The communications channel has the default name "Port."
- A PLC simulator to generate simulated data for your Galaxy. The PLC simulator has the default name "PLC."
- A device integration (DI) object that is used to link the PLC simulator to the instances in your Galaxy (OI.Sim.1).

Simulation server and the OPCClient object

The simulation server uses the OPCClient object as its basis. If your Galaxy does not include the simulation server object, you can configure one from the $OPCClient template. The OPCClient object can be used with the I/O Auto Assignment feature. In simulated environments, application objects in your Galaxy that contain I/O references are linked with the simulation server. In a production environment, these references would be linked to actual PLCs.

To use the simulation server, you must link instances to one of the simulation server's scan groups in the **IO Devices view** (the default scan group is **Fast**). The simulation server will return integer values for all supported data types. See View simulated data for the list of supported data types.

When you view simulated run-time data, a snap shot value is provided for each attribute that has I/O Auto Assignment. Since the simulation server tests I/O Auto Assignment references, references verified as good with the simulation server will automatically reconfigure when you associate the instances with the appropriate DI object and deploy the Galaxy in your actual environment.

I/O auto assignment in the simulation server

When using the I/O Auto Assignment feature, the simulation server uses the normal namespace used by I/O Auto Assignment, that is:

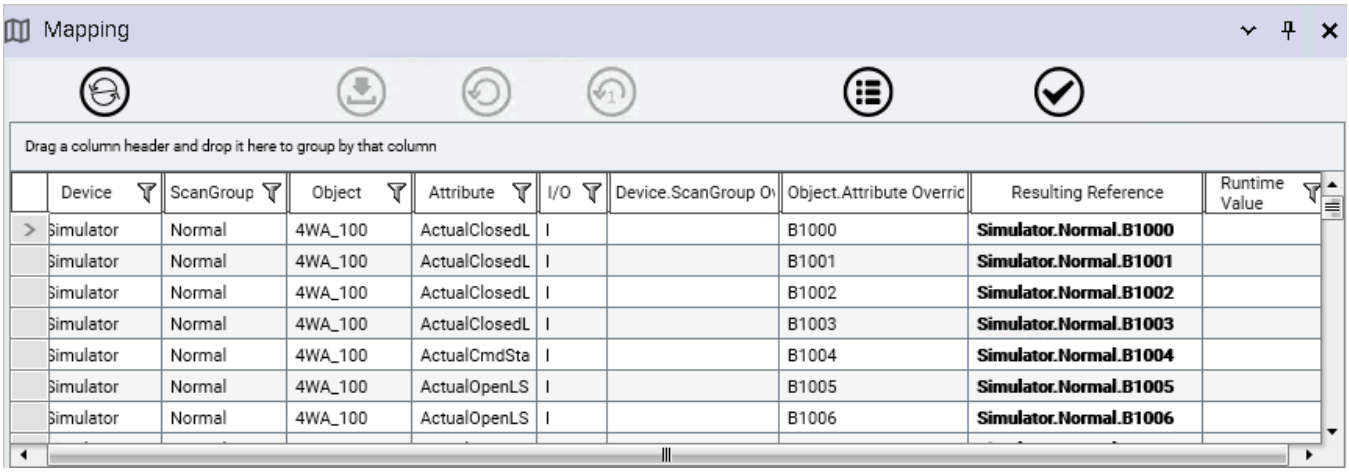
<IODeviceName>.<ScanGroupName>.<ObjectName>.<AttributeName>

You can view I/O assignments for objects in the I/O Device Mapping View. For objects configured with the simulation server and assigned to a Scan Group, the device and scan group for the I/O resulting reference will map as follows:

<IODeviceName> = **Simulator**

<ScanGroupName> = **Fast**, **Medium**, or **Slow**

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.



The IO Device name for the simulation server must not be changed from its default name, **Simulator**.

You assign application objects (instances) to the simulator server's scan groups the same way as you would assign instances to an I/O device's scan groups. The default scan group is **Fast**.

Configure the simulation server

The Simulation Server is automatically installed and configured for Galaxies derived from the Default.cab file. You can create a simulator server for Galaxies derived from other cab files that contain the $OPCClient base template, such as the Default_EMPTY.cab, or you can import the $OPCClient object after exporting it from another Galaxy.

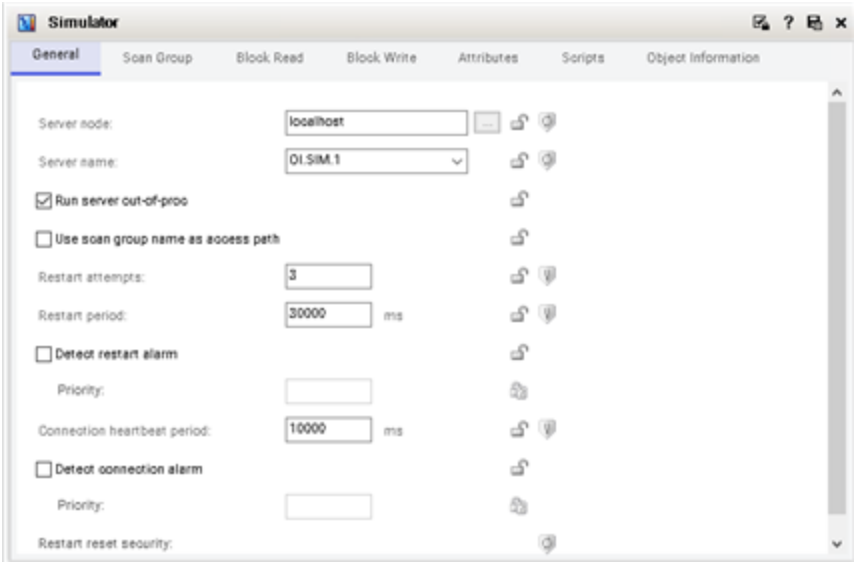
To configure a new Simulation Server

1. Derive an instance from the **$OPCClient** object in the Template folder.
2. Rename the new OPCClient instance **Simulator**.
3. Open **Simulator** in Object Editor and configure it as follows:
   - General page:

   a. Enter the **Server node** name.

   b. Select *OI.Sim.1* for **Server name**.

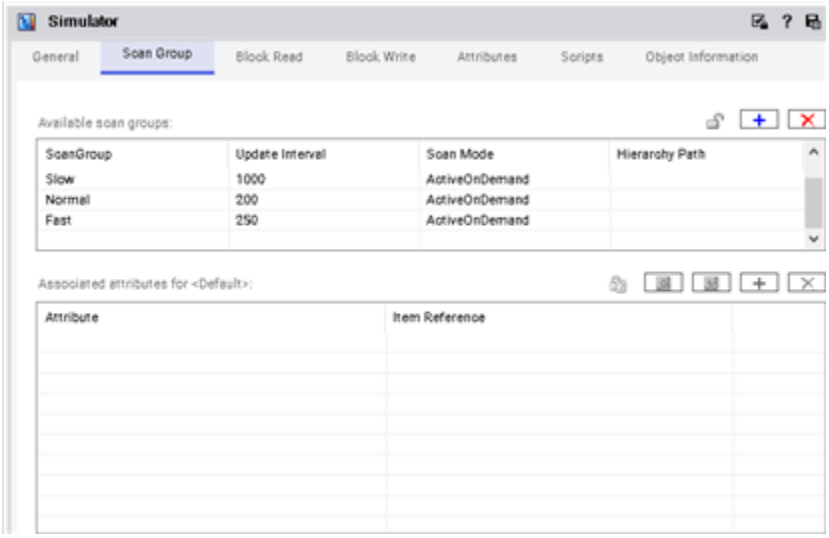
© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.



- Scan Group page:

a. Add a scan group called Fast.

b. If desired, you can also add Medium and Slow scan groups. You can leave *Update Interval* and *Scan Mode* at their default settings.

4. 

5. Save and close **Simulator**.

View simulated data

Before you can view simulated run-time data, your Galaxy:

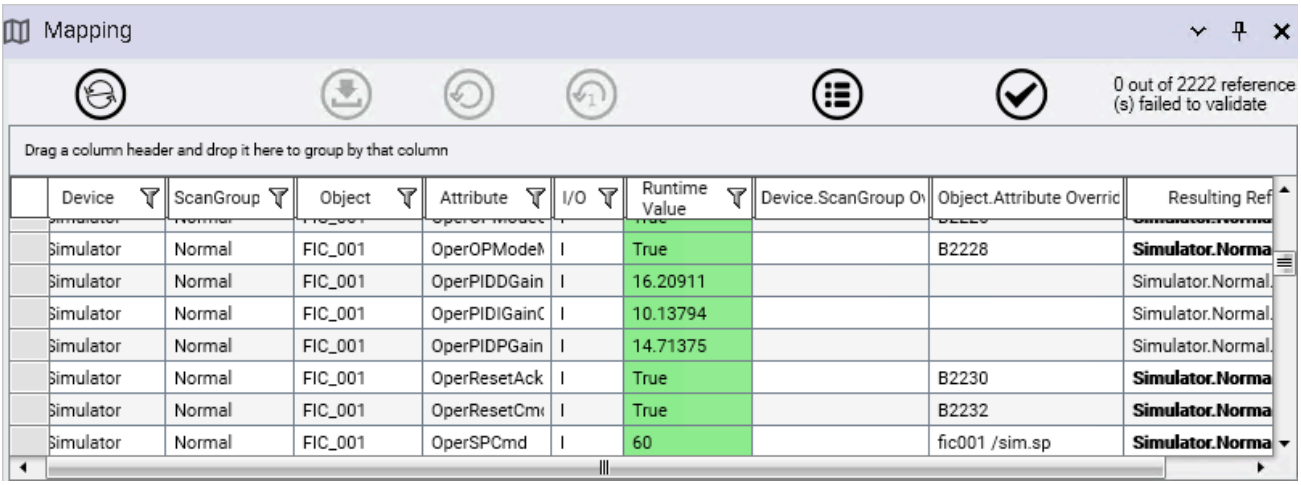
- Must contain attributes that are configured to use I/O Auto Assignment.
- Must be deployed along with the Simulator and any instances with I/O Auto Assignment. If necessary, you can link instances to the simulator server in the IO Devices view.

  Note: When instances are created, they are assigned to the Simulator Fast scan group by default. To change the scan group assignment of an asset, you must undeploy the instance first.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

To connect to the simulation server

1. In the Mapping view, select the **Validate Reference(s)** icon. The simulation server generates simulated numeric data and displays a randomly generated number for attributes that contain valid input references. Validated references are shown with a green background. References that did not validate are shown with a red background.
2. To refresh the simulated run-time value, select the **Validate Reference(s)** button again.



Supported data types

The simulation server supports the following data types:

- Boolean
- Integer
- Float
- Double
- String
- Time
- ElapsedTime
- InternationalizedString

All supported data types are treated as integer and return integer data for validated references.

Unknown data types

The simulation server treats attributes with unknown types as integer.

Deploy objects

You deploy object instances for two reasons:

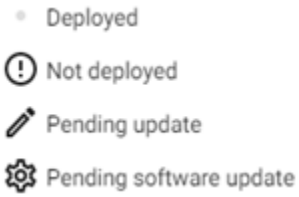
- Testing.
- Placing the application into production to process field data.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

When you are ready to deploy, make sure the following conditions are met:

- Bootstrap software is installed on the target computers.
- The objects being deployed are not in an error state in the Galaxy database.
- You created, configured, and checked in objects to the Galaxy.
- Objects are assigned to a host.
- The object's host is already deployed. A cascade deploy operation, which deploys a hierarchy of objects, deploys all objects in the correct order. This deploys an object's host before the object is deployed.
- Any associated script libraries are ready for use on the target computer. For more information, see Import script function libraries.

You can tell if you have objects that need to be deployed by looking at the status icons to the right of the object name in Model, Deployment, Derivation or IO Devices view. Deployment status icons include:

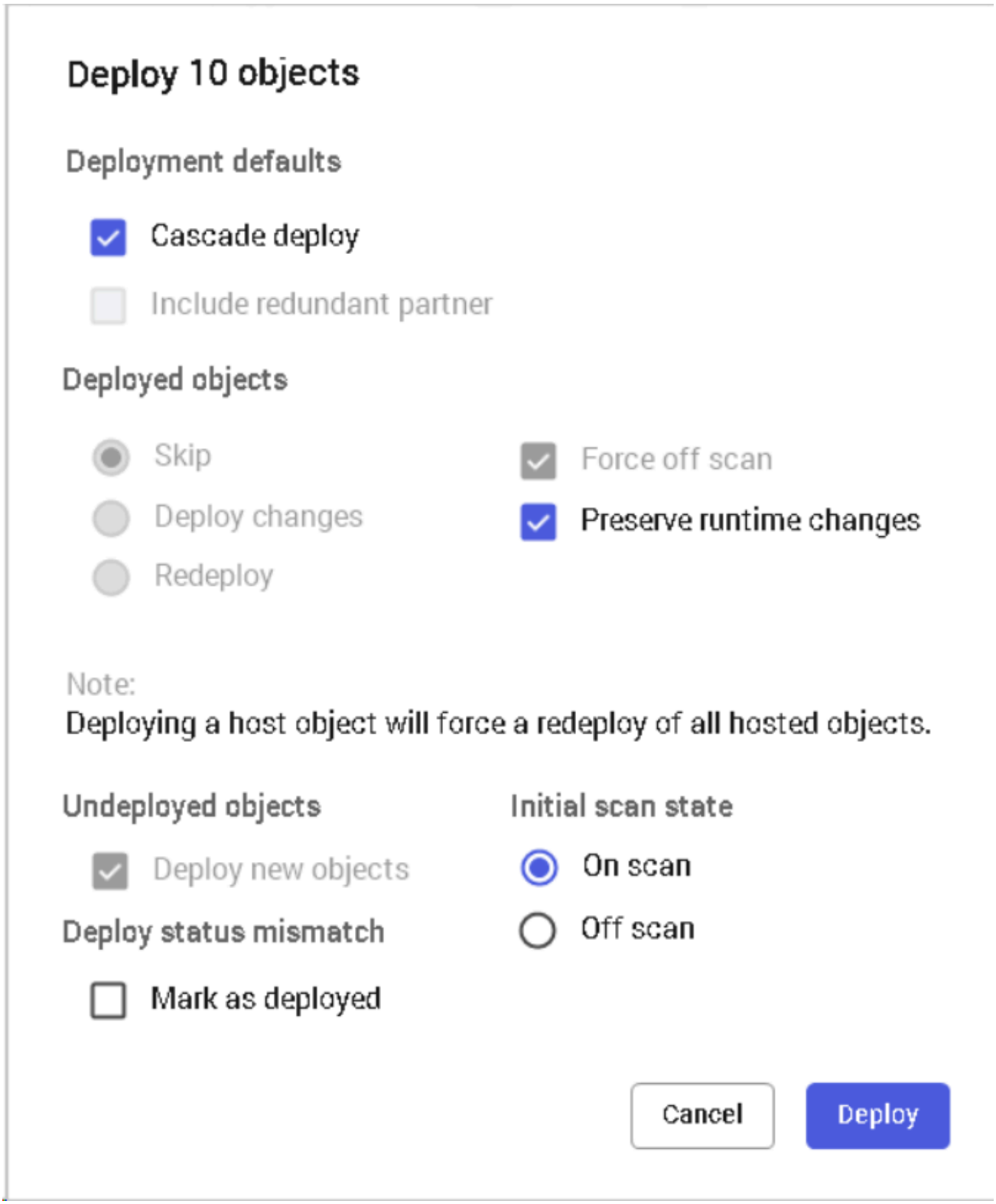


For more information about object and status icons, see Application Object Icons.

To deploy objects

1. Select the objects you want to deploy in an **Application view**.
2. Select **Deploy** from the ribbon. You can also right-click on the selected object(s) and then click **Deploy** in the context menu. The **Deploy** dialog box appears.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.



3. Select one or more of the following:
   - **Cascade Deploy**: Select this check box to deploy the object selected for deployment as well as any objects it hosts. This option is selected by default if the object is a host. If you are deploying an individual host object, clear the check box. Objects being deployed across multiple platforms are deployed in parallel.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

- **Include Redundant Partner**: Select this check box to also deploy an AppEngine's redundancy partner object. This option is selected and unavailable when the redundant engine has pending configuration changes or software updates.

4. In the **Currently deployed objects** area, select one or more of the following options. These options are not available if the selected object has not been deployed before.
   - **Skip**: If one of the objects you are deploying is currently deployed, selecting **Skip** makes no changes to the already-deployed object.
   - **Deploy Changes**: If one of the objects you are deploying is currently deployed, this option updates the object in question new configuration data. If you are redeploying an object, it is updated with new run-time data or new configuration data, depending on whether a change was made to a configured value, to a run-time value, and whether or not the **Preserve Runtime Changes** option is selected.
     - If both **Deploy Changes** and the **Preserve Runtime Changes** option are selected, values changed during run time are loaded as the object is redeployed. The new run-time value is used, even if you configure a new value in the IDE.
     - If **Deploy Changes** is selected but the the **Preserve Runtime Changes** option is NOT selected, new configured values set in the IDE are used, and will overwrite any changes made during run time.
   - **Redeploy Original**: If one of the objects you are deploying is currently deployed, this option deploys the same version as previously deployed. For example, use this option to redeploy an object that is corrupted on the target computer. If the **Preserve Runtime Changes** option is selected, the object is deployed with the last run time value, even if you change the configured value in the IDE.
   - **Force Off Scan**: If one of the objects you are deploying is currently deployed, this option sets the target object to off scan before deployment occurs.
   - **Preserve Runtime Changes:** Selecting this option will produce different results, depending on whether you have chosen **Deploy Changes** or **Deploy Original**, if values were changed at run time, and if configuration values were changed in the IDE.
     - If a value was changed in run time and **Deploy Changes** and this option are checked, the object is deployed with the last run time value, even if you change the configured value in the IDE.
     - If a value was not changed during run time and **Deploy Changes** and this option are checked, the object is deployed with the last configured value from the IDE.
     - If a value was changed in run time and **Redeploy Original** and this option are checked, the object is deployed with the last run time value, even if you change the configured value in the IDE.
     - If a value was not changed during run time and **Redeploy Original** and this option are checked, the object is deployed with the original configured value from the IDE. If you configured a new value in the IDE, the prior configured value is used.
5. In the **Currently undeployed objects** area, select the **Deploy New Objects** check box to start a normal deployment.
6. In the **Deploy Status Mismatch** area, select the **Mark as Deployed** check box to mark the object as deployed in the Galaxy. A mismatch happens when the object is previously deployed to a target node, but the Galaxy shows the object is undeployed. Clear this option to redeploy the object to the target node.
7. In the **Initial Scan State** area, select one of the following:
   - **On Scan**: Sets the initial scan state to on scan for the objects you are deploying. If the host of the object you are deploying is currently off scan, this setting is ignored and the object is automatically deployed off scan. When you are deploying multiple objects, the deploy operation deploys all of the selected objects "off-scan." After all of the objects are deployed, the system sets the scan-state to "on-scan."

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Objects can run only when both the host/engine is "on scan" and the object is "on scan." If either the host/engine or the object is "off scan," the object cannot run.

Always deploy Areas to their host AppEngines on scan. Because Areas are the primary providers to alarm clients, deploying Areas off scan results in alarms and events not being reported until they are placed on scan.

- **Off Scan**: Sets the initial scan state to off scan for the objects you are deploying. If you deploy objects off scan, you must use the AVEVA Operations Control Management Console Platform Manager utility to put those objects on scan and to function properly in the run-time environment.

  Note: The Operations Control Management Console controls the state of the host/engine. The Object Viewer controls the state of the objects.

  The default scan setting is set in the **User Default** settings in the **Configure User Information** dialog box. For more information, see Configure user preferences.

8. Click **OK** to deploy the objects. The **Deploy** progress box appears. If you see error messages, see Deployment error messages. When the deploy is complete, click **Close**.

Deployment error messages

If the object being deployed has configuration problems that were not known during configuration, the **Deploy** progress box shows messages.

The **Progress** dialog box also shows the affected instances and any error and warning messages. The target object can take any actions necessary to achieve a valid state, including changing attribute values provided during deployment.

WinPlatforms are the only objects whose configuration designates its deployment location. Deployment problems unique to WinPlatforms are the following:

- The target computer could not be found on the network. Ensure the WinPlatform was configured properly and the target computer is properly connected to your network.
- Another WinPlatform is deployed already to the target computer. Resolve this problem by undeploying the existing WinPlatform before deploying the new one.
- The target platform is running on the old version of the product. To resolve this problem, the user must upgrade the remote platform.
- If a WinPlatform object is deployed on a slow network and it does not respond to the IDE before the 30-second message time-out, a communication error occurs and the object is shown as not deployed. You may need to adjust the message time-out for the WinPlatform object to accommodate the slow network speed.

Publish managed InTouch applications

You can publish a managed InTouch application using the System Platform IDE. As part of this process, you can create a compressed, self-extracting package file that contains all relevant files and set-up procedures to install an InTouch application on another computer. For more information, see the *InTouch HMI Application Management and Extension Guide*.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Redeploy objects

Updating assets that are already deployed requires a redeployment of the asset. Redeploying an asset or a group of hierarchical assets can be done almost seamlessly, with very little downtime. When you redeploy an asset, a maximum of two scan cycles are needed between the time the existing asset configuration is undeployed and the asset with new configuration information is deployed in its place (typically, less than one second). This redeployment workflow provides two important benefits:

- Minimized downtime. While an asset is down, it does not provide status updates and does not respond to scripts or commands.
- Reduced points of failure. The existing configuration remains in effect until the new configuration is available.

This minimal downtime needed to redeploy assets ensures that uptime is maximized and the impact on operations is minimized. To ensure against a deployment failure due to communications or other error, the existing asset configuration remains deployed and onscan until the new configuration has been successfully deployed. Therefore, if a redeploy attempt fails for any reason, operations are not affected, as the existing configuration remains available within the production environment. The asset is not stopped or taken offscan until the new configuration replaces it. Since less than two scan cycles are needed for a redeploy operation, and this occurs only at the final stage of pushing out the new configuration, there is little risk of adverse effect on operations.

- Redeploying assets follows the same workflow as deploying assets. See Deployment overview for additional information. Redeployment occurs when you select and deploy the asset.

When assets with retentive attributes are redeployed, the Application Engine for the assets restores the retentive attributes to the values previously set at run time, if the **Preserve Runtime Changes** option is selected. See Deploy objects and Retentive attribute properties and behavior for additional information.

You may have an object with a Pending Update deployment state. The Pending Update state means the object changed since it last deployment. When you deploy those changes, the new object is marked as the last deployed version in the Galaxy.

To redeploy

1. On the **Home** ribbon, in the **Deploy** area, select **Deploy**.
2. Follow the procedure for Deploy objects.

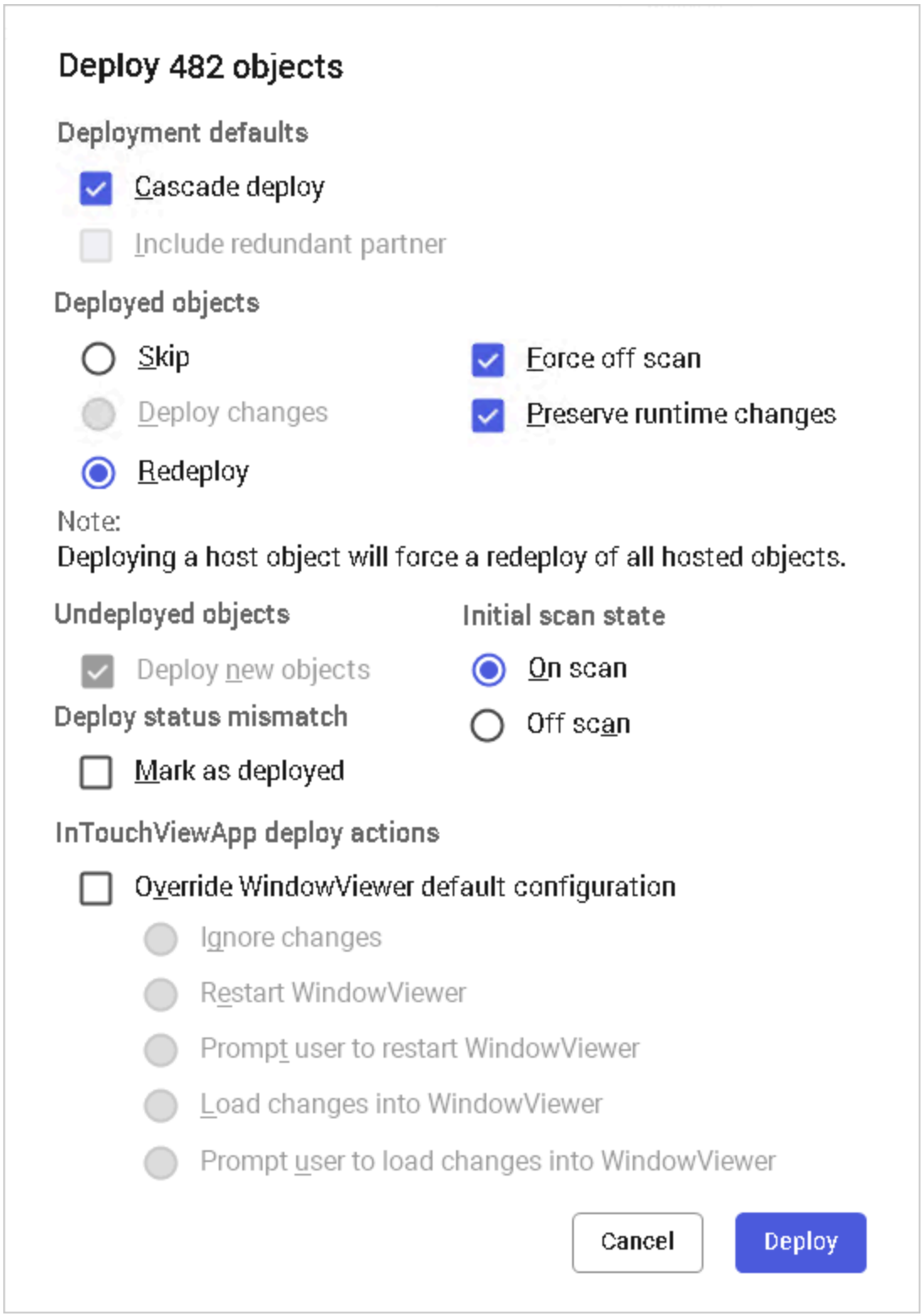
Redeploy the WinPlatform object

You can redeploy a WinPlatform object without undeploying and redeploying the objects hosted on WinPlatform. When you make changes to the WinPlatform object, you can propagate the changes by redeploying WinPlatform alone, without taking AppEngines and other objects hosted on the WinPlatform off-scan.

To redeploy a WinPlatform without redeploying hosted objects

1. Select the WinPlatform object that has updates pending.
2. On the **Home** ribbon, in the **Deploy** area, select **Deploy**. The **Deploy** dialog box appears.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.



3. Clear the **Cascade deploy** check box to prevent other hosted objects from being redeployed. By default, the check box is selected.

Note: It is not necessary to redeploy hosted objects if you are uploading changes only to the WinPlatform object.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

4. In the **Deployed objects** area, select the **Preserve Runtime Changes** option if you want to save overrides to configured settings that may have been made during run time. See Retentive attribute properties and behavior for more information on preserving run-time changes.

   For more information on deploying objects, see Deploy objects.

5. Select **Deploy**. The **Deploy** progress box appears. If an error occurs during deployment, an error message appears. For more information on error messages during deployment, see Deployment error messages.

   Important: The WinPlatform object will briefly shut down during the redeploy operation as checkpoints are synchronized. However, AppEngines and other objects remain deployed.

6. When deployment is complete, select **Close**

   Note: If the WinPlatform object hosts redundant AppEngines, the AppEngine may automatically fail over if the failover timeout is detected by the standby engine.

Retentive attribute properties and behavior

Some changes that are made to attributes at run time are automatically tracked and saved. These attributes are called "retentive attributes." Changes made at run time to retentive attributes are automatically retained. If an object with a retentive attribute is undeployed and subsequently redeployed with the **Preserve Runtime Changes** option selected in the Deploy dialog, the last run time value is loaded into the retentive attribute. If the **Preserve Runtime Changes** option is NOT selected, the object is redeployed with its configured setting, not the value set during run time. See Deploy objects for additional information.

Changes to retentive attributes are retained as long as the new attribute value is different than the previous saved value, and data quality of the attribute is good. You do not have to do anything to mark an attribute as a retentive attribute. Attributes are automatically set to be retentive if all of the following are true:

- The attribute is checkpointed.
- The attribute is User- or Object-Writeable.
- The attribute does not contain an input extension or input/output extension.

At every checkpoint period, the AppEngine looks for changes to retentive attributes on objects that it hosts. A change made at run time to a retentive attribute is saved on the node where the AppEngine is running. When any of the retentive attributes associated with the AppEngine have been changed, the changed values are loaded into memory. After several checkpoint periods have passed, the changed values in memory are written to disk, and memory is cleared. In this way, only values that have changed are retained. However, retentive attribute run time changes are not uploaded to the IDE. Therefore, if an object that contains a retentive attribute is undeployed and moved to a different node or even to a different AppEngine, the run time changes are not preserved when the object is redeployed.

For redundant pairs, changes will be preserved, even if the redundant partner is on a different engine or node. If the value of a retentive attribute changes and it is part of a redundant pair, the changed value will be sent to the redundant partner during checkpoint synchronization, if the following are both true:

- Attribute data quality is good
- The new attribute value is different than the previous value

See Redundancy for more information.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Undeploy objects

You may need to undeploy one of more objects. Undeploying removes one or more objects from the run-time environment.

Before you start, you need to select the object or objects you want to undeploy in the IDE.

Before you delete or restore a Galaxy, undeploy all objects in the Galaxy.

Undeploying can fail if the target object has objects assigned to it. Make sure you select **Cascade Undeploy** in the **Undeploy** dialog box.

To undeploy

1. From the **Home** ribbon, select **Undeploy**. You can also right-click on the selected object(s) and then click **Undeploy** in the context menu. The **Undeploy** dialog box appears.

   At the top of this dialog box, the number of objects to be undeployed is shown. You can select a single object in **Application view** and, if you selected **Cascade Undeploy** and other objects are assigned to the selected object, the total number of objects appears in this box.
2. Select one or more of the following options. Depending on the object(s) you select, some options may not be selectable.
   - **Cascade Undeploy**: Select to undeploy the selected object as well as any objects it hosts.
   - **Include Redundant Partner**: Select to also undeploy an AppEngine's redundancy partner object.

     Note: The AppEngine in a redundant pair that was configured as the Primary can be undeployed alone because objects hosted by it run on the deployed Backup AppEngine, which becomes Active. See Working with AppEngine Redundancy for more information.
   - **Force Off Scan**: If one of the objects you are undeploying is currently on scan, selecting **Force Off Scan** sets the target object to off scan before undeployment. If you do not select **Force Off Scan** and the target object is on scan, the undeployment operation fails.
   - **On Failure Mark as Undeployed**: Marks the object as undeployed in the Galaxy when the object targeted for undeployment is not found.

Upload run-time configuration changes

You can upload run-time configuration changes to the Galaxy database. This lets you set new default attribute values, based on observations of run-time behavior.

Note: A node cannot upload run-time changes to the Galaxy if the version of Application Server it is running is older than the version running on the GR node.

Certain attribute values can be set in the configuration environment, but they can also be changed at run time. As a result, the values of these attributes can differ between the run-time and configuration environments. These types of attribute writeability are:

- Calculated
- Calculated Retentive
- Object Writeable
- User Writeable

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

For example, you create an object with a User Writeable attribute called `myInteger` and specify an initial value of 30 in the Object Editor.

Then you deploy the object. At run time, you write a new value to `myInteger` of 31. If you redeploy this object, the original value of 30 overwrites any value assigned during run time. To avoid losing changes made during run time, upload changes before redeploying an object.

Objects whose configuration are successfully uploaded have a new version number and a change log entry for the upload operation. The run-time object's version number also has a new version number. That version number matches the version in the configuration database.

Upload changes to auto assigned objects and object wizard instances

In most cases, uploading run-time changes sets new default values for the affected objects in the GR. However, if your galaxy uses auto assigned I/O references, or if the attribute values you changed were in objects derived from object wizards, run-time configuration changes are not saved as new default configuration values. The run-time changes are saved instead as overrides to the default attribute values in the GR.

- Run-time configuration changes to auto assigned I/O references are uploaded as overrides to the original I/O assignments. I/O references will continue to be auto assigned and the run-time changes are retained when the objects are redeployed. See Upload runtime configuration changes for auto-assigned objects for additional information.
- If you upload a run-time change to an instance derived from an object wizard, the change does not overwrite the configured object wizard value. Instead, the change is saved as an override to the default value configured for the attribute in the object wizard.

To upload run-time changes to the galaxy

1. Select one or more objects in the **Model** view or **Deployment** view. For example, you could select an entire hierarchy from AppEngine down.
2. On the **Home** ribbon, in the **Runtime** area, select **Upload changes**.
3. The runtime attributes of the selected objects are copied over those in the Galaxy database, with the following exceptions:
   - Auto Assigned Objects: The new run-time values are set as overrides to the default values in the Galaxy database.
   - Objects derived from Object Wizards: The new run-time values are set as overrides to the default values in the Galaxy database.

Upload limitations

Certain run-time changes to the Galaxy cannot be uploaded. Before uploading changes, make sure the selected objects are:

- Not checked out
- Not edited and checked in since last deployment or upload
- Not in Pending Update state

If you select an object that is currently checked out to you, a warning appears during run-time upload. If you

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

continue, you lose all configuration changes you made to the checked out object. The Galaxy performs an Undo Check Out operation on it before the run-time attributes are copied to the Galaxy database.

Note: You cannot upload run-time changes for objects checked out to other users.

Undeployment situations

The following situations occur in these specified undeployment scenarios:

- After undeploying a WinPlatform, only the Bootstrap software remains on the target computer. All other WinPlatforms are notified of the undeployment of the WinPlatform and stop trying to communicate with it over the network.
- Alarm Clients know immediately that an area is undeployed and is no longer available. They remove the area from selection lists. Alarms associated directly with the area (not as a result of containment) are immediately removed from current alarm views.
- Undeploying an object that has Pending Updates status removes that status. It is now marked as undeployed.
- If cascade undeploy fails on one object, then the undeploy continues to the extent possible on other objects. The entire operation is not terminated because the undeploy fails for one object. However, you will not be able to undeploy a container object if one or more of the objects assigned to it cannot be undeployed (for example, if an object is open in the Object Editor).
- Redundant AppEngines: Assume an ApplicationObject is hosted by the Active AppEngine in a redundant pair and a number of subscriptions is configured in that ApplicationObject that refers to items in a DIObject. If you undeploy the ApplicationObject in question, the items are not removed immediately from the item count of the DIObject. How fast those items are removed depends upon the value of the **Maximum time to maintain good quality after failure** option (Redundancy.StandbyActivateTimeout attribute) on the **Redundancy** page in the AppEngine's editor. This behavior does not apply to the undeployment of ApplicationObjects hosted by non-redundant AppEngines. For more information, see Working with AppEngine Redundancy.

Associate all galaxy graphics with an InTouchViewApp

Associating all Galaxy graphics with an InTouchViewApp template enables deployed and published InTouch applications to execute "show graphic" requests made of any graphic in the Galaxy without having to embed them in the application.

- The ShowGraphic() function uses the graphic as a parameter. Associating all Galaxy graphics ensures that the graphic is deployed and available at run time whether or not it is referenced by InTouchViewApp.
- Associating all Galaxy graphics ensures that template graphics referenced by the ShowGraphic() function are deployed and available at run time.

  Note: The term "graphic" includes any graphic or client control present in the Visualization folder and any graphics owned or inherited by templates and instances.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Publish and deploy behavior when all galaxy graphics are associated with an InTouchViewApp

Associating all Galaxy graphics with an InTouchViewApp (ITVA) causes the following deployment and publishing behavior:

- The associate all Galaxy graphics option—"Include all Galaxy graphics"—changes only the ITVA template.The option is inherited by the template instances, and changes the template only if none of its instances are deployed.
- A deployed ITVA with the "Include all Galaxy graphics" option enabled will be marked "pending change" under the following scenarios

Scenario	Operations performed
Creating graphics	Create new graphics in the Visualization folder Check in new graphic(s) in an automation object Import automation object with graphic(s) Import Visualization folder graphics Create a new derived template or instance object that inherits graphics GR Load new graphics into the galaxy
Modifying graphics	Edit and check-in graphic(s) Modifying graphics through an Import operation Modifying graphics localization through an Import operation
Deleting graphics	Delete graphics from the Visualization folder Check in an application object from which graphics have been deleted Delete template or Instance object with graphics
Renaming graphics	Rename graphics in the Visualization folder Check in an application object, any of whose graphics have been renamed
Rename object	Rename Instance or Template which has owned or inherited graphics
Object containment	Containment of automation objects that own or inherit graphics

- Deploying an ITVA with the "Include all Galaxy graphics" option enabled for the first time will deploy all checked-in graphics in the Galaxy. Client controls will be deployed whether or not they are embedded in a graphic.Undeploy behavior does not change.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

- Deploying changes with an ITVA that has the "Include all galaxy graphics" option enabled will deploy the Galaxy graphics changes since the previous ITVA deployment.
- Redeploying the original ITVA that has the "Include all Galaxy graphics" option enabled will not account for graphics that were added, deleted, renamed or modified since the original deployment.
- Publishing an ITVA with the "Include all galaxy graphics" option enabled will copy all the checked-in galaxy graphics to the published application folder. The ITVA will retain its "pending change" status as this is not a deploy operation.
- The "Include all Galaxy graphics" enabled setting will persist through a Galaxy backup and restore operation, but will not persist through an ITVA import or export, or through GR dump and GR load operations.

Configure the associate galaxy graphics option

Access the option through the InTouchViewApp right-click context menu.

Note: The **Associate graphics** option will be disabled if the $InTouchViewApp template has not been initialized or is checked out.

To include Galaxy graphics with an InTouchViewApp

1. Right-click the InTouchViewApp template you wish to configure. The InTouchViewApp context menu appears.
2. Select the **Associate Galaxy Graphics** menu item. The **Associate Galaxy Graphics** dialog box appears.
3. Click the check box to include all galaxy graphics and click **OK**. A status box displays check in progress.

   If an InTouchViewApp instance is deployed when you attempt to modify the **Include all Galaxy graphics** setting, you will see an information message to undeploy all deployed instances.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Run

The topics in this section describe how to view run time data and performance for deployed Application Objects that you select.

For more information about viewing run-time data, see:

- Application Server Object Viewer Object Viewer overview
- AVEVA OMI Run ViewApps Level2_Run Applications
- InTouch HMI View Applications at Run Time Viewing Applications at Run Time

Process late data at run time

Data received by Application Server run-time clients from the Telemetry Server or other sources may arrive late, especially when using publish/subscribe models such as MQTT, with or without Sparkplug B. "Late data" refers to process data that is received after the time of the actual event. For example, an OMI ViewApp may receive data along with its VTQ timestamp x number of minutes or even hours after the triggering event has occurred.

Application Server allows late data received through a DI Object to backfill historical data in the Historian while presenting live data to run-time visualization clients, such as OMI ViewApps. Note that value historization is done only for attributes that have the History extension enabled. Late data received from direct references or through Multi-Galaxy Communication is not sent for historization.

When connection to Application Server from the data source is lost, historical data is backfilled once the connection is restored. References to ScanGroup items directly in scripts allow late data to be processed. For example, "me.Int1 = DiObject.Scangroup.Item1" assigns late data values to user-defined attribute (UDA) Int1.

Late data received from DI objects referenced through the I/O extension is not backfilled. In the meantime, Application Server provides run-time clients (AVEVA OMI and/or InTouch HMI workstations) with last known good value, and resumes the live data feed when the connection is restored. Historical data is sent directly to the Historian and does not affect either Application Server or the run-time clients.

While the Historian is processing the historical data, Application Server uses a Quality and Status animation, "Processing Historical Data," to notify the run-time clients that this is occurring, while continuing the live data feed to the run-time clients. At the end of the historical data processing, the current data/states/values are determined based on either the live data or, in the absence live data, by using the last-known good values from the historical data. Trend pens, the Trend Client, and any other visualized data that relies on historical data are updated when historization is finished.

The historical data is historized appropriately and can be later queried as if it had been received in order.

Object Viewer overview

Object Viewer provides performance and diagnostic information about ApplicationObjects. You can use Object Viewer to show performance parameters, resource consumption, and reliability measurements of objects you select.

In addition to viewing an object’s data value, data quality, and the communication status of the object, you can

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

also modify some of the object's attributes for diagnostic testing. Modifications can include adjusting timing parameters and setting objects in an execution or idle mode.

Object Viewer runs with minimal System Platform and operating system requirements.

Using Object Viewer you can:

- View the data type, data quality, data value, timestamp, and communication status of ApplicationObject attributes.
- Do diagnostic testing on ApplicationObjects.
- Modify selected ApplicationObject attributes.

Object viewer availability

Object Viewer is a common component of any Galaxy application and is available from any workstation in the application with a deployed WinPlatform. You do not need to install the utility on each workstation.

Start object viewer

Since Object Viewer is a run time tool, the Galaxy must be deployed before you can view objects. You can use Object Viewer from any workstation that is part of your Galaxy application.

To start Object Viewer

1. Start Object Viewer from the System Platform IDE or the Windows Start menu.

   From the System Platform IDE, do one of the following:

   - Select a deployed object, then select **Object Viewer**, located in the **Home** ribbon **Runtime** group.
   - Right-click a deployed object and select **View in Object Viewer** from the Context menu.

   From the Windows Start menu:

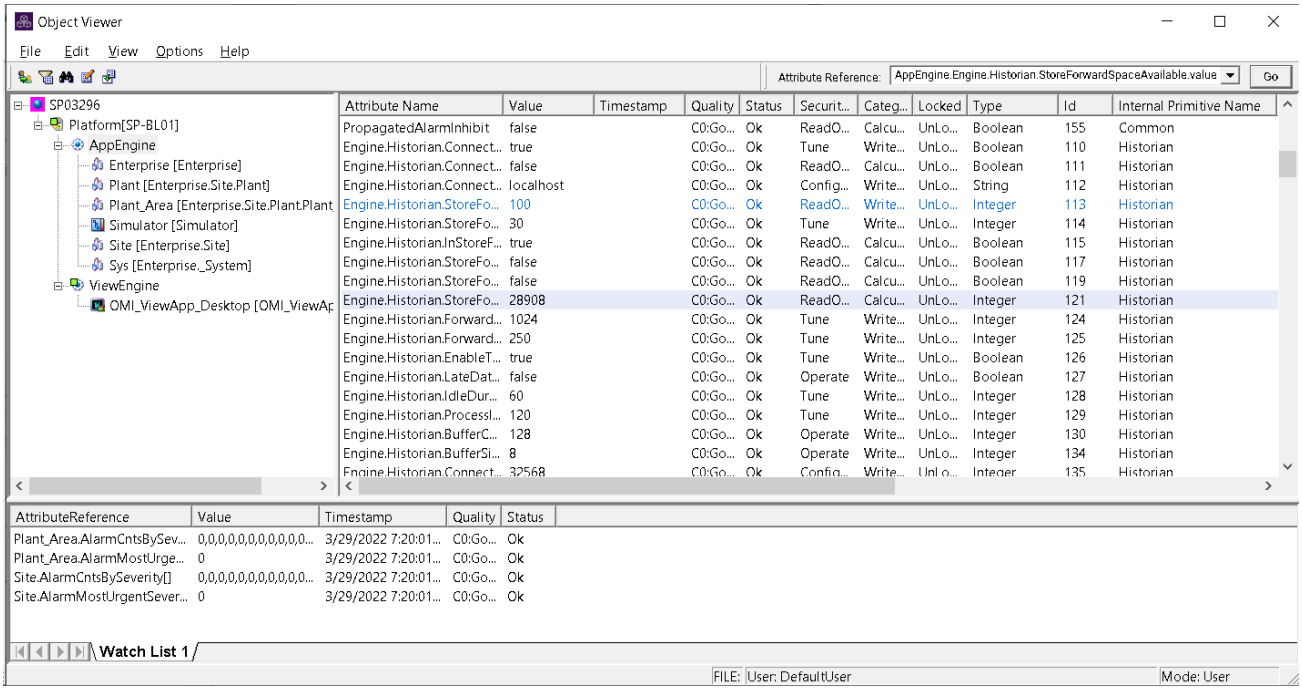
   a. Click **Start**, navigate to **AVEVA,** and click **Operations Control Management Console**.
   b. In the console tree, expand the **Platform Manager** to show the list of Galaxies, and then expand a Galaxy to show a list of platforms.
   c. Right-click the desired platform and click **Launch Object Viewer**.

Navigate in object viewer

The main window of the Object Viewer is composed of three areas:

- Console tree on the left
- Details pane on the right
- Watch window bottom

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.



The console tree works like Windows Explorer. Use the plus (**+**) and minus (**-**) signs to expand or hide the nested list items shown in the pane. When you select an object in the console tree, its associated attributes and their properties are shown in the details pane.

Drag a specific set of attributes of interest to a watch window to monitor their **Quality**, **Status**, **Timestamp**, and **Value** as you manipulate them and other attributes that affect their data values. You can save the contents of a watch window to a file for future reference.

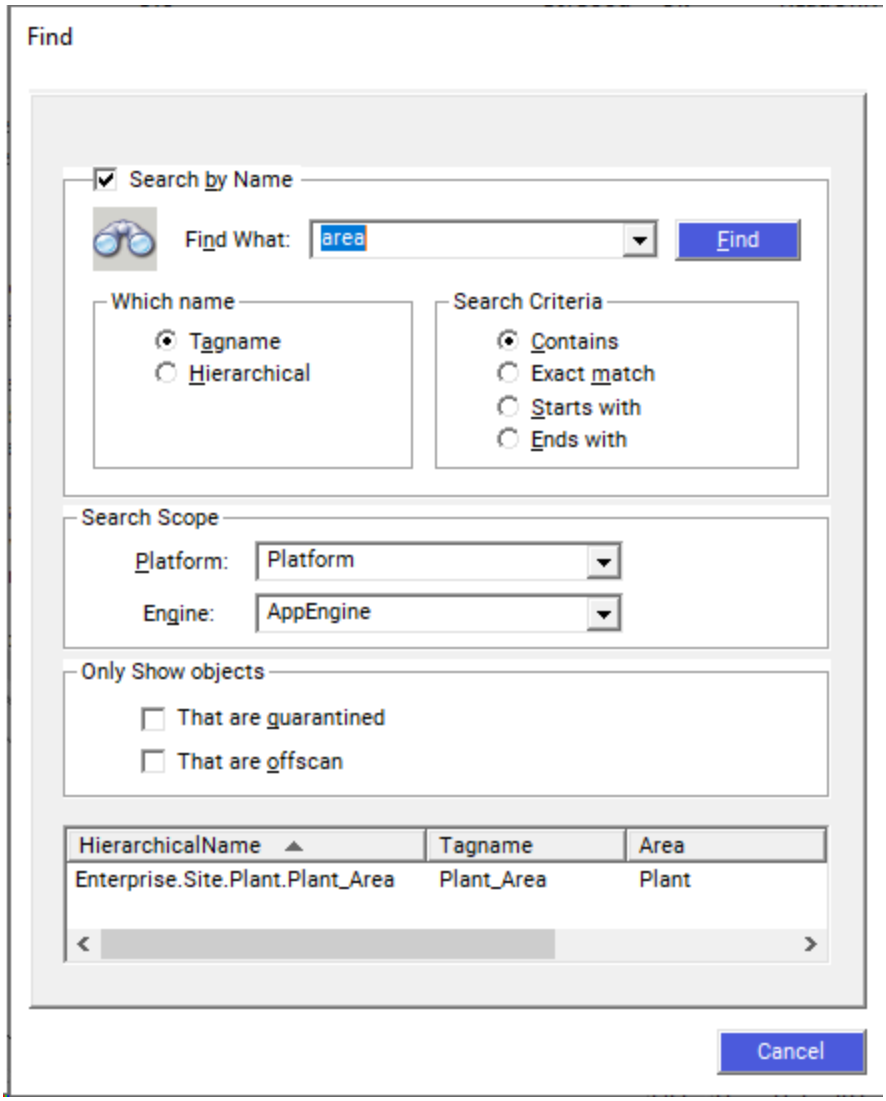
Find objects

You can use the **Find** dialog box to search for an object in your Galaxy application.

To find an object

1. On the **Edit** menu, click **Find Object** to open the **Find** dialog.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.



2. Enable the **Search by Name** checkbox to search by name. You can disable name search if you want, for example, to search for all objects that are quarantined or offscan.
3. With **Search by Name** enabled, select the **Find What** field and enter the text you want to find.
   - Under **Which name**, select either **Tagname** or **Hierarchical name**.
   - In the **Search Criteria** field, select a search option (**Contains**, **Exact match**, **Start with** or **Ends with**).
4. In the **Search Scope** area, select a Platform and Engine.
5. To limit the search scope, you can select either or both of the following options:
   - Only objects **that are quarantined**
   - Only objects **that are offscan.**
6. Click the **Find** button.

Close object viewer

By default, when you close the Object Viewer, your settings are not saved, including the attributes you selected for the **Watch** window. You can choose to save the watch windows you create. For more information about

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

saving watch windows, see Save a watch window.

To close the Object Viewer

- On the **File** menu, click **Exit**.

View attributes

Using Object Viewer, you can view the attributes of all ApplicationObjects in your Galaxy application from any workstation used in the application.

View object attribute data

All AutomationObjects consist of attributes and common properties such as data type, initial value, and value range. Each object's set of attributes appears in the details pane when you select the object in the console tree.

To view an object's attribute data

1. Start Object Viewer. Object Viewer's main window appears. The object in focus in the IDE or Platform Manager is put into focus in the console tree of the Object Viewer.
2. To view another object's attributes, select the object in the console tree.

Quality

Every readable attribute is assigned a quality property that matches one of four categories.

Quality	Description
Good	Data is good and can be used.
Uncertain	Data is questionable, but can be used.
Initializing	Data is not yet available, but will be soon.
Bad	Data cannot be used.

Status

The status of an object's attribute describes whether a particular Read or Write request succeeded or failed. Possible status conditions include:

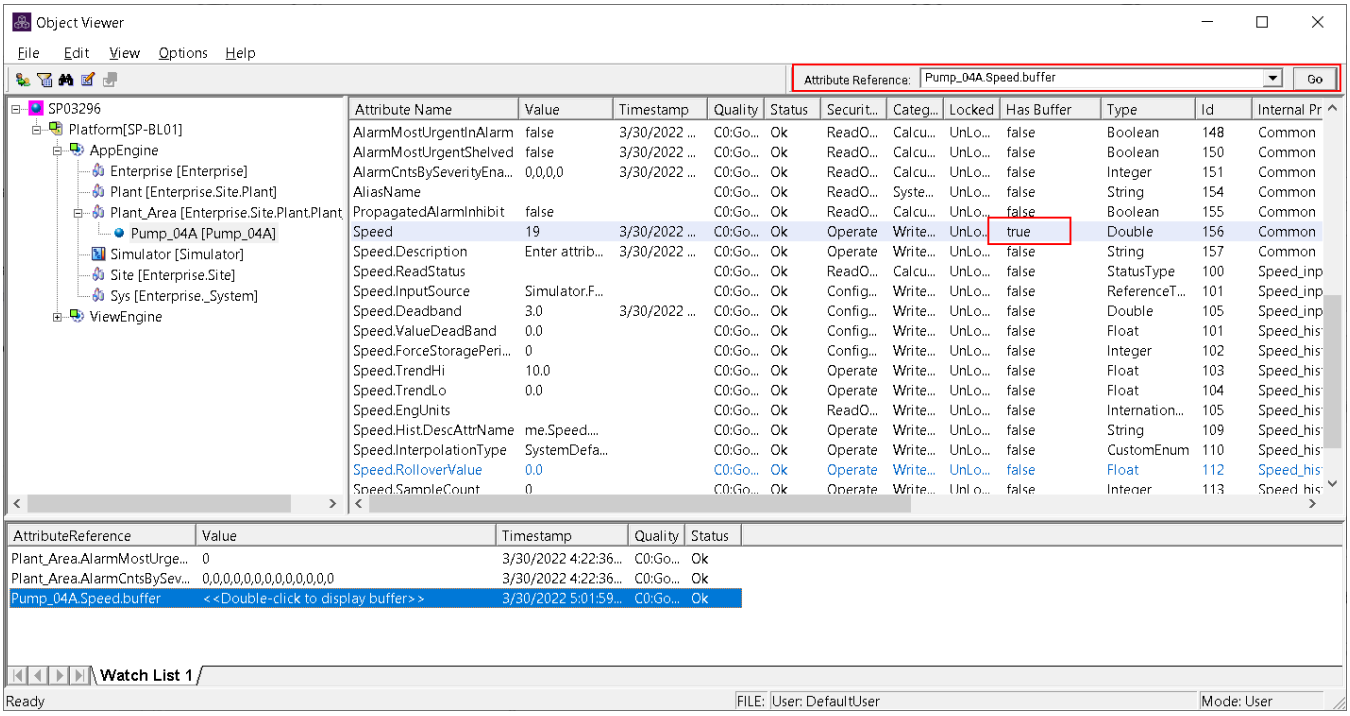
Status	Description
OK	Operation completed successfully.
Pending	Request is received, but is not yet complete. A transitory status.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Status	Description
Warning	Operation completed successfully, but with some warning condition. Applies only to Write actions.
Communication Error	Request failed due to an error communicating with the target ApplicationObject.
Configuration Error	Request failed due to an error in configuration.
Operational Error	Request failed due to an operator error.
Software Error	Request failed due to an internal software error.
Security Error	Request failed due to insufficient security access rights.
Other Error	All other errors.

Has buffer

The Object Viewer **Has Buffer** column indicates if an attribute has a buffer property. The value displayed in the column is True if the attribute has a buffer property or False if the attribute has no buffer property.



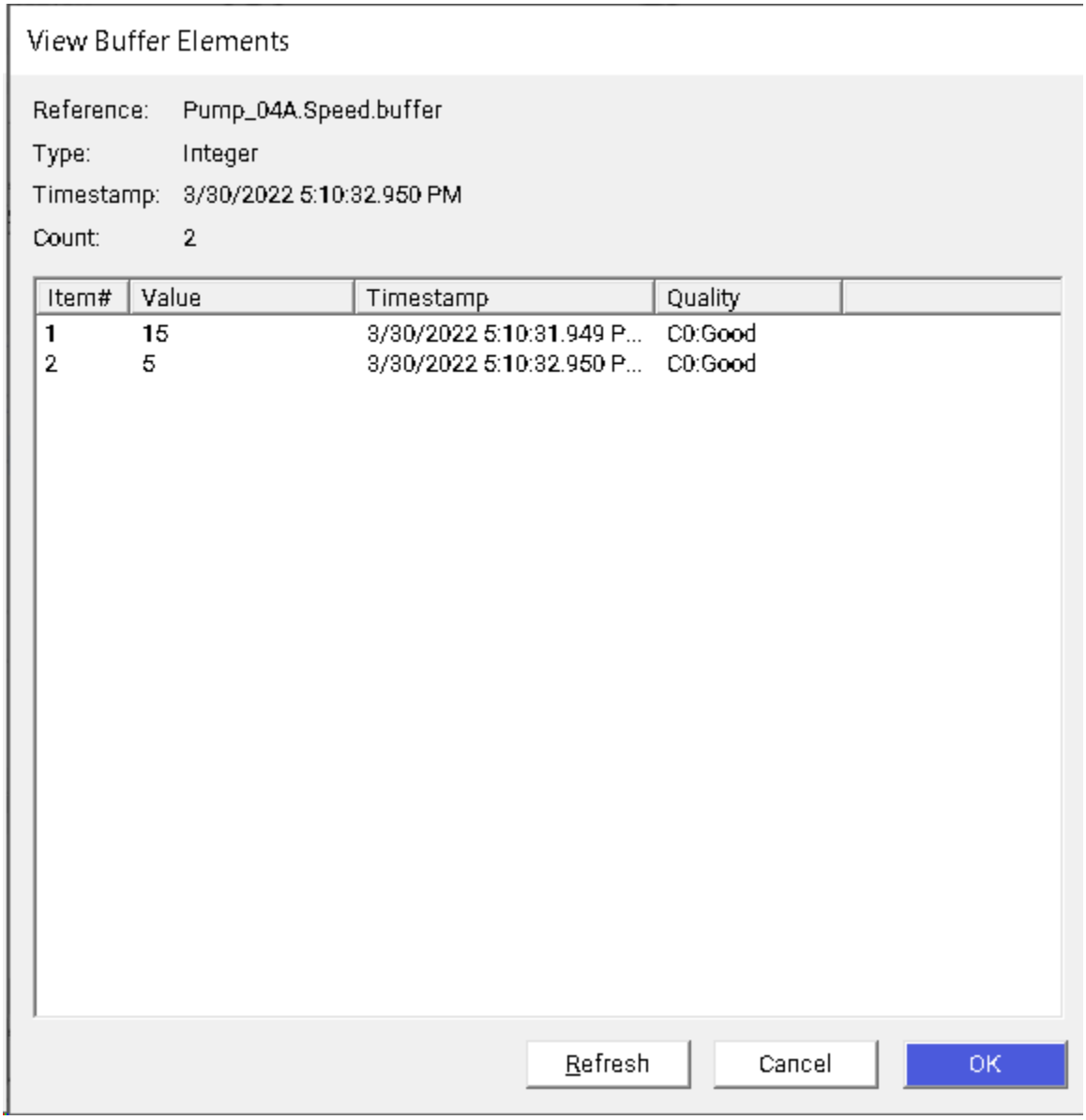
To view the value of the Buffer property

Use the **Attribute Reference** dialog box to add the buffer property to the watch list. You can either:

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

- Enter the property name in the dialog box, using the syntax **<attributename>.buffer**.
- Use an alternate syntax, **<attributename>.property(Buffer)**. This syntax should be used for subscribing to attributes on a DI Object.

Double-click the attribute in the watch window. The **View Buffer Elements** dialog box appears.

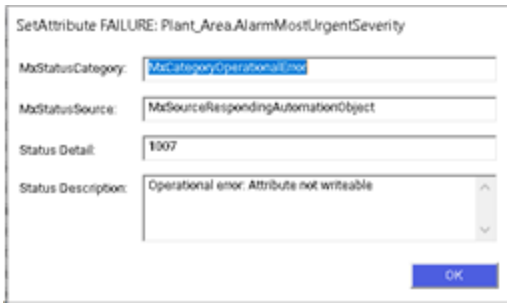


The **View Buffer Elements** dialog box header displays the Attribute Reference, Type, Timestamp and Count. A details window displays the Value, Timestamp, and Quality (VTQ) columns for each item.

Failed requests and errors

When a failure occurs during a request, the details of the error appear in a dialog box, which contains the source of the error and a description of the error.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.



Communication errors

Communication errors include the following:

Source	Description	Reason
Requesting NMX	Request Timed Out	A message was not processed in a timely manner.
Requesting NMX	Communication Error: Platform communication error	Attempt to get or set an attribute from an ApplicationObject on a remote platform with the network down between the two platforms.
Responding NMX	Communication Error: Engine communication error	Attempt to get or set an attribute from an ApplicationObject on a remote engine that is currently not running, but deployed.
Responding NMX	Communication Error: Engine communication error	Attempt to get or set an attribute from an ApplicationObject on a remote engine, but that remote engine is currently undeployed while the remote platform is deployed.
Responding Automation Object	Index out of range	Attempt to get or set an attribute from an ApplicationObject attribute that is an array, but specified an array index that is out of range.

Configuration errors

Configuration errors include the following:

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Source	Description	Reason
Responding LMX	Invalid Object ID	Attempt to get or set an attribute from an ApplicationObject on a remote engine, but that remote ApplicationObject is currently undeployed while the remote engine is deployed.
Requesting NMX	Invalid Platform ID	Attempt to get or set an attribute from an ApplicationObject on a remote platform, but that remote platform is currently undeployed.
Requesting LMX	Invalid reference	Attempt to get or set an attribute from an ApplicationObject that is not yet configured in the system. A bad reference.
Responding NMX	Invalid Engine ID	Attempt to get or set an attribute from an ApplicationObject on a remote engine, but that remote engine is currently undeployed while the remote platform is deployed.

Operational errors

Operational errors include the following:

Source	Description	Reason
Responding Automation Object	Attribute not writable	Attempt to set an attribute in an ApplicationObject that cannot be written to.
Responding Automation Object	Write access denied.	Attempt to set an attribute in an ApplicationObject that can be written to, but that the user cannot write to. This can be an operational error (the attribute is read only) or a security error (the current user does not have permission to write to the attribute).

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Source	Description	Reason
Responding Automation Object	Software error: The data to be set was to the wrong data type.	Attempt to set an attribute in an ApplicationObject can be written to, but with a data type that cannot be written to because it cannot be coerced. For example, trying to write a Boolean value of True to an attribute of type QualifiedStruct or QualifiedEnum.

Sort data

You can sort the data shown in the details pane in either ascending or descending order by clicking any column heading. You can show hidden attributes in the details pane by selecting the **Include hidden** command from the **Options** menu. For example, click the **Attribute Name** heading to sort the list by attribute name. Attribute names appear in ascending or descending alphabetical order.

Hidden attribute names include leading underscores and appear at the top of an ascending order list. For example, all the attributes in the following figure are hidden attributes, including the _AdviseOnlyActiveEnabled attribute, which indicates if Advanced Communication Management is active.

Attribute Name ▲	Value	Timestamp
__Attr_Name_List_1	<AttrXML/>	3/30/2022 4:22:37.152 PM
_AdviseOnlyActiveEnabled	true	
_AggAlarmUpdate	-1	
_AlarmModeEnum	Enable,Disa...	
_AlarmMostUrgentInit	true	
_Attributes	No Data	
_Category	ObjectAppl...	
_CategoryEnum	EnginePlatf...	
_ExecutionRelativeOrderEnum	None,Befor...	
_ExternalName		3/30/2022 4:22:37.136 PM
_ExternalName		3/30/2022 4:22:37.136 PM
_InheritedAttrListNames	__Attr_Nam...	
_InternalName	Common	3/30/2022 4:22:37.136 PM
_InternalName	UserDefined	3/30/2022 4:22:37.136 PM
_TempCntsBySeverity	0,0,0,0,0,0,0,...	
_TempCntsBySeverityEnableShelved	0,0,0,0	
_TempMostUrgentAcked	true	
_TempMostUrgentInAlarm	false	

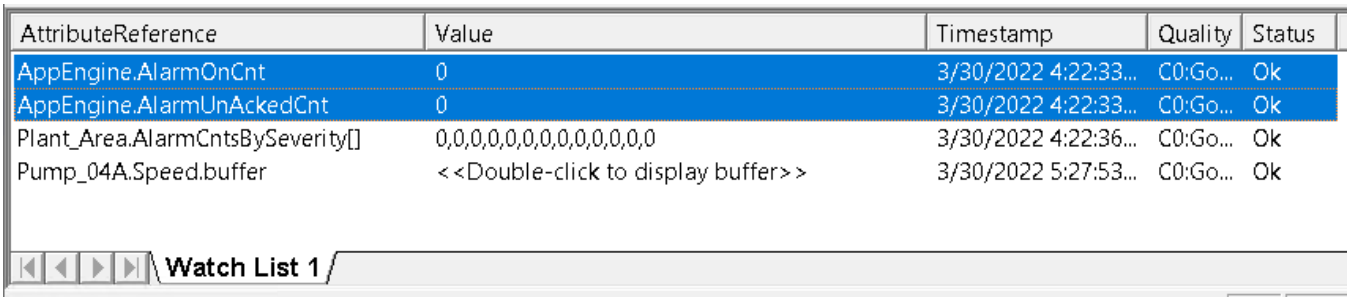
© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

View attributes in a watch window

You can use the watch window to view an attribute or a group of attributes apart from the entire list of attributes.

The attributes in the watch window are listed by their **AttributeReference**, which shows which object the attribute belongs to.

For example, the AppEngine object contains AlarmOnCnt and AlarmUnAckCnt attributes. When you add these to a watch list, the object tag name is prepended to the reference string.



To view selected attributes in a watch list

1. Select an ApplicationObject from the console tree. A list of its attributes appears.
2. Add an attribute to the watch list.
   - In the details pane, right-click an attribute to place it on the watch list. Click **Add to Watch**. The attribute in the **AttributeReference** format, along with its data, appears in the watch list.
   - Right-click in the watch window and click **Add Attribute Reference**. In the **Add Attribute Reference** dialog box, type the attribute name in the following format: Object Reference.Attribute Reference. For example, enter SystemArea.ScanState to view the ScanState attribute of the SystemArea object.
3. To add a block of attributes to the watch window, hold down the SHIFT key and click the left mouse key while selecting the first and last attribute of the block.
4. To add a random list of attributes to the watch window, hold down the CTRL key and click the left mouse button while selecting attributes.

Aveva OPC UA client service

You can use an instance of the Aveva OPC UA Client Service hosted by your Galaxy to subscribe to attributes hosted by an OPC UA server. Using an instance of the Aveva OPC UA Client Service in Object Viewer requires the use of the specific service instance name as a prefix.

For example, using an instance of the UA Client Service with **Scope Name** of "OPCUAClient_010'" would be specified as follows:

```
OPCUAClient_010:AppEngine.AlarmOnCnt.value
```

Add a watch window

By default, a single watch window appears in the Object Viewer. You can add more watch windows to view different groups of attributes or view attributes of different ApplicationObjects.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Note: If attributes already appear in the watch window, right-click away from the list of attributes to open the correct shortcut menu.

To add a watch window

1. Right-click in the watch window and click **Add Watch Window**. A new watch window appears with the default tab name, Watch List 2.
2. To rename the new watch window, right-click it and click **Rename Tab**. Type the name of the watch window in the **Rename Tab** dialog box and click **OK**.
3. Add attributes to the watch window or add another watch window.
4. Move from one watch window to another by clicking the tabs.

Save a watch window

You can save a watch window and reuse it each time you start Object Viewer.

To save a watch window

1. To save a watch window, either:
   - On the **File** menu, select either either **Save** (for an existing watch window file) or **Save As**.
   - Right-click in the **Watch List** and select either **Save** (for an existing watch window file) or **Save As**.
2. When the **Save As** dialog box opens, select a folder path in which to save the watch window.
3. Click **Save**. The watch window is saved as an XML file.

Remove attributes from a watch window

The attributes that you add to watch windows remain until you remove them or until you close Object Viewer. For example, if you are viewing the attributes of an AppEngine and place some of its attributes in a watch window, those attributes remain even when you select another object in the console tree.

Note: This action does not delete the attribute from the object. The attribute is viewable in the details pane.

To remove an attribute from a watch window

- In the watch window, right-click the attribute you want to remove and click **Remove from Watch**. The attribute is removed from the watch window.

Modify attribute values

You can modify an attribute's value for diagnostic testing. You must have the proper security permissions to do this and the attribute's value must also be changeable.

Before making modifications to any attribute, you must be familiar with the attributes and their functions. Also, you must understand the effects that changing attribute values can have on your Galaxy application.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Attribute properties

Each data point of an attribute is associated with a unique VTQ identifier (Value, Time, and Quality). Only the value can be modified, provided that the attribute is user-writable. When value is modified, a new VTQ is generated.

When you try to modify an attribute value, a dialog box appears that shows the type (datatype) and current value of the attribute. Use this information as a guide for continuing.

Before modifying an attribute value, check its category shown in the **Category** column to determine whether modification is possible.

Category property

An attribute's Category property determines if the attribute is writable. The only categories that you can write to are:

- Writeable_U
- Writeable_US
- Writeable_UC
- Writeable_USC
- Writeable_UC_Lockable
- Writeable_USC_Lockable

The letters included in the Writable categories mean the following:

Letter	Description
U	UserSets are OK. You can modify these in the Object Viewer.
S	SupervisorySets are OK. You can modify these only with an application function. You cannot modify them in the Object Viewer.
C	Configtime time sets only. You can modify these only when you are configuring the object. You cannot modify them in the Object Viewer.

Value property

You can modify the value property of an attribute shown in the **Value** column, if the attribute is user-writable (see Category property). The datatype of the attribute is shown in the **Type** column, and indicates the kind of value that you can assign to the attribute, for example, Boolean, integer, double, etc.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Timestamp property

The timestamp of an attribute is the Time element of the VTQ packet. It represents the time of the last recorded change to the attribute, and is shown in the **Timestamp** column. The **Timestamp** cannot be modified, or turned on or off. An attribute will have a timestamp if the value in the **Has Quality/Timestamp** column is true.

Modify an attribute value

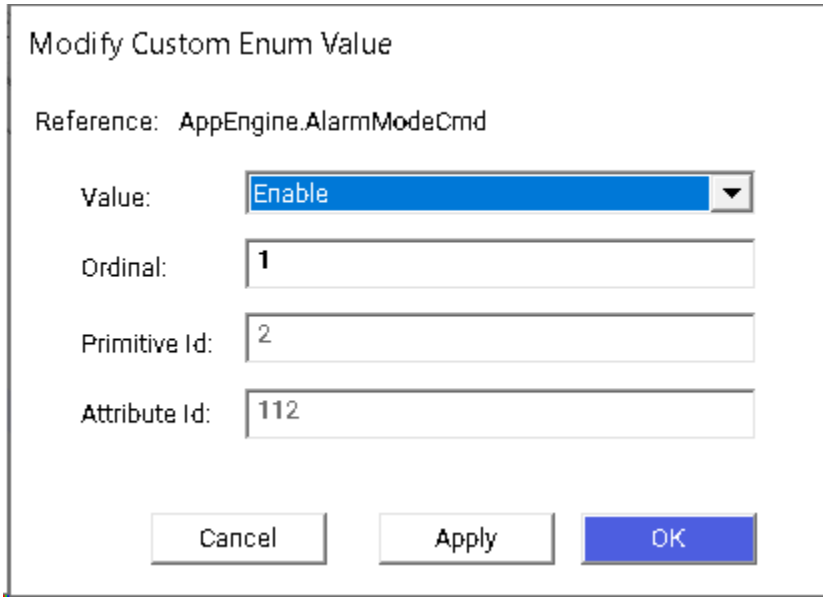
You can only modify attribute values if you have the proper privileges and the attribute is writable. Check the **Category** value in the details pane for the attribute you want to modify. You cannot modify Calculated and SystemWriteable attributes.

Some attribute values affected by the changed attribute are updated only after you select the attribute. For example, an object's ScanStateCmd controls the object's scan state, with True being OnScan, False being OffScan. The object also has a ScanState attribute that indicates the current scan state of the object.

Assuming ScanStateCmd and ScanState are False, change ScanStateCmd to True. Notice that the object's ScanState attribute value remains False. Select the ScanState attribute. Its value changes to True.

To modify an attribute value

1. Select the object that contains the attribute you want to change.
2. Right-click the attribute and click **Modify**. A **Modify <data type> Value** dialog boxes appears, related to the data type for the selected attribute.



3. Modify the values as needed.
4. Click **Apply** and click **OK**. The new value appears in the **Value** column of the details pane for that attribute.

Modify an attribute from the watch window

You can change an attribute's value while in a watch window.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Important: Attributes in the watch window update immediately. Attributes in the details pane update when you click them.

To modify an attribute from the watch window

1. In the watch window, double-click the attribute. A dialog box related to the attribute's data type appears.
2. If the attribute is writable, enter or select the new value for the attribute.
3. Click **Apply** and click **OK**. The new value appears in the **Value** column of the watch window for that attribute.

View changes dynamically

Attribute values shown in the details pane are static. You must click on an attribute in the details pane to update the display.

You may have a configuration in which the Galaxy Repository node was upgraded, but other nodes were not upgraded. If you use ObjectViewer on the upgraded node to view information for attributes from a non-upgraded node, the time stamps should not be considered as valid source times. In this case, the time stamps reflect the time of the local node (the one running ObjectViewer) for the attributes running on the non-upgraded node.

Attribute values shown in the watch window are updated immediately when the attribute's value changes.

For example, objects have two common attributes, ScanStateCmd and ScanState. ScanStateCmd controls the scan state of the object (OnScan or OffScan). ScanState shows the last known scan state of the object.

If you place the ScanState attribute in a watch window and change the ScanStateCmd attribute in the details pane, the ScanState attribute is updated immediately. Instead, if you place the ScanStateCmd attribute in a watch window, leave the ScanState attribute in the details pane, and then change the value of ScanStateCmd, the ScanState attribute is not updated until you select it in the details pane.

Set an object OnScan

The OnScan state indicates the object's run-time state, which means it is performing its normal processing as part of its execute method.

When setting AppEngines OnScan:

- The hosting WinPlatform's status must also be OnScan.
- Any object hosted on the AppEngine is automatically set OnScan.

To set an ApplicationObject OnScan

1. Select the object in the console tree.
2. In the details pane, right-click the **ScanStateCmd** attribute and click **Modify**. The **Modify Boolean Value** dialog box appears.
3. Assuming the current value is **False** (OffScan), click **True**, click **Apply**, and then click **OK**.

   Note that the **ScanState** attribute's value in the details pane remains False (OffScan).
4. To update the display, click the **ScanStateCmd** attribute.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Set an object OffScan

OffScan is the run-time state that indicates that an object is idle and not ready to run.

Because WinPlatforms host AppEngines, you must ensure that all AppEngines hosted by the WinPlatform are OffScan before setting the WinPlatform OffScan. An error message appears if you try to set a WinPlatform OffScan while its hosted AppEngines are still OnScan.

Although AppEngines host other objects, it is not necessary for all the objects to be OffScan before setting the AppEngine OffScan. When you set an AppEngine OffScan, all of its hosted objects are automatically set OffScan.

To set an ApplicationObject OffScan

1. Select the object in the console tree.
2. In the details pane, right-click the **ScanStateCmd** attribute and click **Modify**. The **Modify Boolean Value** dialog box appears.
3. Assuming the current value is **True** (OnScan), click the **False** option button, click **Apply** and then click **OK**.

   Note that the **ScanState** attribute's value in the details pane remains True (OnScan).
4. To update the attribute value to False, select it.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Manage

The topics in this section describe many aspects of managing your System Platform installation, from creating or renaming objects to maintaining overall system security and configuring the development environment.

Manage product licensing and user authorization

System Platform can run under different types of licensing and security. Originally, System Platform was sold with a perpetual license that allowed customers to use the product for an unlimited length of time.This release of System Platform introduces Operations Control mode, and under Operations Control, additional capabilities for managing user access called the connected experience.

Operations Control uses AVEVA Identity Manager to provide identity service for AVEVA products. The AVEVA Identity Manager helps administrators to implement authentication and allows access tokens to be reused across different AVEVA products.

Connected experience extends Operations Control to enable single sign-on of all your users, and allows centralized user management in CONNECT

The section that follows describes the behavior of System Platform when running under Operations Control and connected experience. If you do not use Operations Control and connected experience, System Platform will continue to function as it has in previous releases.

Connected experience and operations control

Connected experience requires an CONNECT account with a valid Operations Control subscription and user management. Connected experience allows single sign-on to the AVEVA products supported by Operations Control that you are entitled to use. Note that licenses must be installed and activated.

- Unified User Management: Selecting the Connected Experience option enables all Operations Control products on the node to require log in authentication with CONNECT. CONNECT-based authorization is the only security mode available under connected experience.
- Compatibility across nodes: Connected experience must be enabled on all nodes in your system. Applications previously build on nodes not enabled for connected experience must be reconfigured to function in the connected experience environment.

You can disable the connected experience at any time, but if you do, connected experience must be disabled on all nodes in your system. If connected experience is disabled, any applications that were built under connected experience must be reconfigured to function without connected experience, including both authentication methods and product licensing.

Configure operations control and connected experience

To operate Application Server and other System Platform products under AVEVA Operations Control, either as a product subscription only, or with connected experience, you must configure your installed and licensed

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

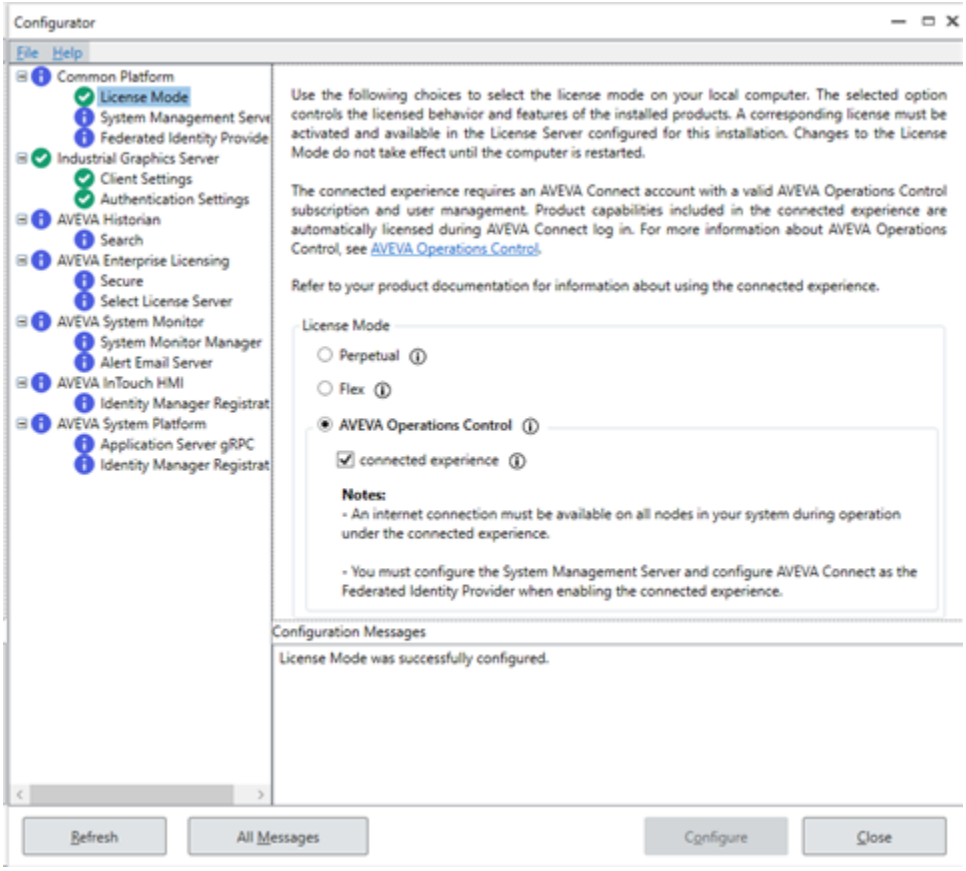
products.

For more information refer to:

- Operations Control Configurator
- AVEVA Identity Manager Help

To configure Operations Control connected experience mode:

1. Open the Configurator from **Start** > **AVEVA** > **Configurator**.
2. Select **License Mode** in the left pane to configure license mode.
   a. Select the **AVEVA Operations Control** radio button to enable a subscription license for one of the AVEVA Operations Control packages (Edge or Supervisory); that includes unlimited use of all products in the product package for your defined set of users.
   b. Select the **connected experience** check box to enable a Single Sign-on (SSO) experience across all Operations Control products on this node with CONNECT cloud capabilities, available in the products by default.

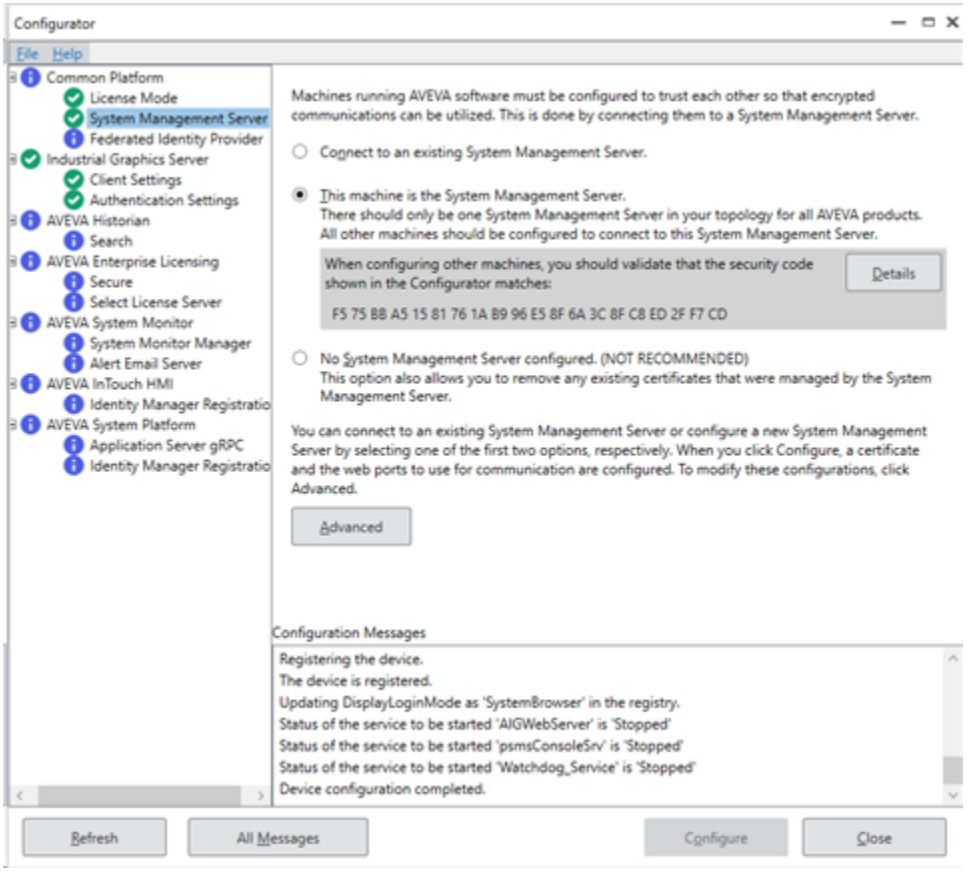


3. To configure the System Management Server, in the left pane, under **Common Platform**, select **System Management Server**.

Note: If you are prompted for user credentials for the System Management Server, use the following format to enter the user name: DomainName\UserName. The prompt for user credentials may be displayed if you have domain admin privileges but are not an admin on the local machine. You must be a member of the Administrators or aaAdministrators OS group to configure the System Management Server.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

- Connect to a remote server or configure the local machine as the SMS.
- No System Management Server option is **not recommended**. You must configure a System Management Server to use Operations Control and connected experience.
- Only one of the machines in the network is identified and configured as a System Management Server. Machines running AVEVA products can then connect to that single System Management Server to establish trust and configure encrypted communications.
- Configuring a Redundant SSO server does not configure a separate System Management Server. A Redundant SSO server is not required, but provides a backup in case the System Management Server loses access to the external authentication provider.

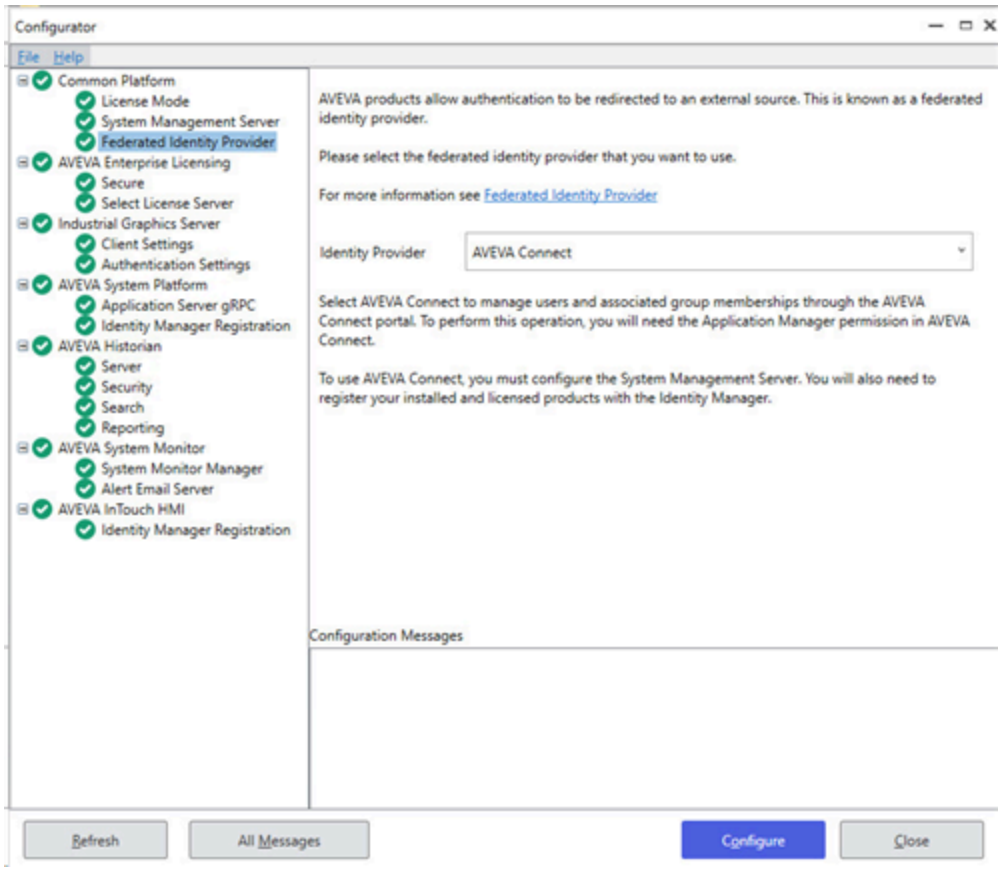


4. Select **Federated Identity Provider** under Common Platform in the left pane to configure a Federated Identity Provider.

   The Identity Manager component in the PCS Framework, available on the System Management Server (SMS) and Redundant SSO (RSSO) nodes, can be configured to enable "federated" login with CONNECT. This allows a user to enter the email address associated with their CONNECT account into the Identity Manager login form. Then, they are redirected to CONNECT to complete the log in process.

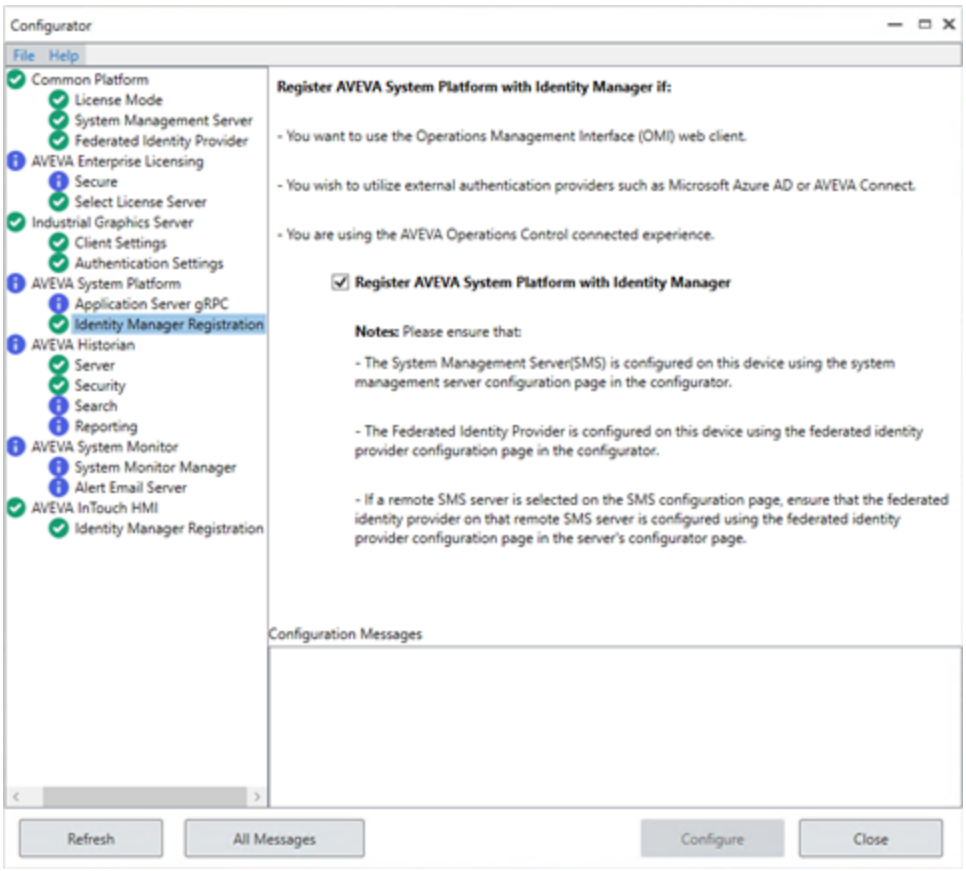
   Note: You must have a valid CONNECT account and be an administrator on that account to configure the Federated Identity Provider.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.



5. Under **AVEVA System Platform**, select **Identity Manager Registration**. Check that the required configuration requirements are complete. Then, Enable "Register AVEVA System Platform with Identity Manager" and click the **Configure** button. The configuration requirements that must be met before you can register with Identity Manager are:
   - The System Management Server (SMS) is configured.
   - The Federated Identity Provider is configured.
   - If you are connecting to a remote SMS, make sure that the Federated Identity Provider on the remote SMS is configured correctly.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.



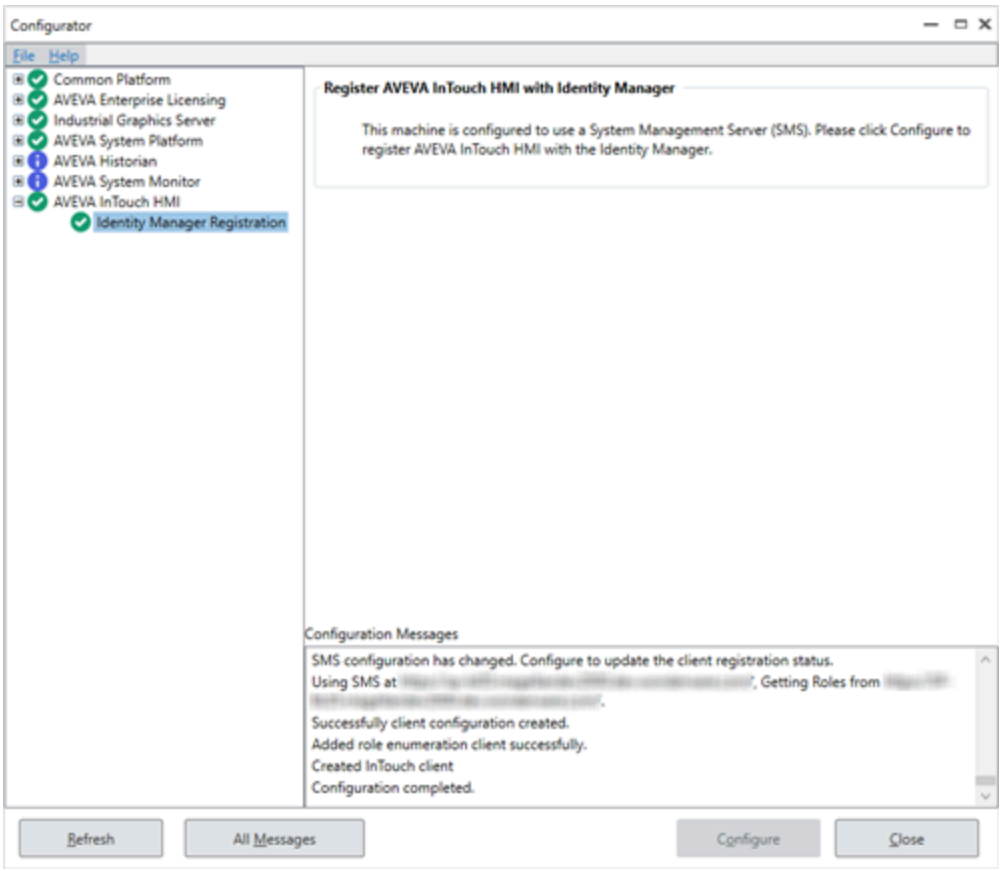
Other Components

1. To register InTouch with AVEVA Identity Manager, in the left pane, under **AVEVA InTouch HMI**, select **Identity Manager Registration**.

   Register InTouch HMI with Identity Manager only when you have completed the preceding steps:

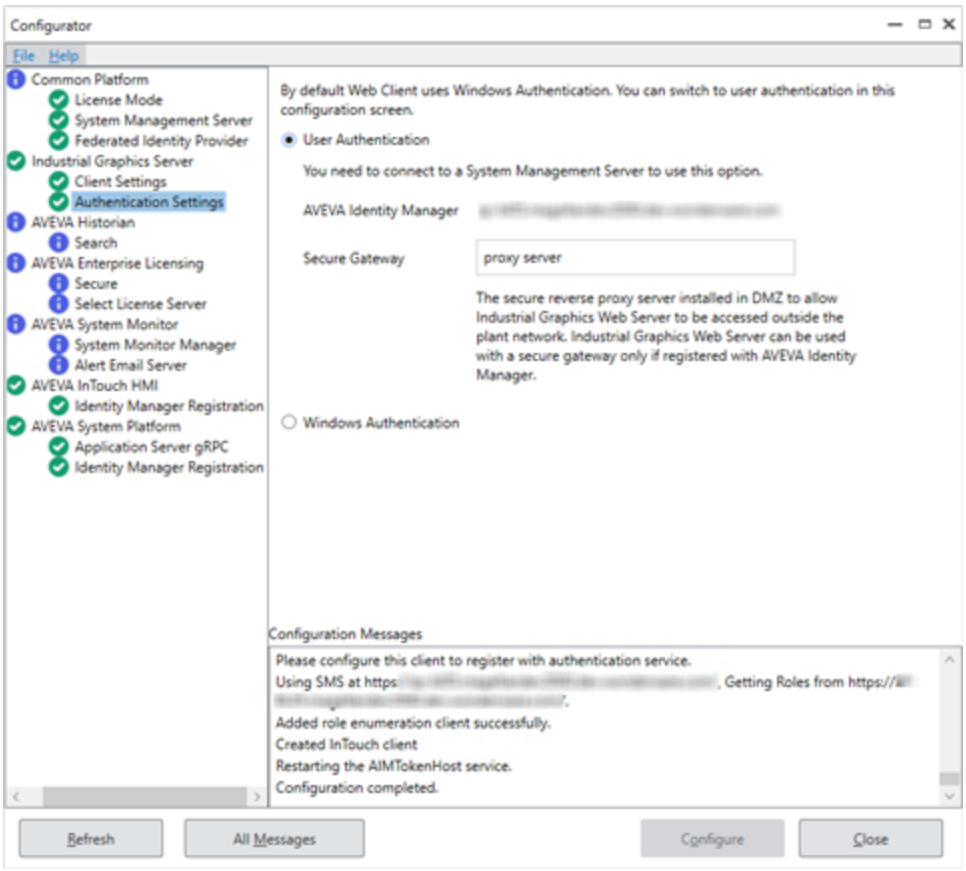
   - You have enabled AVEVA Operations Control connected experience as your license mode.
   - You have configured the System Management Server.
   - You have selected CONNECT in the Federated Identity Provider.
2. Click the **Configure** to register with Identity Manager.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.



3. To Configure the InTouch Web Client to use User Authentication, in the left pane, under **Industrial Graphics Server**, select **Authentication Settings**.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.



Authentication and entitlement

Authentication and entitlement, along with authorization, are security elements that serve to control user access to the AVEVA products that your organization uses.

- *Authentication* refers to verifying the identity of a user, through the use of a set of unique credentials (user name and password).
- *Entitlement* refers to the products that your organization has subscriptions for, and to which the authenticated user has been granted access. A user, even though they have been authenticated, may not be entitled to use a particular product, if the user does not have a valid Operations Control subscription.
- *Authorization* refers to the level of privileges associated with an authenticated user. This is generally done through the assignment of a user to a user group or through federated identity management.

  Federated identity is managed through a *Federated Identity Provider* (FIDP), and allows unified user management and single sign-on for users across the complete portfolio of AVEVA products. This is enabled through the Configurator, and is generally completed at the time of installation but can be reconfigured at any time.

The license mode you select plays an important role in how authentication, entitlement, and authorization are handled within your System Platform environment. See License mode for more information.

When Operations Control connected experience is enabled, each time a user logs in to a System Platform component, such as the IDE, the Historian, an OMI ViewApp, and so on, the log in is saved as an event in the CONNECT audit log.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

View the Data Log

Audit logs let authorized users from both your organization and AVEVA view the log of events or operations that occur in your account from the CONNECT portal. The logs display all operations that have taken place in your organization's account during the selected time period. You must be an Account Administrator, or must have the Report Viewer role to view the audit logs.

Note: Data in the audit logs is available from the time when the Audit functionality is available for your account.

To view the audit logs, go to the CONNECT web site. On the site navigation menu, select Audit. The Audit log page is displayed.

The audit logs provide the following information:

- Timestamp (date and time) when the operation was performed.
- Details of the operation, such as username, machine name, and the application name (for example, AVEVA OMI or System Platform IDE. The information that is logged includes which product or products were accessed by which authenticated user, as well as their available entitlement access.
- The audit log data can be filtered by using the **Entitlement Check** function in CONNECT. This function provides all data related to product usage through connected experience.

  Note: Local time is used while displaying the data on the user interface. However, the UTC time is shown in the operation details and when the data is exported.

See the CONNECT help for more information about audit logs.

Authenticate with the AVEVA Identity Manager

When Operations Control connected experience is enabled in the Configurator, you are prompted to authenticate with the AVEVA Identity Manager when when you start your first AVEVA application. Once you are authenticated, single sign-on (SSO) is enabled. With SSO enabled, you can run additional AVEVA applications without having to re-enter your credentials each time.

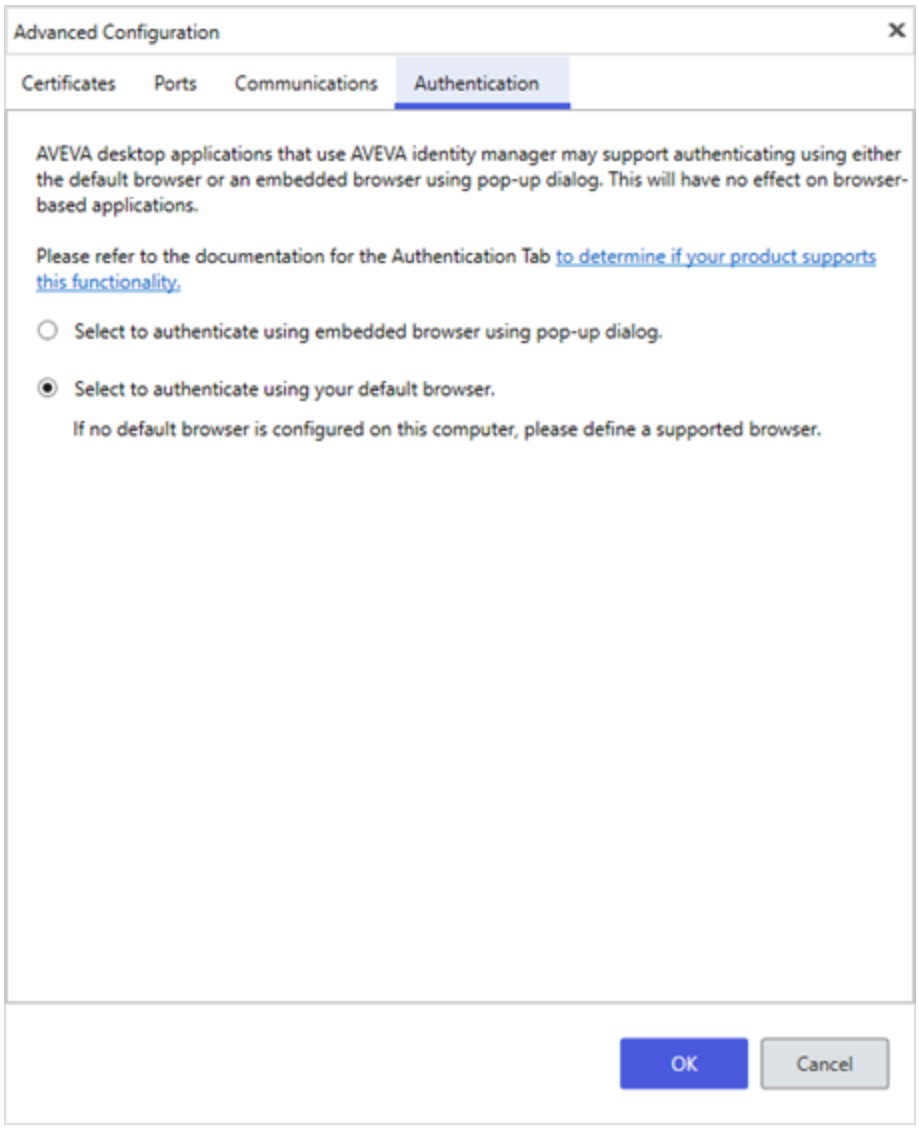
AVEVA Identity Manager allows two options for signing in, which are set in the Configurator, usually during System Platform installation. This is listed in the Configurator under **Common Platform** > **System Management Server** > **Advanced** > **Authentication**.

- To use your default browser, choose the **Select to authenticate using your default browser** option.

To authenticate using AVEVA Identity Manager on a web browser:

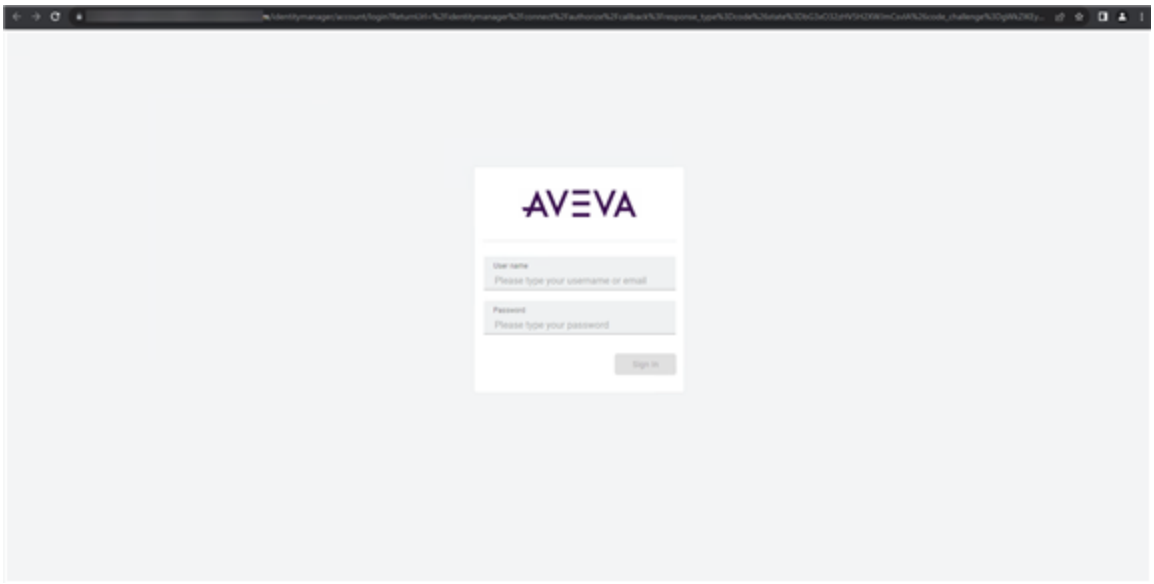
1. Set the option to use the default browser in the Configurator.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.



2. When you open an AVEVA Operation Control application, the AVEVA Identity Manager login page opens in the default browser set for your system.
3. In the AVEVA Identity Manager login page, your **User name** is your email address that is registered with an CONNECT account. Enter your email address to login to AVEVA Identity Manager.
4. Next, enter your password In the **Password** field and click **Sign In**.

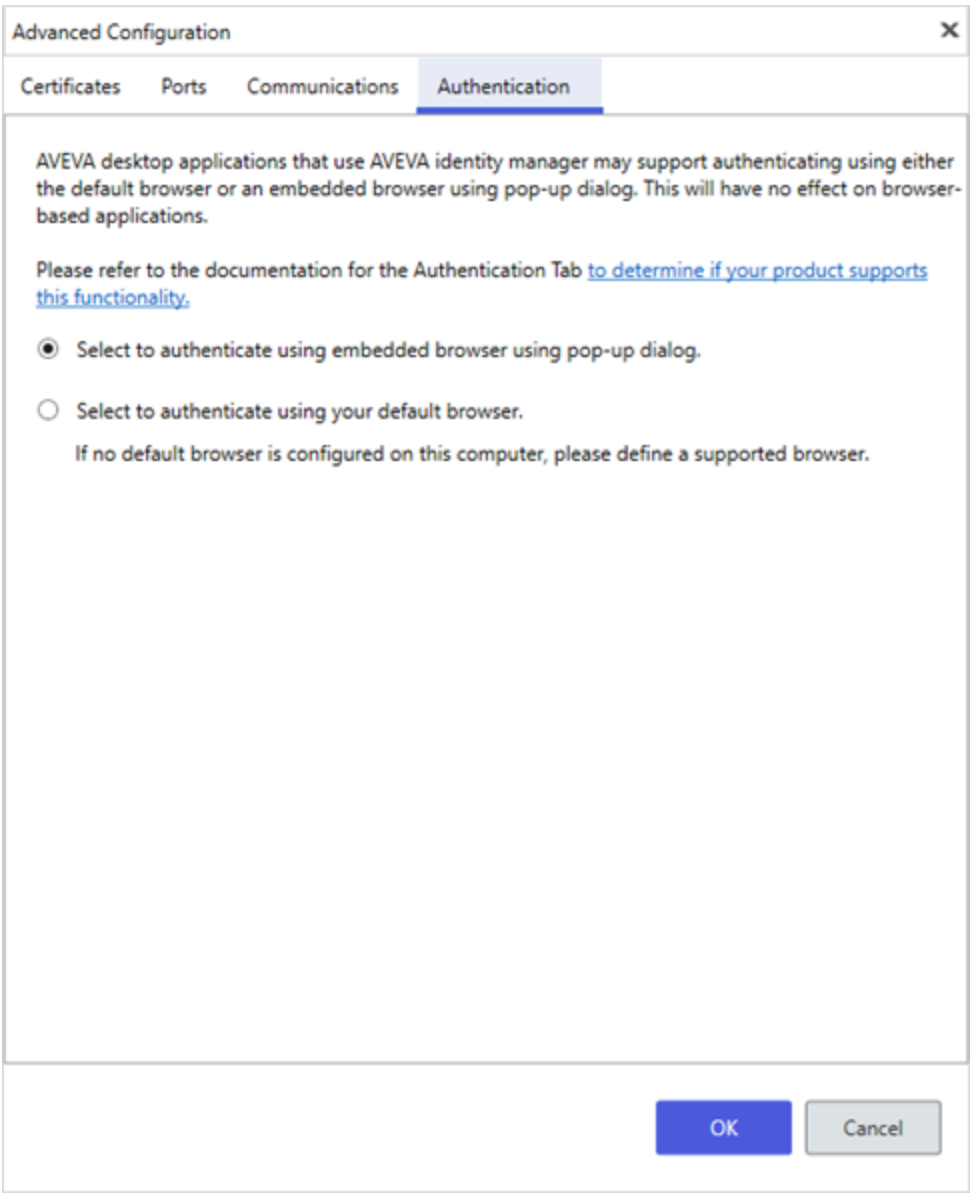
© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.



To authenticate using AVEVA Identity Manager on an embedded browser:

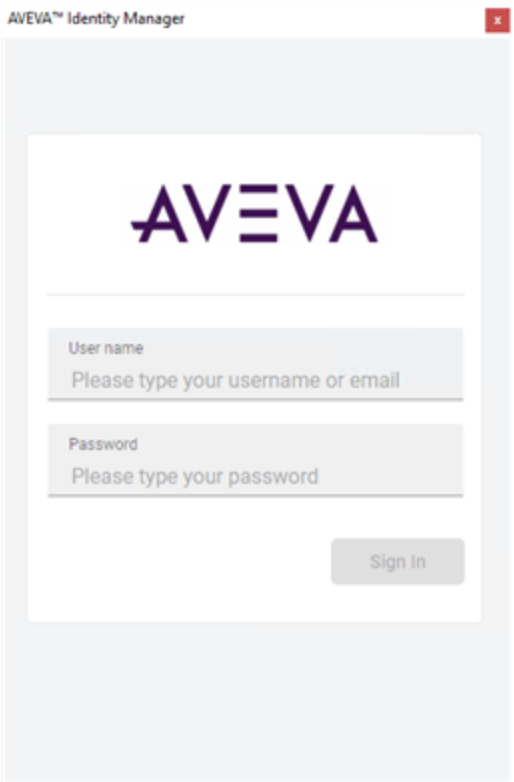
1. Set the option to use the embedded browser in the Configurator.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.



2. When you open an AVEVA Operation Control application, the AVEVA Identity Manager login page opens in the embedded browser.
3. In the AVEVA Identity Manager login page, your **User name** is your email address that is registered with an CONNECT account. Enter your email address to login to AVEVA Identity Manager.
4. Next, enter your password in the **Password** field and click **Sign In**.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.



Behavior if connection is lost

When you successfully sign in, your user profile which shows your user name and account number, appears in the title bar.

Two different types of connection loss may occur. These are:

- The connection between the application and the AVEVA Identity Manager.
- The connection between the AVEVA Identity Manager and CONNECT.

Lost connection between AVEVA Identity Manager and CONNECT

If, while your application is running, the connection between AVEVA Identity Manager and CONNECT is lost and prevents you from signing in again:

- On-premise capabilities will continue to function.
- Connected experience capabilities will stop working.
- If you try to start another connected experience application on the same node, it may not successfully run until the connection is restored. This applies to another instance of same product or a different Operations Control product.
- You have to sign in again using the Sign in option once the connection is restored.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

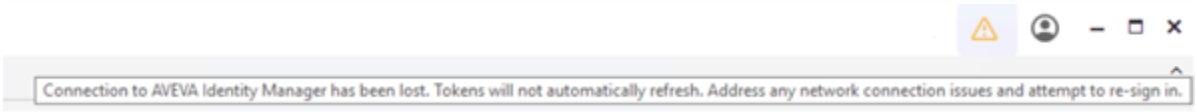
Lost connection between the application and AVEVA Identity Manager

If, while your application is running, the connection between the AVEVA Identity Manager and the IDE or an OMI ViewApp is lost:

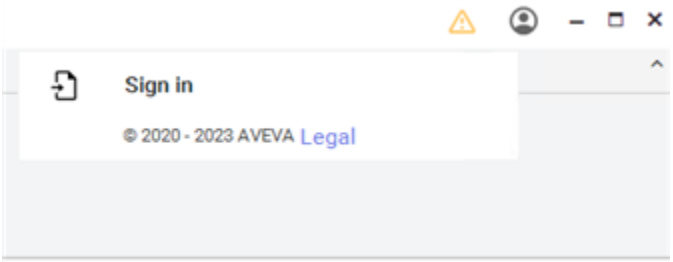
- A warning icon is displayed and provides a notification about the connection state.
- Any open OMI ViewApps will continue to run.
- If you try to open a ViewApp, it will run in read-only mode.
- You will not be able to perform any writes.
- The IDE will continue to function, but you will lose access to any shared Cloud folders.

To address connection loss issues

- If there is loss of connectivity to CONNECT or AVEVA Identity Manager, a warning icon is displayed and provides a notification about the connection state. When you hover the cursor over the warning icon, a tooltip with the connection loss message appears.



- Address any network connection issues. In the IDE, click on the profile icon and sign in again. For OMI ViewApps, you must sign out before you can sign in again.



- When the connection is restored, the warning icon will no longer be visible.

Single Sign-On (SSO)

The connected experience option enables Single Sign-on (SSO) experience across all Operations Control products with CONNECT cloud capabilities.

The connected experience requires an CONNECT account with a valid Operations Control subscription and user management.

Selecting the connected experience option enables all Operations Control products to require log in authentication with CONNECT when starting the first product. Products that are subsequently opened will authenticate using SSO. CONNECT-based authorization is the only security mode available under the connected experience.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Disable connected experience

You can disable the connected experience at any time, but the connected experience must be disabled on all nodes in your system. Galaxies and ViewApps built under the connected experience must be reconfigured to function in an environment that does not have connected experience capabilities, including both authentication methods and product licensing.

License mode

License Mode is set in the Configurator under the **Common Platform** tab. The Configurator lets you select from the following License Mode options:

- **Perpetual:** A specific AVEVA product license purchased for use in perpetuity.
- **AVEVA Operations Control:** A subscription license for at least one of two AVEVA Operations Control packages (Edge, Supervisory); includes unlimited use of all products in the product package for your defined set of users.
  - **Connected experience:** Enables Single Sign-on (SSO) experience across all Operations Control products with CONNECT cloud capabilities. See About connected experience configuration for more information.

Note that **Operations Control** mode is required in order to enable **connected experience**.

Start and run the IDE

The System Platform IDE in connected experience mode works according to the following steps. These steps assume that the IDE is the first Operations Control application that you are opening. Subsequent applications that you open will not require a separate sign-on (SSO).

1. Open the IDE.
2. You are prompted to authenticate with the AVEVA Identity Manager. For more information, see Authentication and entitlement.
3. Enter your username (email address) and password to sign in to AVEVA Identity Manager. The email address must be registered to an CONNECT account.
   - If the credentials are not recognized, an error message appears. If you made a mistake entering your username and/or password, re-enter your credentials. If you are unable to log in after correctly entering your credentials, contact your administrator for assistance.
   - When you sign successfully, and if you have a current (unexpired) Operations Control subscription available for your account, the IDE opens and your username is shown as the logged-in user.
   - If a valid subscription is not found, the IDE does not open.
4. IDE capabilities are enabled as per the permissions associated with the groups and roles assigned to you as the signed-in user.
5. To connect to a galaxy, see the next section.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Connect to a galaxy

Once you have started the IDE, you can connect to a galaxy. The process for connecting to a galaxy in connected experience mode is similar to the process for connecting to a galaxy where connected experience is not enabled. See Connect to a Galaxy for more information.

Galaxy migration to support connected experience

When you open an existing galaxy that was configured with the security mode set to Authentication providers (using Azure AD), you must set the security mode to “None” (when in non-connected mode), prior to opening the galaxy in connected experience mode. Connected experience mode is enabled and disabled under **License Mode**, in the System Platform Configurator. See License mode for details.

Failure to connect to the galaxy

However, the internal system checks are different when connected experience is enabled. You will not be able to connect to the galaxy if:

- Operations Control mode is not enabled on the GR node.
- The galaxy has a security mode that is not compatible with Operations Control connected experience mode. The only compatible security modes are **None** and **Authentication Provider**.
- The credentials you used to log into the IDE do not give you authorization to open the galaxy. In this case, you must restart the IDE and enter the correct credentials.

Successfully connect to the galaxy

The galaxy will open if the above system checks allow you to connect to an existing galaxy or create a new galaxy.

- Your user information will be shown in the IDE.
- CONNECT records that you are using the IDE.

For details about creating a galaxy, see Create a Galaxy.

Connect to a galaxy with no security configured

For compatibility with connected experience, you must set the security mode of the galaxy to either None or Authentication Providers. This applies to both new and existing galaxies.

This section describes what happens after you started the IDE and have connected to a new or existing galaxy. For more information, see:

- Start and run the IDE
- Connect to a galaxy

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Security mode configuration

1. Log into the IDE as described previously. See Start and run the IDE for details.
2. Create a new galaxy, or open a galaxy that does not have security configured.
3. Go to security configuration. See Configure security in the *Application Server User Guide* for details.
4. Select either **None** or **Authentication Providers**. If connected experience is enabled for System Platform, you are logged in once you start the IDE. You are not prompted to log in again when you connect to a galaxy.
   - If the galaxy security mode is **None**, users will still need to sign in to the IDE before they can connect to the galaxy. Sign in can be explicit or through SSO, if the user has already signed in to another connected experience application. Signing in to IDE is required, regardless of whether or security has been enabled for the galaxy.

     This completes security mode configuration for the galaxy; you can disregard steps 5 through 7.
   - If the galaxy security mode is **Authentication Providers**, select the **Roles** tab, and then click the **+** (add new group) button. The list of CONNECT user groups will be shown. Continue to step 5.
5. From the list of groups that are shown, associate IDE permissions to at least one of the groups. These permissions must include:
   - Can Start the IDE
   - User Configuration > Can Create/Modify/Delete Users
   - User Configuration > Can Modify Own User Information
6. Save the security information.
7. If prompted, log in again with your username and password.

Connect to a galaxy not using Operations Control connected experience (configuration mismatch)

This workflow describes how to connect to an existing galaxy that is not configured for Operations Control connected experience, when connected experience has been enabled for System Platform.

1. Open the IDE and select a galaxy.

   You are notified that the galaxy security mode is not compatible with Operations Control connected experience mode, and that you need to change the security mode to one that is compatible.
2. If you proceed, you are allowed only to change the galaxy security setting to **None** or **Authentication Providers**, as needed.
3. Configure the galaxy security as needed.
4. Save your changes. After the security changes are saved, the galaxy restarts.
5. If prompted, log in again with your username and password.

For additional details, see Galaxy migration to support connected experience.

Sign out of the IDE

The following information applies if you are signed into the IDE in a Operations Control connected experience

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

environment.

- Once you sign out of the IDE, the IDE closes.
- However, you remain signed into any other connected experience applications, such as OMI ViewApps, that you have signed into.
- You can sign back into the IDE with SSO, provided that you are still signed into other connected experience applications.
- If another user signs into the IDE and you are still signed into other connected experience applications on the same node, you remain as the active user on the other applications.

Start Application Manager

Using Operations Control mode and connected experience has no effect on starting and running Application Manager.

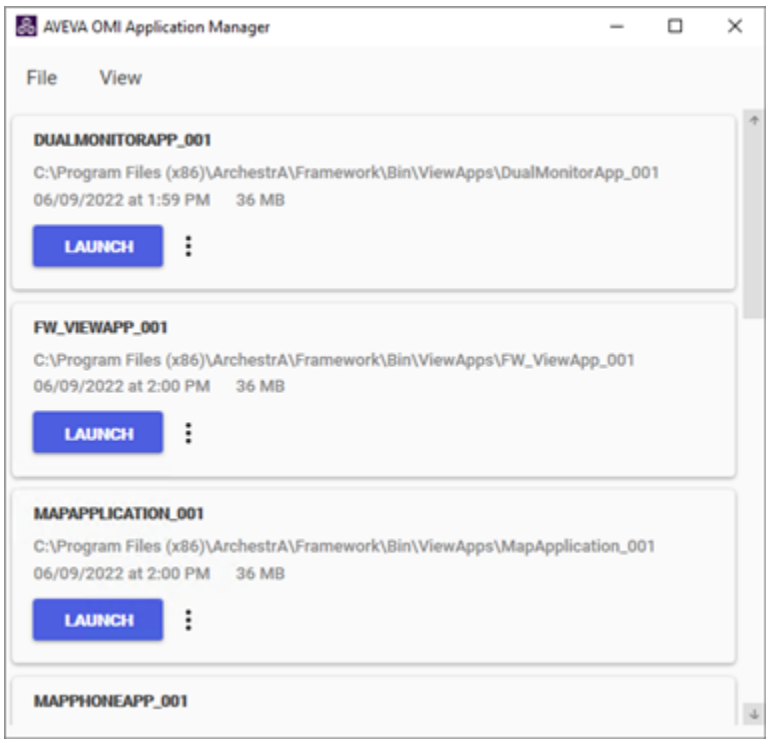
Launch a ViewApp using the connected experience

If your installation uses AVEVA Operations Control and the connected experience for user authorization and security, follow these steps to start an OMI ViewApp.

To launch a ViewApp using the connected experience

1. Show the desktop of the computer where the ViewApp is deployed.
2. Select the AVEVA OMI Application Manager desktop shortcut.

   AVEVA OMI Application Manager appears with an alphabetic list of ViewApps deployed on the computer.



   For more information about the fields of the AVEVA OMI Application Manager, see New Topic (59).

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

3. Select **LAUNCH** for the ViewApp you want to run.
    - **Already signed in (Single Sign On):** If you are already logged in to another ViewApp or other AVEVA product that uses Operation Control connected experience, you are logged in to the ViewApp automatically using Single Sign-on.
    - **Not yet signed in:** If you are not already logged in to a product that uses Operation Control connected experience, a browser window opens for you to enter your user name and password. The browser window is either the System Platform embedded browser or your default browser, depending on what which authentication option has been configured for the System Management Server.

After you successfully log in, the ViewApp opens. Your user name is shown near the right side of the titlebar in the ViewApp (but only if the ViewApp includes the OMI TitleBarApp).

If you have a problem logging in (authentication or entitlement failure)

- Make sure your internet connection is working.
- If user authentication fails, the ViewApp opens in read-only mode after you click OK. In read-only mode,
- If entitlement fails, you will see a message that a valid subscription could not be found. After you click OK, the ViewApp opens in read-only mode. In this mode, you cannot interact with the ViewApp (for example, to change a setpoint or to acknowledge an alarm); you can only monitor what the ViewApp is showing.

Corrective action

If you receive an authentication error message, repeat the log in procedure, being sure to enter your user name and password correctly. If you still cannot sign in, contact your CONNECT administrator.

If you receive an entitlement error message, there may be a problem with your AVEVA subscription. Contact your CONNECT administrator.

Implementing SSO in your ViewApp

OMI ViewApps leverage the Security.LoggedInUserSsoToken property. OMI Apps consume the user access token via the SDK using security attributes. These attributes are updated whenever the token is refreshed by a user signing in or out.

Configure user permissions for connected experience

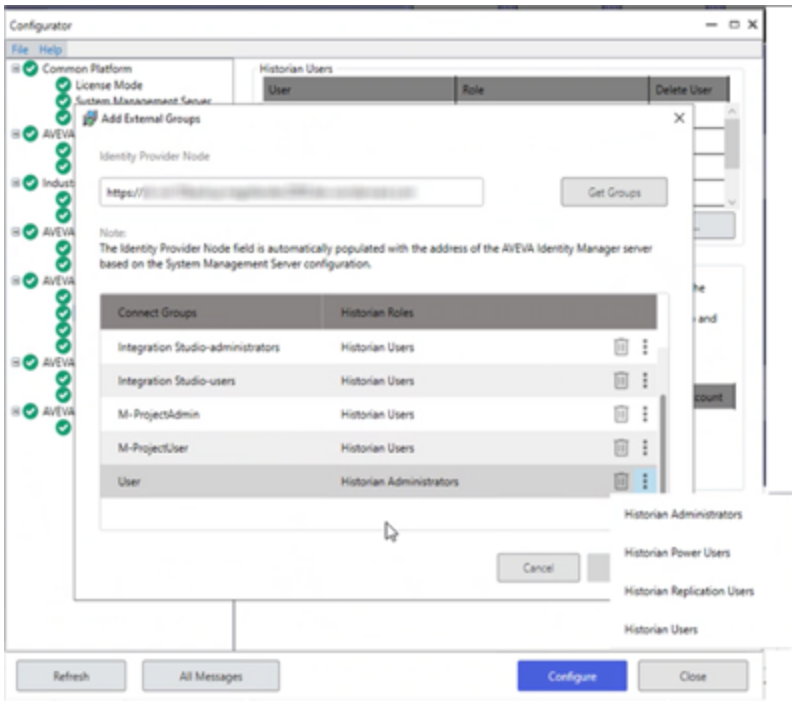
When connected experience is enabled, ViewApp users will need to be granted additional permissions by granting them either the "Historian Power User" or "Historian Administrator" role.

You can add users to these roles on a Historian node by using the System Platform Configurator.

1. Open the Configurator and select the **Historian Security** plugin.
2. Click the **Add External Groups** button. The **Add External Groups** window opens.

    Note: Add External Groups is only active if the System Management Server is configured with a federated identity provider, as is the case when .

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.



3. Click the **Get Groups** button in the **Add External Groups** window.
4. Select the group or groups that require additional permissions (roles).
5. Add the appropriate role or roles to the group(s).

Launch ViewApp in the OMI Web Client using connected experience

If your installation uses AVEVA Operations Control connected experience for user authorization and security, follow these steps to start an OMI ViewApp.

To launch a ViewApp in the OMI web client using the connected experience

1. In a supported web browser, go to the URL for the OMI web client:
   - On the machine where the WebViewEngine is deployed, the URL is `https://localhost/omi`. You can also use either of the two formats below.
   - On a machine on the same domain as the one where the WebViewEngine is deployed, the URL is `https://<node_name>/omi`, where `<node_name>` is the name of the platform where the WebViewEngine is deployed.
   - On a machine that is not on the same domain as the WebViewEngine, the URL is `https://<fqdn>/omi`, where `<fqdn>` is the fully-qualified domain name of the platform where the WebViewEngine is deployed.

   If you are already logged in to another AVEVA product that uses Operation Control connected experience, you are logged in to the OMI web client automatically using Single Sign-on. Skip to step 3.
2. If you are not already logged in to a product that uses the connected experience, a sign-in window opens for you to enter your user name and password.

   After you have entered your user name and password, the OMI web client page opens.
3. The OMI web client page lists all of the OMI ViewApps available through the web client. Select the ViewApp to run.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

When the ViewApp opens, the user name that you are logged in as shows in the top-right corner of the page.

If you have a problem logging in (authentication or entitlement failure)

- Make sure your internet connection is working.
- If user authentication fails, the ViewApp opens in read-only mode after you click OK.
- If entitlement fails, you will see a message that a valid subscription could not be found. After you click OK, the ViewApp opens in read-only mode.

Corrective action

If you receive an authentication error message, repeat the log in procedure, being sure to enter your user name and password correctly. If you still cannot sign in, contact your CONNECT administrator.

If you receive an entitlement error message, there may be a problem with your AVEVA subscription. Contact your CONNECT administrator.

Sign out from an OMI ViewApp

You can sign out from an OMI ViewApp by clicking the **Sign Out** option under the **Profile** icon. When you sign out of an ViewApp:

- Operations Control connected experience supports running multiple applications with with different users signed in at the same time.
- Sign out closes the user session from CONNECT.
- Inactivity time-out is not supported when connected experience is enabled. Users must actively log out of the OMI ViewApp to end their connection.
- When you sign out of a ViewApp, you are only signed out of that specific ViewApp. Other ViewApps and connected experience applications will continue to run with their current signed in user.
  - The ViewApp from which you signed out continues to run, but in read-only mode. No user information is shown in the title bar.
  - If another user signs into that ViewApp, the CONNECT login dialog is shown. Once the user is authenticated, the ViewApp is loaded again, based on the security mode of the associated galaxy. The information for the new user is shown in the title bar. The information for the new user is now used for SSO.
- When you sign out of a desktop application using the embedded browser, web applications will continue to run.
- When you sign out of a web application, all web applications will close.
- Closing an application does not do anything with the session. Only that instance of the application gets closed. The session is still on the CONNECT server. When you reopen that application, your session is available and you will be authenticated with Single Sign On.
- When you sign out of the IDE, the session is closed and the IDE exits.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Switch OMI ViewApp users

To change the user signed into an OMI ViewApp:

1. The current user signs out from the ViewApp.

   This action logs the user out of the ViewApp. However, the ViewApp reloads under the context of the logged-in Windows user, and is now running in read-only mode. The TitleBarApp in the ViewApp does not show any user information.
2. A new user now signs into the ViewApp. The user uses their AVEVA login credentials to sign in.

   The new user is authenticated, and a message is logged in CONNECT for this user.
3. The OMI ViewApp reloads, based on the security mode in the associated galaxy.

   The TitleBarApp in the ViewApp now shows the new user as the signed-in user.

Secured and verified writes

Operations that require you to log in every time you use them include secured writes, verified writes, and acknowledging alarm states. The behavior of these features does not change, whether or not you are using the connected experience.

To perform a secured write in Operations Control mode

1. Change a runtime value. The secured write dialog is displayed.
2. Enter your AVEVA username and password. The AVEVA Identity Manager authenticates you and the value change is allowed.

To perform a verified write in Operations Control mode

1. Change a runtime value. The verified write dialog is displayed.
2. Enter your AVEVA username and password. The AVEVA Identity Manager authenticates you.
3. The verifier clicks on the authentication button and enters their username and password.
4. Both identities are verified and the value change is allowed.

Deploy scenarios

Deployment in an Operations Control connected experience environment

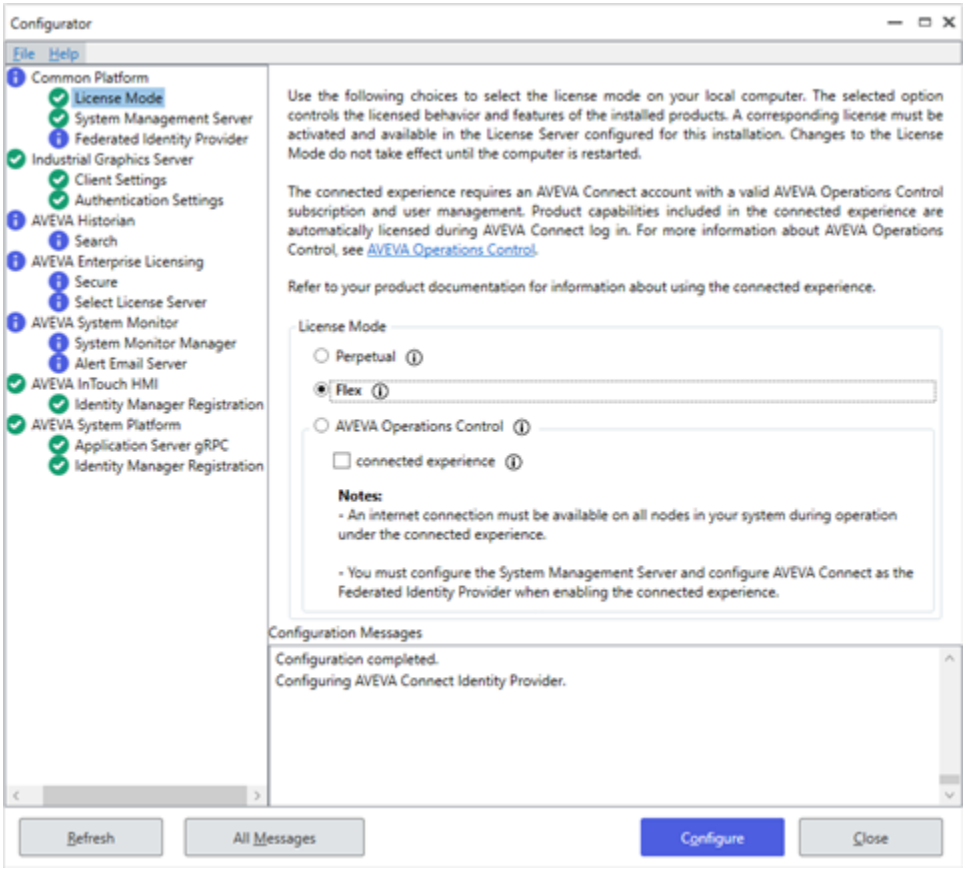
- As long as all nodes are configured for Operations Control connected experience, you can deploy a galaxy that has been enabled for Operations Control connected experience from the GR node to a remote Application Object Server node, provided that the license types match.
- If your GR node is not configured for Operations Control connected experience, but the remote node is, you cannot deploy objects to that node. The opposite case is also true: you cannot deploy from a GR node configured for Operations Control connected experience to a remote node that is not similarly configured.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Change Operations Control connected experience mode

If System Platform is using Operations Control connected experience mode, you can switch out of this mode if necessary. You will also need to change the security settings for any galaxies that use Operations Control connected experience. However, keep in mind that you will have to turn off Operations Control connected experience on all nodes, not just the GR or IDE node.

The license mode change is done through the Configurator. To change the license mode, you can simply uncheck the connected experience option, if you will still be using Operations Control mode. Or, you can change the license mode from AVEVA Operations Control to either **Perpetual** or **Flex**, depending on the type of license you have. Note that a system restart is required before any change to License Mode will take effect.



1. Now that you have changed the license mode, restart the GR.
2. Start the IDE. If the IDE is on a different node, you must change the license mode on this node as well. The IDE is no longer under Operations Control connected experience.
3. Connect to a galaxy. Note that the galaxy is still configured for Operations Control connected experience (using either **None** or the **Authentication providers** security setting).
4. Log into the galaxy as administrator.
5. Optional: Change the galaxy security model to the type of security you want to use.
6. Configure user credentials as needed to ensure that you and other legitimate users can still access the galaxy.
7. Save your changes and redeploy the galaxy.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Using the OCMC with connected experience

When using the Operations Control Management Console (OCMC) with AVEVA Operations Control and the connected experience, you are not required to log in when you start the OCMC itself. Instead, you are asked to log in with your user name and password the first time you open a node under Platform Manager in the tree pane of the OCMC.

If you have previously logged in to another Operations Control application, or to the OCMC, you will be authenticated using Single Sign-on and will not have to log in again.

Galaxy migration to support connected experience

If the galaxy security mode is Authentication providers (using Azure AD), the security mode has to be set to “None” (when in non-connected mode), prior to opening the galaxy in connected experience mode. Use the following procedure to migrate an galaxy to use connected experience if:

- You have a galaxy created or modified under System Platform 2023.
- The galaxy is configured with the **Authentication providers** security mode using Azure AD.
- You want to use Operations Control - connected experience.

This procedure is applicable to both fresh installs of System Platform 2023 R2 and upgrades from System Platform 2023 to System Platform 2023 R2.

To migrate an existing galaxy that uses Authentication providers security

1. Install System Platform 2023 R2 (fresh install) or upgrade your installed System Platform components to System Platform 2023 R2, as applicable.
2. Set the **License Mode** (under Common Platform) to **AVEVA Operations Control**.
3. Leave the **connected experience** option disabled.
4. Use the IDE **Import** function to migrate the existing galaxy to System Platform 2023 R2. The galaxy **Security mode** remains set to **Authentication providers**.
5. Connect to the galaxy.
6. Change the **Security mode** to **None**.
7. Open the Configurator and enable **connected experience**.
8. Connect to the galaxy.
9. Change the **Security mode** to **Authentication providers**.

   Note: If you enable connected experience initially (see step 3, above), you will not be able to connect to the galaxy. This is because connected experience uses CONNECT as the federated identity provider, while the galaxy is using Azure AD as the identity provider. In this scenario, when you attempt to connect to the galaxy, you will not be permitted to sign in.

Manage objects

After you create several objects, like templates, you need to manage them. For example, you need to check objects in and out, you need to validate objects, and you may need to rename objects. You can also export and

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

import objects, allowing you to reuse objects created in one Galaxy in another Galaxy.

Check out objects

To make changes to an object, you must check out the object. Then, you can modify the object and save private versions of it before checking the object in for other users to use. You can select more than one object for checking out at the same time.

The Galaxy marks the objects as checked out to you and it updates the object's **Change Log** that you can view in the **Properties** dialog box. A check mark is shown next to an object's icon in the IDE. No one else can check out the object until you check it back in or until you perform an Undo **Checkout operation**. However, others can open the object for read-only viewing.

To find objects that have been checked out, use the **Find** dialog box.

To check objects out

1. In the **Template folder** or **Application views**, select the objects you want to work with.
2. Do one of the following:
   - To check out the object without opening the Object Editor, on the **Home** ribbon, select **Check-in**, then select **Check Out**.
   - To check out and edit the object, open it in the Object Editor. An object is automatically checked out to you when you open its editor.

Check in objects

After you finish making your changes, you check the object back into the Galaxy. When you check the object back in, a dialog box prompts you to enter comments about changes you made.

You can turn this dialog box off if you do not want to enter information about your changes. For more information, see Customize your workspace.

Note: If the object was automatically checked out when the editor was started and you close the editor without making any changes to the object's configuration, an undo-checkout is automatically performed.

To check in an object to the Galaxy database

If the object is open in the Object Editor:

- Click the **Save and Close** icon at the top right of the Object Editor window.

If the object is not open in the Object Editor:

1. In an **Application** view or the **Template folder**, select the object after you are finished making your changes.
2. On the **Home** ribbon, select **Check-in,** then select **Check-in**. The **Check-in** dialog box appears.
3. Type any comments you want and click **Check-in**.

Validate objects

Objects need to be validated before they can be deployed. An object validates its configuration either when you

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

are configuring it, or typically when you save that configuration to the Galaxy database.

Validating an object's configuration includes:

- Checking allowable attribute value ranges.
- Compiling its scripts.
- Verifying its attribute references.
- Validating its extensions.
- Validating other configuration parameters that are unique to the object.

Important: Script validation on a template does not resolve references used in the script. For example, references to attributes that do not exist will not be discovered.

Typically, each option on the Object Editor that requires a string or numeric input has an allowable range of inputs. If you type an input outside the allowable range and then try to change the Object Editor page, close the Object Editor or save the object's configuration, a message appears about the input error, showing the allowable range.

To open the Validation area

- On the **View** ribbon, select **Operations**. The **Operations** pane, where validation information will be displayed, opens.

Validate scripts and other external components

Some objects depend on external components to run, such as script function libraries and references to other objects' attributes. The status of these external components can change, perhaps disabling some capability of the object.

For example, an object refers to a value of an attribute of another object, which is subsequently deleted. This will result in the remaining object going to a Warning status.

Normally, the system will update the validation status of an object when the missing script function or object/attribute is later added to the system. But there are a few cases where the status of an object needs to be manually validated by the user.

For example:

- When importing scripts and script libraries, there are cases when the script will import before the associated library and validate incorrectly, and
- When graphics associated with an object are imported along with a graphic they embed, the containing graphic may be imported first and validated incorrectly.

In each of these situations, the object may incorrectly have a status of either Bad or Warning. In this case, you may want to manually validate the object to update its status, especially if the status is preventing the object from being deployed. For more information, see Validate an object manually.

Two kinds of indicators are shown in the object icons:

- Deployment status for instances only.
- Configuration status for templates and instances.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Validate an object manually

After you check an object in, you can verify that an object's configuration is valid and update its status by manually validating it. You can use the **Template folder**, the **Application views** or the **Find** dialog box to find objects that need to be validated.

To validate all objects in the Galaxy, validate the Galaxy object.

Note: For a large Galaxy this is potentially a time consuming operation, and should be used only when necessary.

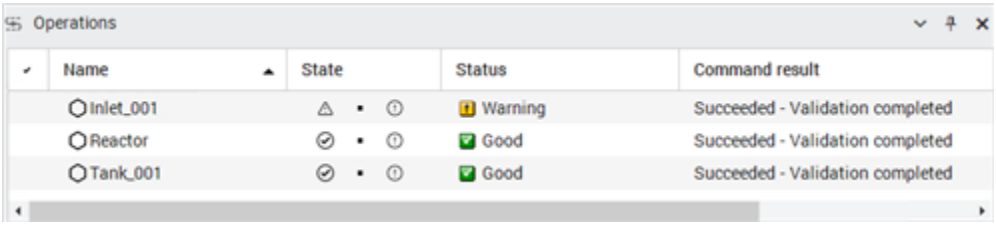
You can select more than one object for validation.

If an object is being edited, validation may not be performed. Also, if validation is in process on an object, other operations you start on the object will fail.

Note: You cannot cancel validation operations.

To manually validate one or more objects

1. In the **Template folder**, an **Application** view, or the **Find** dialog box, select the objects to validate.
2. On the **Home** ribbon, in the **Save** area, select **Validate**. The **Operations** view in the IDE opens and shows the status of the selected objects.



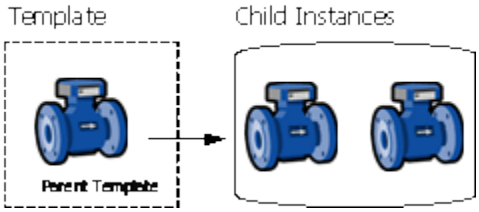
3. Continue using the IDE to perform other operations, if needed, while validation is going on, including work on other objects in the Galaxy. If you are validating a Galaxy, then you must leave the Galaxy alone until the validation process is complete.
4. When the validation process is complete, you see the results of the validation in the **Operations** view. If the validation failed on an object, you see a message. Correct the problem and validate again.

   Note: You can also see the errors or warnings that led to an object having a status that is not Good by looking at the Error/Warnings tab within the object's Properties.

Create an instance

After you create templates, you can create instances. Creating instances makes a specific object from a template, with all the characteristics and attributes of the template.

After you have created an instance of an object, it can be deployed.



You can also customize an instance, if needed. For example, you can have a valve template. When you create an instance of that valve, you can specify the inputs and outputs for that specific valve on your factory floor.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

To create an instance

1. Select the template you want to use for the instance. For example, to create a valve instance, select a valve template.
2. On the **Home** ribbon, in the **Create** area, select **Instance**. An instance is created.
3. Type the name for the instance. Instance names can be up to 32 alphanumeric characters. You cannot use $ as the first character. The name must include at least one letter. Instance names cannot include spaces.
4. To move the new instance in the **Deployment** view or **Model** view, drag the instance to the new location.

   You are ready to configure the instance, if needed. For more information, see Edit objects.

Object names

Object names must be unique within each namespace, not within the Galaxy.

- Template names can be up to 32 alphanumeric characters, including the required $ as the first character. The second character cannot be $ and the name must include at least one letter. You cannot use spaces in an object name.
- Instance names can be up to 32 alphanumeric characters. You cannot use $ as the first character. The name must include at least one letter. You cannot use spaces.
- The object name extension "_New" cannot be used if another object in the Galaxy uses the same name without the "_New" extension. For example, if you have an object named "Pump," you cannot have a second object named "Pump_New." You could, however, use "Pump_Old" and "Pump_New."

  Note that this restriction applies to templates as well as instances. For example, you cannot have a derived template named "$Area_New" or "$ViewApp_New" since these add the "_New" extension to a base template name.

  Note: You cannot use the following reserved names for objects: Me, MyContainer, MyArea, MyHost, MyPlatform, MyEngine and System.

An object can have three kinds of names if it is contained by another object. The three names include:

Name	Description
Tagname	The unique name of the individual object. For example, `Valve1`.
Contained name	The name of the object within the context of its container object. For example, the object whose Tagname is Valve1 may also be referred to as `Tank1.Outlet`, if `Tank1` contains it and it has the contained name "Outlet".

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Name	Description
Hierarchical Name	Hierarchical names are fully-qualified names of contained objects that include the name of the objects that contain it. Because the object that contains it may also be contained, there are potentially multiple hierarchical names that refer to the same object. For example, if: "Reactor1" contains Tank1 (also known within Reactor1 by its contained name "SurgeTank"). "Tank1" contains Valve1 (also known within Tank1 by its contained name "Outlet"). Valve1 could be referred to as: "Valve1" "Tank1.Outlet" "Reactor1.SurgeTank.Outlet".

Rename an Object

You can rename objects. Although you can change an object's containment relationship with another object, you cannot directly rename an object's hierarchical name. You can rename its tagname and contained name, and its hierarchical name will change automatically. See Rename contained objects for more information about renaming contained objects.

When you rename an object, references from other objects to the object being renamed can be broken. Objects deployed with broken references receive bad quality data during run time.

Note: Some objects may refer to themselves or to parent/host objects up in the parent/child hierarchy. References that go up or down the hierarchy to refer to other objects are called relative references. Objects with relative referencing are updated automatically if you rename them.

After renaming, all IDEs connected to the Galaxy show the new object name.

To rename an object's tagname

1. Select the object you want to rename.
2. On the **Home** ribbon, in the **Edit** area, select **Rename**.
3. Type the new name for the object.
4. When you are done, press **Enter**.

Delete an object

You can delete both templates and instances with the following exceptions. You cannot delete:

- Deployed instances

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

- Containers for other objects
- Objects checked out by other users
- Templates that have children (derived templates or instances)

Note: Make sure you correctly select the objects you want to delete. After you delete an object, you cannot undelete it. You must recreate it.

To delete an object from the Galaxy

1. In the **Template folder** or an **Application** view, select the object to delete. Select multiple objects by using **Shift+click** or **Ctrl+click**.
2. On the **Home** ribbon, in the **Edit** area, select **Delete**.
3. A message appears asking you to confirm that you want the object deleted. Select **Delete**.

Export objects

You can export some or all of your Galaxy objects. When you export, you are exporting the objects' associated templates, configuration states, and containment states. The information is saved in an .aaPKG file.

After the Galaxy objects are exported, you can import them into the same or another Galaxy.

If your objects have scripts associated with them, you need to export the script library separately. For more information about exporting script libraries, see Export script function libraries. For more information about scripts and script libraries, see the *Application Server Scripting Guide*.

If you are exporting an InTouchViewApp object and want to include graphics that are not embedded in the application, such as graphics called from a script, you need to embed the specific desired graphics in the InTouchViewApp.

Note: You can associate all Galaxy graphics with an InTouchViewApp for backup, restore, and deployment operations, but not for import and export operations.

Before you start, make sure all objects you want to export are checked in. If an object selected for export is checked out, the checked in version of that object is exported instead. This can lead to old versions of objects being exported.

Exporting an entire Galaxy is different than backing up the database. Unlike the case with backups, change logs for the objects are not exported. When you export objects, only the related security information for the specific object is exported.

To export an object

1. In the **Template folder** or an **Application** view, select one or more objects to export.
2. On the ribbon, select **Galaxy**, then select **Export.**
3. Select **Objects**, then **Selected objects**. The **Export Automation Object(s)** dialog box appears.

   To export all of the objects in the Galaxy,select **All objects** instead of **Selected objects**.
4. In the **Export** dialog box, browse to a path and type a name for the exported file.
5. Click **Save**. The file is saved with the specified name and an .aaPKG extension.
6. When the export is complete, select **Close**.

   Now you can import the .aaPKG file into another existing Galaxy.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Protect objects on export

Protect graphics and derived templates by flagging the objects as protected in the galaxy database. Protected graphic and template behavior is similar to that of a base template. This option is available through a specialized form of the **Export Object(s)** operation.

System Integrators and other system designers can use this functionality to protect objects designed in a master galaxy that are intended for use in production galaxies or galaxies on runtime nodes.

About protecting objects on export

Protecting an object on export does not change the object in the galaxy from which it was exported. Protection is effective only on import of protected objects. Specific behavior is described as follows:

Element or Function	Description
Base Templates	Are protected by default and cannot be checked out, edited or renamed.
Protected templates and graphics	Cannot be checked out, edited, or renamed, but can be deleted. A template derived from a protected template is unprotected. Protected templates and graphics are marked with a lock icon.
Ancestor objects	Are protected in the exported .aaPKG file when a child object is protected.
Protection effective	Protection is effective on import of protected objects. A protected object retains its protected status even when exported using the standard **Export Object(s)** workflow.
Instances	Cannot be protected. Only templates and graphics can be protected.
Select both template and instance for export	The option to export objects as protected is disabled. Only templates and graphics can be protected.
Export workflow	The export workflow is the same as for all objects except that you can select the option to export **As Protected Object(s)**.
Graphics	Graphics and client controls directly or indirectly embedded in a protected graphic are also protected. Protected graphics can be opened in the Graphic Editor as read-only.

Protected objects are marked in the **Template folder, Visualization folder**, or Application views (**Model**, **Deployment**, or **Derivation View**) with a shield icon to the right of the object name..

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.



Export objects as protected

The export objects workflow is the same as for all objects with the exception of specifying the export of selected objects as protected.

To protect objects on export

1. In the **Template folder** or an **Application** view, select one or more objects to export.
2. On the ribbon, select **Galaxy**, then select **Export.**
3. Select **Objects**, then **Selected as protected**. The **Export Automation Object(s)** dialog box appears.
4. In the **Export** dialog box, browse to a path and type a name for the exported file.
5. Click **Save**. The file is saved with the specified name and an .aaPKG extension.
6. When the export is complete, select **Close**.

   Now you can import the .aaPKG file into another existing Galaxy.

Export objects with I/O auto assignment

Objects configured for I/O auto assignment can be exported the same way as other objects. See Configure objects for additional information.

Application objects configured for I/O auto assignment are linked to DI objects and scan groups. Therefore, when you import these objects into a different Galaxy, the objects will look for DI object and scan group names that match the linkages that were made in their originating Galaxy. If a matching DI object and scan group are not found, all I/O auto assignment information is discarded, and the application objects are placed under the "Unassigned IO Device" folder in the **IO Devices** view.

- When exporting application objects, some I/O mapping assignments may be unavailable when importing the objects into a different Galaxy. Application objects that do not find a DI object and associated scan group that match their I/O references will lose their I/O mapping assignments and will have to be remapped.
- When exporting an application object with overrides in the I/O mapping table, overrides are preserved if a DI object and scan group that match its assignment from the original Galaxy are present in the new Galaxy.

  Note: The I/O mapping references are permanently deleted if matching DI objects and scan groups are not found. The I/O references will not reconstitute, even if you later add the scan groups and DI objects to which the application objects were formerly mapped.
- When exporting DI objects, there are no special considerations related to I/O assignments. I/O references for application objects will not change when the DI objects are imported into another Galaxy.

  Note: When importing a DI object into a different Galaxy, an existing DI object with the same name as the imported DI object may be overwritten. This can break device linkages to application objects previously configured within the Galaxy.

Export script function libraries

If you want to export objects that use scripts, the scripts are exported with the object.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Some scripts include functions that depend on external files called script function libraries. In this case, you must export the script function libraries separately.

To export a script function library

1. On the ribbon, select **Galaxy**, then select **Export.**
2. Select **Script**. The **Export script function library** dialog box appears.
3. If needed, browse to folder where you keep your script libraries. In the **Script function library** list, select the library or libraries you want to export.
4. Click **Export**. The selected script library is exported. Each script is named with the name of the script and an .aaSLIB file name extension.
5. When the export is complete, click **Close**. Now you can import the .aaSLIB file into another existing Galaxy.

Exporting Objects with Linked Content

When an object with linked content is exported, the linked content is exported along with the object. The exported object or objects can be a template, instance, or both. The linked content types that are exported include graphics, layouts, and external content items.

After import, relative references within linked graphics and layouts resolve to the current owning object.

Import objects

You can reuse objects from another Galaxy in your Galaxy. This saves you a lot of time if the objects are already set up in another Galaxy.

Importing instances previously exported from a Galaxy retains previous associations, when possible, such as assignment, containment, derivation, and area.

You can import objects from exported .aaPKG files or from an .aaPDF file. An .aaPDF file contains the configuration data and implementation code for one or more base templates. It is created by a developer using the Application Object Toolkit.

You cannot have two objects with the same name or more than one copy of the same version of an object in the same Galaxy. When you import an object, you can choose how you want naming and version conflicts handled.

You should perform an "Upload Runtime Changes" before importing a new version base template if instances of the template are deployed. This saves changes made at Runtime to the Galaxy database.

Objects that were created in Application Server 2014 R2 or later cannot be imported into a Galaxy running an older version. For example, you cannot import an object created in Application Server 2017 into a Galaxy running Application Server 2012.

Note: The Application Server version is not the same as the Codebase version. Objects with newer Codebases can be imported.

You cannot directly import script libraries, Client Controls and OMI Apps compiled with .NET 8 Framework unless the node is upgraded first. We recommend using NET 4.8.1.

If you import the .NET 8 WPF/Winform controls, currently no runtime can utilize them even when you import them.

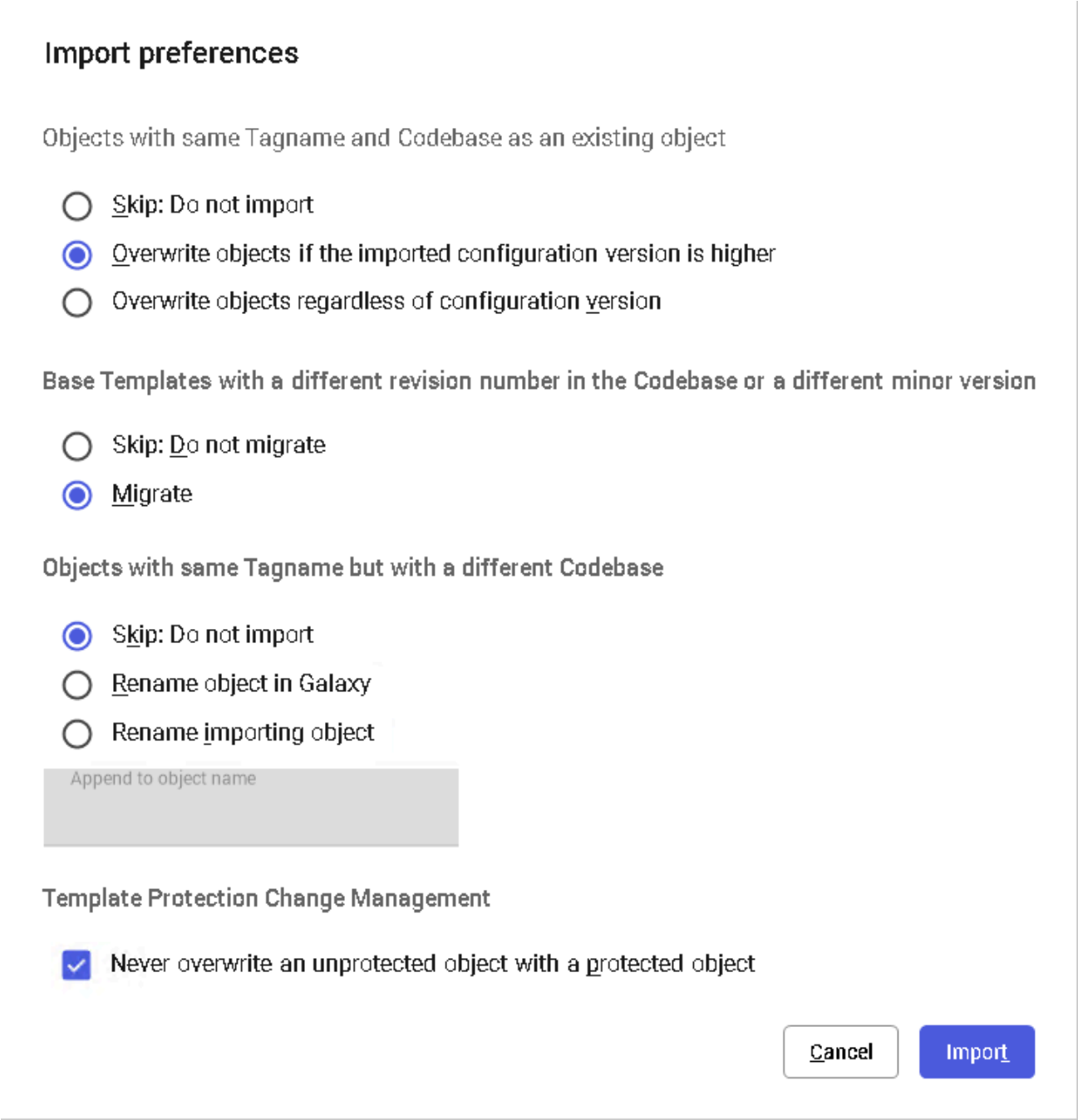
For the OMI web client, there is an internal workaround to import a .NET 8 Blazor component along with the original WPF 4.8 app, so the WPF will be used in the IDE/Layout Editor and deployed to the desktop, and the

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

.NET 8 Blazor app will be deployed to the web client.

To import objects

1. On the ribbon, select **Galaxy**, then select click **Import.**
2. Select **Objects**, then select **Objects from package**. The **Importing objects** dialog box appears.
3. Browse for the file with either an .aaPKG or an .aaPDF extension. You can select more than one file. Click **Open**. The **Import Preferences** dialog box appears.



4. In the **Objects with same Tagname and Codebase as an existing object** area**,** select one of the following:
   - **Skip: Do not import** leaves the existing object unchanged.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

- **Overwrite objects if the imported configuration version is higher** replaces the existing object with the object being imported if the imported object has a newer configuration. This is the default.
- **Overwrite objects regardless of configuration version** replaces the existing object regardless of whether the existing object has an older configuration or the same configuration.

5. In the **Base Templates with a different revision number in the Codebase or a different minor version** area, select one of the following:
   - **Skip: Do not migrate -** objects with an older codebase are not migrated when a newer codebase exists in the Galaxy.
   - **Migrate -** objects with an older codebase are migrated when a newer the replacement object.

     For more information about migrating, see After you import.
6. In the **Objects with same Tagname but with a different Codebase** area, select one of the following:
   - **Skip: Do not import** leaves the existing object unchanged.
   - **Rename object in Galaxy** imports an object with a matching tagname but a different codebase from the existing one. The existing object is not overwritten but is renamed.
   - **Rename importing object** imports an object with a matching tagname but a different codebase from the existing one. The existing object is not overwritten. The imported object is renamed.
7. In the **Template Protection Change Management** area, select the checkbox to prohibit overwriting an unprotected object with a protected object (default). Clear the checkbox to allow overwrites.
8. Click **Import**. The import process starts.
9. When the import process is complete, you can start using the objects you imported.

Import protected objects

The import objects procedure is the same for all objects, protected or unprotected. The following describes import preferences and results specific to importing protected objects:

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Import Preferences	Result for Protected Objects
Objects with same Tagname and Codebase as an existing object: • **Skip: Do not import** • **Overwrite objects if the imported configuration version is higher (default)** • **Overwrite objects regardless of configuration version**	Overwrite preferences are governed by the **Template Protection Change Management** option, "Never overwrite an unprotected object with a protected object" (default). • Overwrites can occur if both the packaged and galaxy objects have the same protection status. • A protected object can overwrite an unprotected object only if specifically permitted by clearing the "Never overwrite ..." option. • When overwriting a protected parent object, protection is not cascaded to child objects. Unprotected child objects remain unprotected.
Base Templates with newer or older Codebases (or minor version updates): • **Skip: Do not migrate** • **Migrate**	Import and migration of base templates is unchanged by protected object import functionality.
Objects with same Tagname but with a different Codebase: • **Skip: Do not import** • **Rename object in Galaxy** • **Rename importing object**	Object renaming on import can occur only if the object to be renamed is not protected. • If the object in the Galaxy is protected, and the importing object is not, and **Rename importing object** is selected, the importing object is renamed to have the suffix "_new". • If the object in the Galaxy is unprotected, and the importing object is protected, and **Rename object in Galaxy** is selected, and the **Never overwrite an unprotected object with a protected object option** is not selected, the object in the Galaxy is renamed with the suffix "_old".

Import objects with I/O auto assignment

Objects configured for I/O auto assignments can be imported the same way as other objects. See Import objects for additional information.

When importing objects that utilize I/O auto assignment, be aware of the following:

- If you are importing DI objects, there are no special considerations related to I/O auto assignment. I/O device mapping is not affected. An imported DI object will overwrite an existing DI object with the same name.
- If you are importing application objects, different outcomes can result, depending on the DI objects and scan

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

groups that already exist in the Galaxy.

- I/O device mapping is recreated, if possible, for each application object that is successfully imported and for which a DI object and scan group are found that match the existing I/O assignment. This includes all user-configured I/O attribute overrides.
- Objects for which a DI object and scan group with matching names cannot be found are moved to the unassigned area. All I/O information for those objects is discarded. This includes default I/O auto assignment information as well as any custom configured I/O attribute overrides. You will need to assign these objects to a DI object and scan group in the new Galaxy and recreate all user-configured overrides. None of this information is preserved for application objects that do not find a matching DI object and scan group.

Import objects with linked content

When you import an object with linked content, the object's links are restored and resolve to the current owning object. See Exporting Objects with Linked Content for more information.

Import a SQLData object

The SQLData object is used to store and retrieve data from a SQL Server database. Typical applications for this object include basic recipe management and database capture of attribute values.

The SQLData object can be configured to use one of the following authentication modes:

- Windows Integrated Security
- Windows Account
- SQL Server Authentication

When you import an object from another galaxy that is configured with SQL Server Authentication, the encrypted password does not follow the object, and the object will be in a warning state in the new galaxy. You must open the object in the Object Editor and enter a new password to remove the warning state.

Import script function libraries

You can enhance an object’s functionality by attaching a script to it. Some scripts include functions that depend on external files called script function libraries. Scripts are included in the object import operation, but you must import the script function libraries separately.

Note: You cannot directly import script libraries, Client Controls and OMI Apps compiled with .NET 8 Framework unless the node is upgraded first. If you attempt to import these elements, the system will say that the task was successful when in fact it was not. The solution is to upgrade the node. We recommend using NET 4.8.1.

If you import the .NET 8 WPF/Winform controls, currently no runtime can utilize them even when you import them.

For the OMI web client, there is an internal workaround to import a .NET 8 Blazor component along with the original WPF 4.8 app, so the WPF will be used in the IDE/Layout Editor and deployed to the desktop, and the .NET 8 Blazor app will be deployed to the web client.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

To import a script function library

1. From the ribbon, select **Galaxy**, then **Import**, then **Scripts**.
2. Browse to the directory that contains the script function library you want to import.
3. Select the script function library to be imported, and then select **Open** to start importing the selected library. Acceptable file types and file name extensions for script function libraries are:
   - Script Library Files (.aaSLIB file extension)
   - .NET or .COM files (.dll, .tlb, .olb, or .exe file extensions)
   - InTouch Script Extension Files (.wdf file extensions)

   An information window opens when the import successfully completes.
4. Validate any graphics that use the custom script functions contained in the imported library. See Validate objects for additional information.
5. Redeploy any InTouchViewApp instances containing graphics that use the script functions in the imported library.

   Note: When you import a new version of an existing script library into a Galaxy, you must stop and restart the engine that hosts the script's owning object. The new script version does not replace the old version until after the hosting engine has restarted and you have redeployed the object.

If you import an object whose script references a script function library that is not resident in the Galaxy, the imported object is set to Bad state and cannot be deployed. To correct this, import the script function library and validate the object. For more information about scripts, see the *Application Server Scripting Guide*. For more information about validating scripts, see Validate objects.

If you import a script function library that is a different version than the current library, a message is displayed to notify you that there are dependent objects. The message indicates how many objects will need to be redeployed if you continue with the import. If you continue with the import, the dependent objects are marked for redeployment.

Script function libraries that are COM libraries developed using Visual Studio 6 or earlier are not automatically deployed. To place this COM library on the target platform, you can either:

- Install it directly on the target platform and register it.
- Import it to the Galaxy, and then export it as an aaSLIB. Modify the aaSLIB xml to designate that the library is to be registered as a COM object. Reimport the aaSLIB so that it is automatically deployed and registered.

Import client controls

If you import a client control that is a different version than the current object, a message is displayed to notify you that there are dependent objects. The message indicates how many objects will need to be redeployed if you continue with the import. If you continue with the import, the dependent objects are marked for redeployment.

When the import process is complete, restart the IDE to apply the changes made by the import process. After the IDE has restarted, you can start using the objects you imported.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Import an app from the AVEVA digital exchange

A number of pre-built AVEVA OMI apps are available for download from the AVEVA Digital Exchange.

To access the Digital Exchange

1. From the ribbon, select **Galaxy**, then **Import.**
2. Select **Objects**, then **Explore product hub**.

   Your browser will open and connect to the AVEVA Digital Exchange.
3. Log in, or sign up for new account.
4. Click on the link for the **System Platform Developers Community** and browse for apps. Select the app you want and follow the online instructions.

Import AVEVA apps and client controls (ASUG)

You can import AVEVA apps developed with Windows Presentation Foundation (WPF), or Winforms/.NET controls through the IDE. Once you import the app or control, you can configure its properties and deploy it in runtime.

An AVEVA App is collection of one or more controls primarily developed with Windows Presentation Foundation (WPF) for use in AVEVA OMI ViewApps. Other technologies such as Windows Forms (WinForms) and HTML5, can also be used. You can create your own AVEVA Apps, which can be imported via WPF interoperability. For more information about creating your own AVEVA Apps, see the AVEVA OMI SDK Help. The AVEVA OMI SDK is installed automatically when you install the IDE.

You can enhance a Galaxy's functionality by adding an embedded control from an AVEVA App to a pane in a layout for inclusion in a ViewApp.

A control developed with Winform or .NET can be imported in much the same way, and like an AVEVA app, you can configure the control within a layout for inclusion in a ViewApp.

In order to qualify for import as an AVEVA App, the App should contain at least one WPF FrameworkElement Class control (System.Windows.FrameworkElement).

AVEVA Apps should be organized in a single folder which can optionally include subfolders. These subfolders typically contain locale files.

To import an AVEVA App

1. Assemble the App, together with all required dependencies and related files, into a folder.
2. From the ribbon, select **Galaxy**, then **Import.**
3. Select **Visualization**, then **OMI Apps**.
4. Select the folder that contains the App to be imported.

   The contained App, along with all dependent files, including dependent files contained in subfolders, are imported when you select **OK**.

   If the App does not import successfully, see Troubleshoot AVEVA or WPF apps that fail to import.

After the import process completes successfully, a new object appears in the Visualization folder named DisplayModule_00*n*, where *n* is 1 or the next available integer. The new object can be renamed, deleted, or exported as an .aaPKG file for use in another Galaxy. See Export objects for additional information.

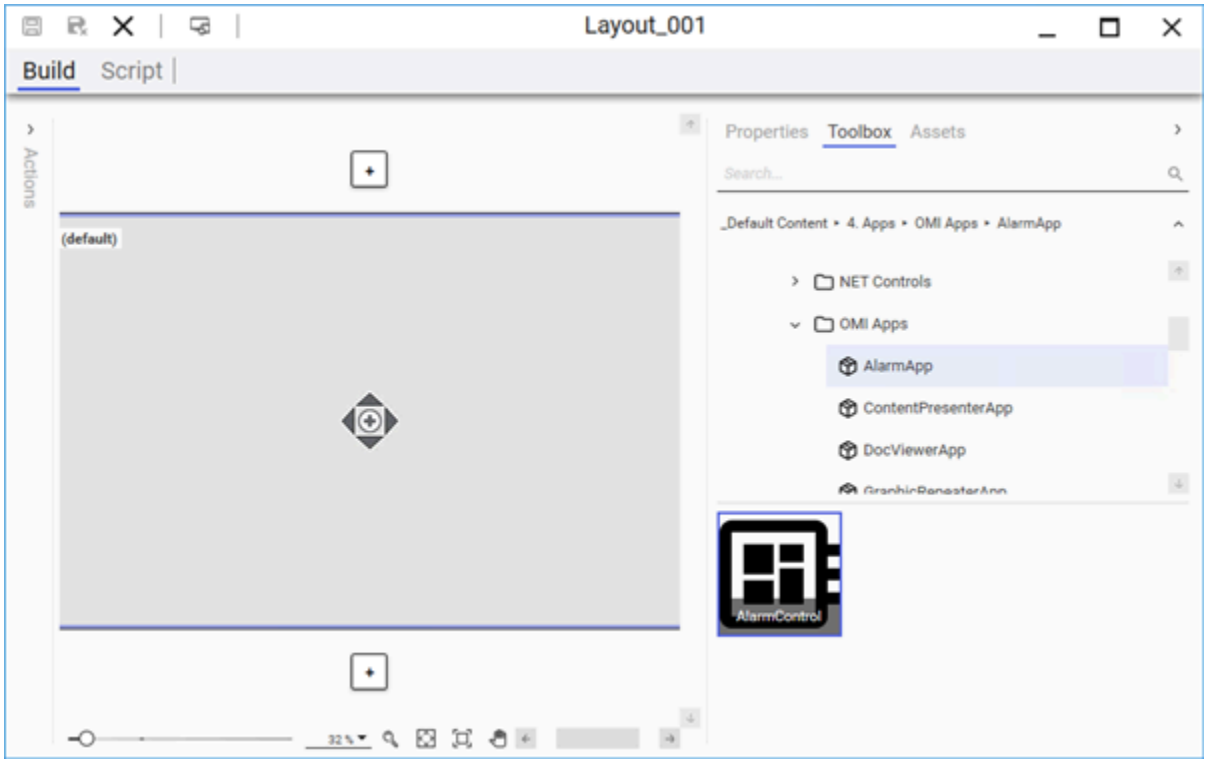
© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Add a control from an AVEVA app to a ViewApp

Controls from the imported App appear in the **Toolbox** tab of the **Layout** or **ViewApp** editors.

To assign a control from an App to a ViewApp

1. Open a layout in the **Layout** Editor and select the **Build** tab at the left of the editor.
2. Select the **Toolbox** tab on the right side of the editor and select a folder that contains the AVEVA Apps you want to add.
3. Select the AVEVA App to be added. The controls available in the App are shown.
4. Select a control and drag it onto a pane.
5. To view or override the control's public properties, select the **Properties** grid. Enter overrides as needed.
6. Save the layout. You can then use the layout that contains the embedded control in a ViewApp.



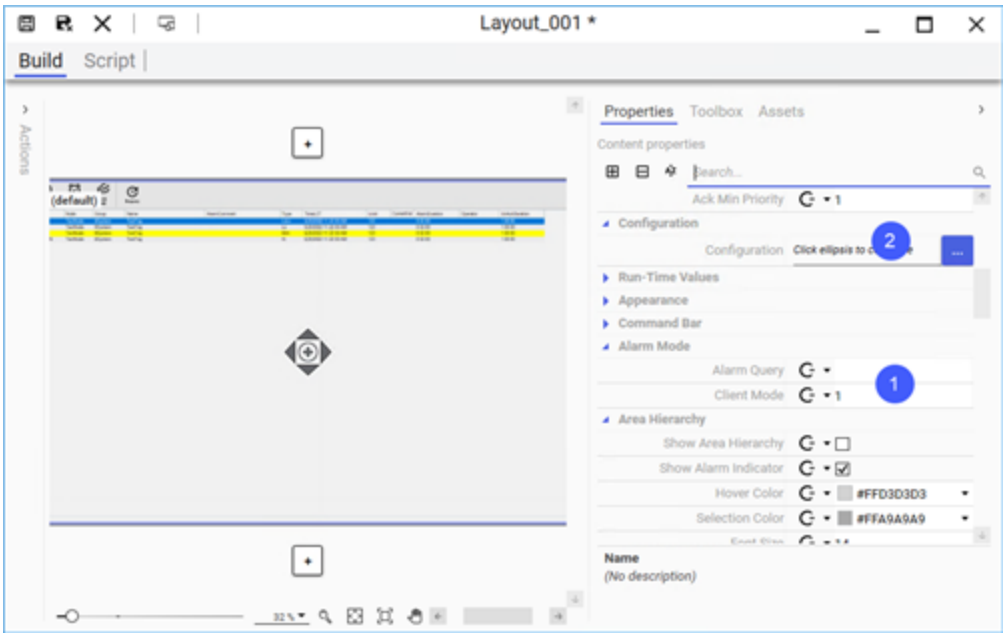
Configurable properties in AVEVA apps

AVEVA apps can contain two types of configurable properties:

1 — Dependency properties

2 — CLR properties (.NET framework properties)

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.



Configurable Property Data Types

- System.Boolean
- System.Byte
- System.Char
- System.Decimal
- System.Double
- System.Int16
- System.Int32
- System.Int64
- System.SByte
- System.Single
- System.String
- System.UInt16
- System.UInt32
- System.UInt64
- System.DateTime
- System.Drawing.Color
- System.Windows.Media.Color
- System.Windows.Media.Brush
- Any property that can be converted to or from string

CLR properties can only be set at configuration (design) time, while dependency properties can be bound to any attribute and can be set at runtime. If a dependency property is changed at runtime, the new value is propagated to the attribute. Dependency properties are data-bindable at design time, and you can set the binding direction

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

to in (read), out (write), or both. To change the binding direction, click on the arrow to toggle the binding direction.

For dependency properties, you must specify the attribute to which you are binding the property, as well as the direction of the binding (in (read only), out (write only) or both. The attribute naming convention is the same you use, for example, when configuring an animation in the Graphic Editor. Note that the ViewApp Editor does not provide syntax validation for attribute names, but the Logger will list any configuration errors.

Binding is only supported for the following basic data types.

DataBindable Properties

- System.Boolean
- System.Byte
- System.Char
- System.Decimal
- System.Double
- System.Int16
- System.Int32
- System.Int64
- System.SByte
- System.Single
- System.String
- System.UInt16
- System.UInt32
- System.UInt64

Any data type can be converted to and from the System.String type property.

AVEVA app development guidelines

The following types of properties are available for configuring WPF-based displays:

- Public writable CLR properties on the display that are not hidden using the browsable attribute
- Dependency Properties
- Nested Properties which satisfy the criteria of configurable properties
- Enum and Flag properties
- Any property that can be converted to or from string

The following types of properties are NOT available for configuring WPF-displays:

- Any property defined by the System.Windows.UIElement class
- Any property defined by the System.Windows.FrameworkElement class
    - The Tooltip property is an exception to this and can be configured

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

- The following properties defined by the System.Windows.Controls.Control class
  - IsTabStop
  - TabIndex
  - Template
- Any property defined by the System.Windows.Controls.ContentControl class

Optional Filter File: AppManifest.xml

You can include an optional file called **AppManifest.xml** in your imported AVEVA App. See the *AVEVA OMI SDK Help*, located in the **AVEVA Documentation** folder, for additional information about AppManifest.xml. This file will allow you to filter properties when you are configuring the app. The basic file structure is:

- DLL name
  - Control name
    - Property

      The following is a sample AppManifest.xml file, from the Hamburger App. The files specifies controls and properties that are exposed to System Platform IDE users as they configure a layout or ViewApp.

```
<?xml version="1.0"?>
<AppManifest AppVersion="1.0"?>
   <Filters>
      <!--List all the Controls that need to be exposed to the Systm Platform IDE
      user to utilize them in a View Application.
      Only the controls listed here will be available for the IDE users to place
      them on panes within a Layout or a ViewApp.-->
      <Control AssemblyName="AVEVA.Apps.HamburgerApp"
      ControlFullName="AVEVA.Apps.HamburgerApp.HamburgerButton">
         <!--List the public properties on this control that need to be exposed
         when user is configuring this control in a Layout or
         a ViewApp. The Property Editor within the Layout Editor and ViewApp
         Editor includes these properties when this control is
         selected and allows the user configure new default values.-->
         <Property Name="HamburgerColor" />
         <Property Name="SlideInPanePosition" />
         <Property Name="Background" />
      </Control>
   </Filters>
</AppManifest>
```

AVEVA app limitations and restrictions

- Events and scripting are not supported
- Only basic datatype properties configuration
- Apps cannot be imported from a network share location
- Collections and arrays are not supported
- Upgrading Apps is not supported
- Nested properties can be viewed and configured, but they cannot be configured with data binding
- To host the .NET controls, use the WPF element **WindowsFormHost** to wrap the control into WPF.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

```
<UserControl x:Class="AVEVA.Visualization.TrendApp.TrendControl"
      xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
      xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
      xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
      xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
      mc:Ignorable="d"
      d:DesignHeight="300" d:DesignWidth="300">
   <Grid>
      <WindowsFormsHost x:Name="FormsHost" />
   </Grid>
</UserControl>
```

Troubleshoot AVEVA or WPF apps that fail to import

There are two major factors that may stop an AVEVA or WPF app from successfully importing.

Expose controls and properties: Make sure that you have included an AppManifest.xml file with the app, and that all controls and properties that are needed in your app are explicitly exposed in it. For additional information about the AppManifest.xml file, see AVEVA app development guidelines and the *AVEVA OMI SDK Online Help*.

Block business logic from executing during design time: If you have business logic executing in the control constructor, or in static properties or static constructors, the import operation will not be successful. The control constructor is invoked when the control is instantiated. Therefore, do not include business logic within the control or static constructors, or in static properties.

- If you cannot remove business logic from a control or static constructor, or a static property, make sure the property is not invoked or that the property does not execute while in design mode.
- To block execution of logic within a constructor while in design time, add the following line to the constructor:

  ```
  !DesignerProperties.GetsInDesignMode
  ```

Import HTML5 widgets

Widgets are small web components that can extend the functionality of a web page or website. Custom-built websites can also incorporate widgets, by using open-source code or frameworks to provide certain functionality in whole or in part. A widget is a self-contained code block that slots into a website without changing any of its features. Widgets are most frequently used to provide on-screen user interface elements that ingrate with other platforms and data sources. A widget can be run on any web page on a website, with consistent placement and user interface, for example, social media, weather, RSS or podcast widgets.

By default, the following widgets are available under the Widgets folder in the Visualization folder:

- Carousel
- Web Browser
- QR Code Scanner

You can import a widget for a standalone and managed InTouch application, or for use in an AVEVA OMI ViewApp. The file format is Custom Widget Package (.cwp), which includes HTML5, CSS, and Javascript files.

Note: Widgets can also be imported and exported from a galaxy as objects. In this case, import/export the

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

widget as an object (aaPKG file type).

Import HTML Widgets

1. From the ribbon, select **Galaxy**, then **Import**.
2. Select **Visualization**, then **HTML5 Widget**.
3. Browse to the correct folder, if necessary. Select a .cwp file.
4. Click **Open**.

The widget will appear in the Visualization folder.

After importing the widgets

1. Open a graphic or layout.
   - For graphics, embed the widget.
   - For layouts, navigate to the widget and drag it to a pane.
2. Set the widget properties as needed. To open object help for a widget, select the widget in the Visualization folder and press **Ctrl + F1**.

Insert the widget on a window

The widget can now be viewed on WindowViewer and any web browser. Depending on the properties set in the design time you can manipulate the widget in runtime. Scripts using Custom Properties under 'Widget Properties' to modify widgets are not supported.

Carousel widget

A carousel widget allows you to cycle through elements—images or slides of text—like a carousel, without any input. This widget can be used to display dashboards, alerts or alarm information on large monitors on the plant floor.

Properties

In addition to the standard graphics properties, you can also configure properties specific to the widget, under Widget Properties.

Name	Description
Autoplay	If the Autoplay property is set to true, the carousel widget will automatically start on load. If it is set to false, the user must select the next item to start the carousel. The default value is True.
GraphicNames	A comma separated list of graphics the carousel will display in runtime.
Interval	The amount of time delay (in milliseconds) between automatically cycling an item.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Keyboard	If the Keyboard property is set to true, the carousel will respond to keyboard inputs. The default value is True.
Loop	If the Loop property is set to true, the carousel will cycle through the graphics continuously, else it will stop after a single cycle. The default value is True.
Pause	If the Pause property is set to true, the carousel will pause the cycling of the graphics, when it detects the mouse hovering or a touch down event. The graphics will resume cycling when the mouse is moved away. The default value is True.

The carousel widget is based on the Bootstrap 4.0 Carousel component, for more information on bootstrap, go here: https://getbootstrap.com/docs/4.0/components/carousel/

QR code scanner

The QRCode_Scanner widget connects to a camera to scan for a QR code and returns the resulting string.

Properties

Name	Description
QRCode	The resulting string of the scanned QR code. The default value is empty.

Limitations

- The device must have a camera.
- Using the QR Code on a physical machine instead of a virtual machine is recommended.
- Access the web client using the secure URL (https://) when using the web client remotely.

Usage

You can configure a script to read the QR code and display a graphic based on the scanned value.

In RunTime, the QR Code Scanner object will appear with two buttons - Start Camera;Stop Camera.

Click Start Camera and scan the QR Code. If the scanned code matches the script return value or graphic name, the graphic will be displayed.

Web browser widget

Using the web browser widget, users can display a web site in WindowViewer and the Web Client.

If Web Client is running in HTTPS, then only HTTPS URL page can be loaded. If Web Client is running in HTTP, then

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

HTTP and HTTPS can both be loaded. If the policy of the web site blocks cross domain access, then this widget will not work. The URL need not be in double quotes, but must be a valid URL.

Properties

Name	Description
URL	The address of the website.

Limitations

- If no protocol is specified, by default the https protocol will be used.
- If the Web Client is configured to use the HTTPS protocol, then only the HTTPS URL page will be loaded. If the HTTP URL is used, the web browser widget will display a message "`Mixed Content: The page at 'https://localhost/intouchweb' was loaded over HTTPS, but requested an insecure frame 'http://**********'. this request has been blocked: the content must be served over HTTPS.`"
- The web browser widget will not function, if the web site policy blocks cross domain (cross origin) access. A link will be provided to open the web page in a separate tab.

After you import

Imported templates are listed in the proper folder in the Template folder as defined in the object. Imported instances are shown in the Application views.

The following post-import rules apply:

- If a folder does not exist, it is created.
- If the object belongs to a security group that does not exist, it is associated with the Default security group.
- If the object belongs to an area that does not exist, it is associated with the Unassigned Area.
- If the host to which the object is assigned does not exist, it is assigned to the Unassigned Host.
- If you selected **Migrate** from the **Import Preferences** dialog box, the migrated objects are marked with "software upgrade required" if they are deployed. These objects will be upgraded when the objects are redeployed.
- If you import a new version of an existing instance, the new version is marked as requiring deployment if the existing object is already deployed.

The multi-galaxy environment

This section describes how to pair Galaxy Repositories to configure multi-galaxy communication. This lets you connect to and configure remote galaxies, and use standard run-time tools such as the Galaxy Browser, Object Viewer, and ViewApps to browse and subscribe to attributes across multiple galaxies deployed across multiple GR nodes.

Before you can enable multi-galaxy communication, your System Platform environment must be configured to

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

use a System Management Server (SMS). Only one SMS should be configured per System Platform environment, and all nodes within the environment should be connected to the single SMS node. Refer to System Management Server or see the *System Platform Installation Guide* for details.

Important! If you are using an IDE node that includes a local GR to configure galaxies on remote GR nodes, System Platform enhanced security will not let you connect to a remote GR unless you pair to the GR first. You must configure multi-galaxy communication to allow the connection. See Connect to a remote GR from the IDE for additional information.

About the multi-galaxy environment

Multi-Galaxy communication lets you configure a scalable, distributed automation system that spans multiple galaxies. It also allows you use the IDE to work with galaxies on remote Galaxy Repository (GR) nodes. You can configure connectivity for multiple GRs, browse attributes in multiple System Platform namespaces, and subscribe to attributes at run time across remote GRs.

A basic multi-galaxy environment consists of an IDE-only node (no GR), and two or more GR-only nodes. The GR nodes should not have the IDE installed.

As of System Platform 2017, communication from a GR with a local IDE to a remote GR is blocked by default. See Connect to a remote GR from the IDE to configure multi-galaxy communication between a GR with a local IDE and another GR node, with or without its own local IDE.

Set up multi-galaxy communication

Multi-Galaxy communication (MGC) lets multiple GR nodes to talk to each other at runtime. MGC is also required for configuring remote GR nodes. To configure multi-galaxy communication, you must configure servers that will perform galaxy service discovery, and then pair the galaxies that will be part of the multi-galaxy.

The functionality of tools such as Object Viewer, Galaxy Browser, ViewApps, and Managed InTouch applications is very similar in a multi-Galaxy environment as it is when working within a single galaxy. However, setting up the multi-galaxy environment requires some basic configuration.

The configuration workflow, using a two-galaxy pair as an example, is as follows:

1. Configure a three node system.
   - IDE node 0 (no GR)
   - GR node 1 (no IDE): contains Galaxy 1
   - GR node 2 (no IDE): contains Galaxy 2
2. Prepare the multi-galaxy environment by adjusting galaxy names as necessary. Each galaxy in a multi-galaxy environment must have a unique name.
3. From the IDE, connect to Galaxy1 on Node1.
4. To configure the service discovery servers for Galaxy 1, select **Configure** from the **Galaxy** menu, then select **Communication** and open the **Service discovery servers** dialog.

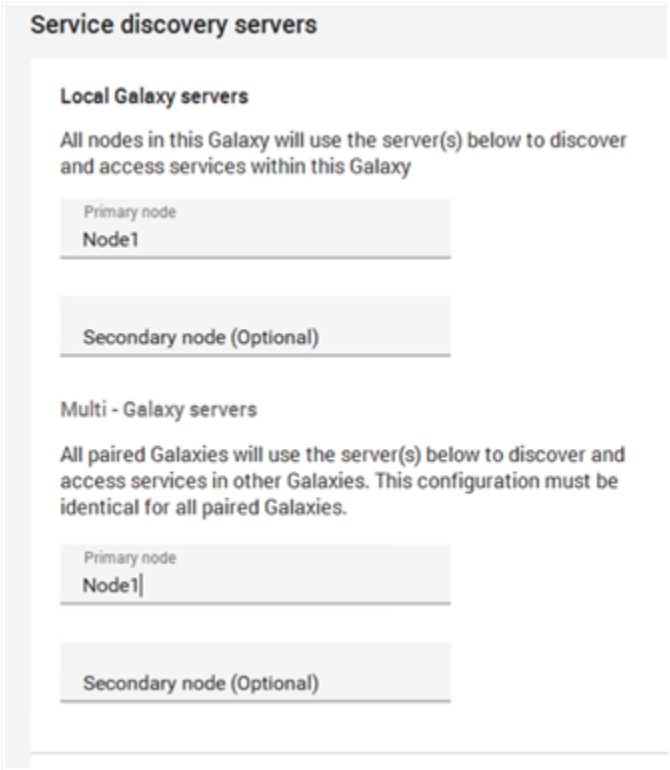
   Set **Node 1** as the **Primary node** for both **Local Galaxy servers** and the **Multi-Galaxy servers**. See Defining Service Discovery for additional information about the Local and Multi-Galaxy servers. You can configure secondary nodes for redundancy if needed.

   Important! There can be only one Cross Galaxy Server in a multi-galaxy environment. All GRs in the

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

environment must point to the same Cross Galaxy Server node. You can specify nodes by name or by IP address. However, we recommended that you use the hostname of the node for galaxy pairing.

If you specify the node by IP address instead of using the node name, make sure the reverse DNS lookup zone is configured properly on the DNS server. If the reverse DNS lookup zone is not set up, galaxy pairing with the node IP address may be successful. However an error message similar to "Failed to get the Discovery Service information for the target Galaxy Repository 'x.x.x.x'" may be generated. You can ignore the error message.



5. To configure multi-galaxy communication, select **Configure** from the **Galaxy** menu, then select **Communication** and open the **Multi-galaxy configuration** dialog.
   a. From the IDE, click the **[+]** button and add **Galaxy 1** (on GR **Node 1**) as a paired galaxy.
   b. Select the checkbox to enable Remote Pairing. Then, enter a passphrase. The passphrase must be the same for each galaxy to be paired, and must meet a minimum complexity standard:
      a. Minimum length is nine characters, including at least eight alphanumeric characters (with at least one upper case and one lower case letter and at least one number)
      b. At least one symbol: ! @ # $ % ^ & * ( )

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.



c. Click OK. A message that pairing has been enabled is displayed.

d. Now, add **Galaxy 2** (on GR **Node 2**) the same way that you added Galaxy 1. The Remote Galaxy Repository Pairing dialog appears.



e. Enter **Node 1** as the Target Galaxy Repository, and enter the same Passphrase as you entered for Galaxy 1.

Node 1 is now listed as a paired Galaxy Repository in the **Multi-Galaxy Configuration** dialog.

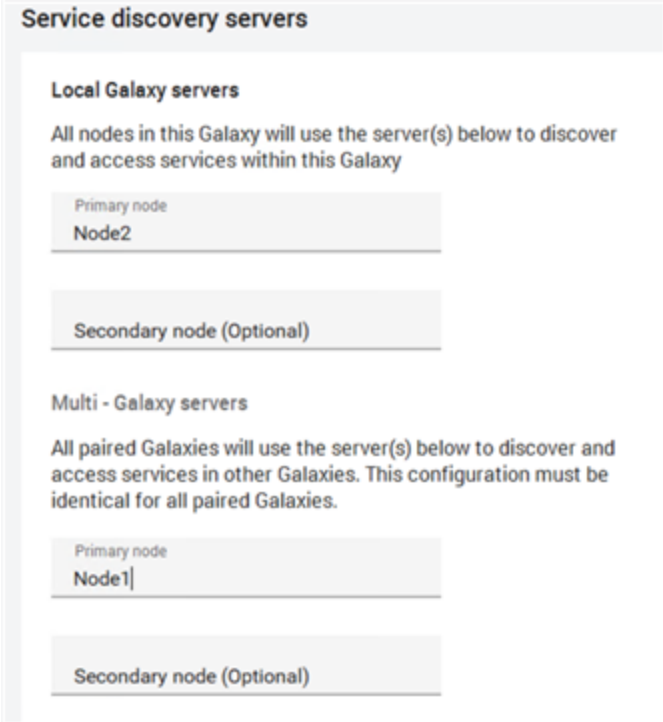
To change the Cross Galaxy Server

Use this procedure to change the Cross Galaxy Server for a previously paired Galaxy:

1. From the IDE, connect to the Galaxy on a remote node.
2. Configure Service Discovery Servers for the Galaxy.

   Set the node as the **Local Galaxy Server.** Set the new node as the **Multi-Galaxy Server**. You can configure secondary nodes for redundancy if needed.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.



When pairing, the GR node initiating the pairing gets a copy of the primary Multi-Galaxy server node if none has been configured.

Galaxy names in a multi-galaxy environment

Setting up a multi-galaxy environment requires a unique name for each galaxy in the environment. This may require you to rename existing galaxies if you plan to include galaxies in your multi-galaxy environment that currently have the same name.

We recommend performing any necessary renaming operations prior to implementing the multi-galaxy environment.

For more information about creating and backing up galaxies, see Getting Started with the IDE and Manage galaxies.

For more information about renaming galaxies, see Rename a galaxy.

Important: You must create a new galaxy with a new, unique name, from a backup .cab file rather than by creating a galaxy and performing a restore of the backup .cab file.

Unpair galaxies

You can modify your multi-galaxy configuration by unpairing individual galaxies.

To unpair galaxies

1. From the ribbon, select **Galaxy**, then select **Configure.**
2. Select **Galaxy**, then select **Multi-galaxy configuration**.

   The **Multi-galaxy configuration** dialog appears.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

3. In the **Paired Galaxies** area, select the Galaxy Repository you want to unpair, then click the **Delete (-)** button.

Under certain conditions, the unpairing operation can fail. For more information, see Troubleshoot a remote galaxy connection.

Rename a galaxy

To rename a galaxy for use in a multi-galaxy environment

1. Select a galaxy you want to rename, undeploy it and back it up to create a .cab file.
2. Use the .cab file as a "template" by placing it in the BackupGalaxies folder:
   \Program Files (x86)\ArchestrA\Framework\Bin\ BackupGalaxies
3. Create a new galaxy based on the backup .cab file, with a new name. The name must be unique, not in use anywhere else in the multi-galaxy environment.
4. Repeat the preceding steps for each galaxy to be renamed with a unique name.
5. Redeploy each newly created galaxy.
6. Delete the original galaxy from the Galaxy Repository node.

Your galaxies can now be configured for use in a multi-galaxy environment.

Rename paired nodes

Renaming a node that is paired with other nodes will disable communication with all other nodes to which the renamed node has been paired.

If a paired node is renamed, particularly if it is the designated Cross Galaxy Server, you must unpair all paired nodes and re-pair them using the new node name.

If you need to rename a node that is already paired, you should unpair the node first, then re-pair it after renaming it.

For information about naming Galaxies for use in a multi-galaxy environment, see Galaxy names in a multi-galaxy environment.

Access multiple galaxies

You can browse from one galaxy to another in a multi-galaxy environment through several methods.

- Use the Galaxy Browser, as described in Use the galaxy browser to reference objects .
- Use the Object Viewer, as described in the *Object Viewer User Guide*.
- Browse galaxies while configuring and running Managed InTouch applications, as described in the *InTouch HMI Application Management and Extension Guide*, the *InTouch HMI Data Management Guide*, and other InTouch HMI user guides and online help.
- Browse galaxies while configuring and running AVEVA OMI View Applications, as described in the *AVEVA OMI Online Help*.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Use the galaxy browser with multiple galaxies

Open the Galaxy Browser as usual to display the objects and attributes in your galaxy. For illustration purposes, this galaxy's name is Factory2 and you have paired it with Factory1.

The names of all paired galaxies will appear in the **Namespace** selection box. Select Factory2 to view the Tagnames and Attributes in Factory2.

Using the same instance of the Galaxy Browser, you can select Factory1 from the **Namespace** selection box. The Factory1 Tagnames and Attributes will appear.

You can work with the Factory1 Tagnames and Attributes in the same way as with those of Factory2.

For example, you are working in the Object Editor scripts tab, and you want to add a reference from Factory1 in a script. You can open the Galaxy Browser, select Factory1 in the **Galaxy Selection** box, select the Tagname "Outlet_001", select the Attribute "udInt3", then click OK. The Attribute reference is inserted into the script as

```
Factory1:Outlet_001.udInt3
```

at your cursor location. This functionality is unchanged except for the addition of the GR node prefix to the Tagname and Attribute.

Use Object Viewer with multiple galaxies

Open Object Viewer as usual to view data types, quality, values, timestamps, and status of the Application Objects, to perform diagnostic testing, and to modify selected attributes.

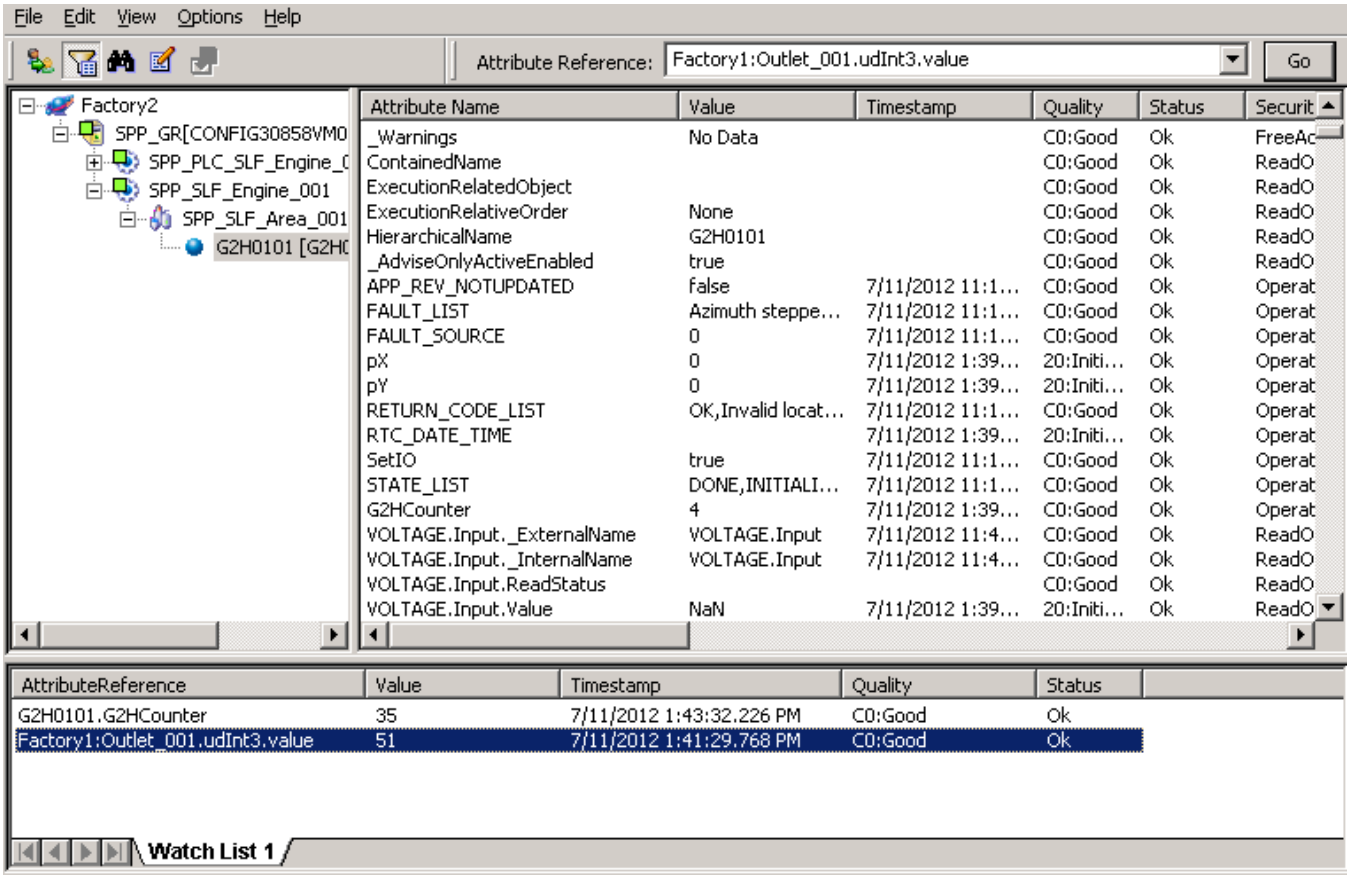
Object Viewer will display the console tree hierarchy of the GR node on which you have opened it. In this example, open Object Viewer from the Factory2 GR node. The attribute names, values, timestamps, and other data will display for a selected object. Add and manage attribute references to the Watch Window from Factory2 as usual.

In this example, if you want to add the attribute reference "Outlet_001.udInt3".value to the Watch Window, you must prefix it with the paired galaxy name as follows:

```
Factory1:Outlet_001.udInt3.value
```

Everything following the prefix is addressed as usual. Adding the attribute reference to the Watch Window is the same as usual. The attribute reference will appear in the Watch Window with the GR node name prefix.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.



Use InTouch with multiple galaxies

You can use an InTouch application as a client to access data in a multi-galaxy environment in the same way you would access data within a single, local galaxy.

You can use the InTouch Tag Browser to select an Application Server object as a tag source and browse the Galaxy database in a multi-galaxy environment in an almost identical fashion to the traditional browsing workflow described in the *InTouch HMI Data Management Guide*.

Application Server object attributes or attribute properties can be used in remote references or as items for InTouch I/O tags in a multi-galaxy environment.

The address syntax changes for use in a multi-galaxy environment. The following is an example of a Galaxy reference used within a single galaxy:

```
Galaxy:Pump1.PV
```

The following is the same reference with the same access name, but used in a multi-galaxy environment to target a remote galaxy named "Factory2", containing object Pump1 with attribute PV:

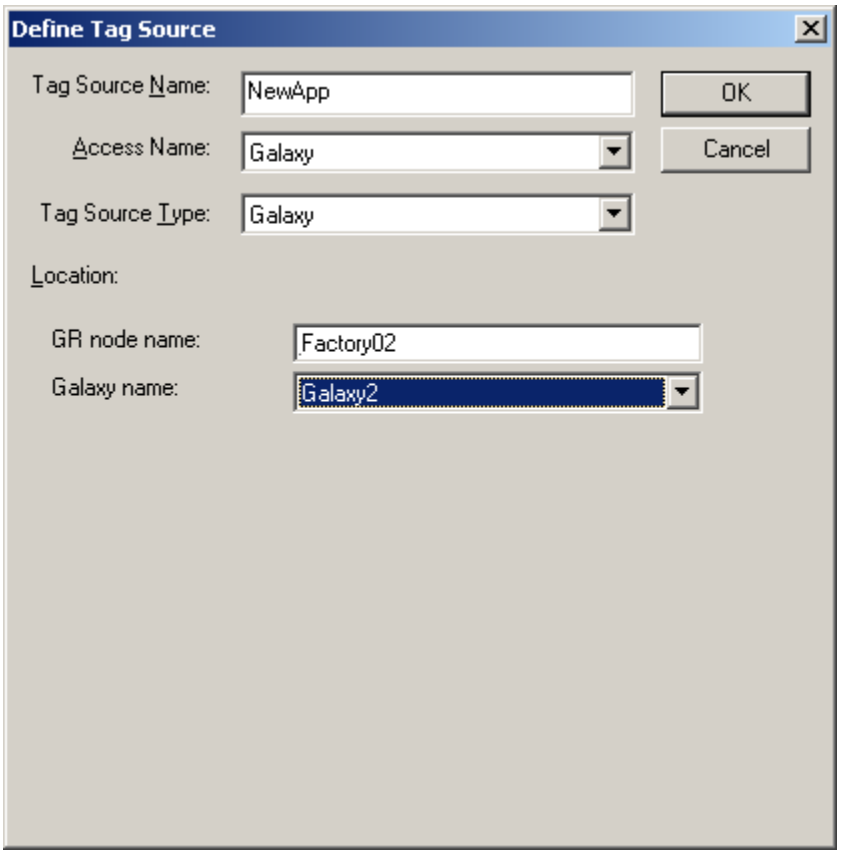
```
Galaxy:"Factory2:Pump1.PV"
```

The entire expression, enclosed in quotation marks, becomes an item name. This syntax can be used in Industrial Graphics .NET scripting, and in InTouch scripting.

Important: If you manually enter custom property overrides for an Industrial Graphic, do not enter the double quotation marks shown in the example. Browsing automatically provides the correct syntax, without the quotation marks, for custom property overrides. This syntax variation applies to custom property overrides only.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

You can define a paired galaxy as a tag source, with an InTouchViewApp leveraging the galaxy pairing already configured in Application Server.



You can then use the Galaxy Browser to access the Galaxy namespaces of the paired galaxies.

Use PCS with multiple galaxies

You can use PCS to access data in a multi-galaxy environment or in a single, local galaxy.

For details, refer to Accessing Data Using PCS in the *AVEVA Communications Drivers Help*.

Guidelines to access to multiple galaxies

Working with multiple Galaxies is basically the same as working with a single Galaxy, but there are some important variations.

The general guidelines outlined in the sections that follow can help you to understand and work with those variations.

- Security in a multi-galaxy environment
- Pair galaxies with multiple network interface cards
- Work with IDE extensions on a GR node
- Alarms in a multi-galaxy environment

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Security in a multi-galaxy environment

The multi-Galaxy environment supports the OS User and OS Group security models.

The Galaxy security model is not supported:

In order for writes to succeed in a multi-galaxy environment:

- The logged-on user must be a member of the domain, either OS User or OS Group.
- Both galaxies in the pair must be configured to use the same security model.

A user with domain security credentials in a galaxy from which a write is initiated can perform writes, secured writes, and secured and verified writes, depending on configured privilege level—to a remote galaxy without logging on to the remote galaxy.

- Use InTouch as a client to write to attributes in a remote galaxy whether or not the user has logged on to InTouch.
- Use Object Viewer to write to attributes in a remote galaxy.
- Use the MxAccess client to write to attributes in a remote galaxy.

For more information about security, see Configure security.

Pair galaxies with multiple network interface cards

When using more than one Network Interface Card (NIC), galaxy pairing can fail if the NICs are not properly configured. For more information about configuring multiple NICs, see Multiple network interface cards.

Prevent PCS services disruptions

Windows includes a power management option to place a NIC into sleep mode to save power during periods of low activity. When a NIC is placed into sleep mode, PCS services are disrupted and multi-Galaxy communication is no longer possible.

To prevent disruptions to PCS services

1. On the computer hosting a multi-Galaxy environment, click **Start** and select **Control Panel**.
2. Select **Device Manager** to show the list of installed hardware devices.
3. Click **Network Adapters** to show a list of installed NICs on the computer.
4. Right-click on a listed NIC and select **Properties** from the context menu.
5. Click the **Power Management** tab.
6. Clear the **Allow the computer to turn off this device to save power option**.
7. Click **OK**.
8. Clear this option for all nodes in your multi-galaxy environment.

Work with IDE extensions on a GR node

Multi-Galaxy configuration options are either limited or are not available in the following scenarios:

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

- **IDE running on the GR node**

  When you run the IDE on the GR node, you can only configure multi-galaxy communication for the GR on the same node. From the GR node, you can connect the IDE to a different GR to perform other configuration operations, but any attempts to perform multi-galaxy configuration will result in an informational message that the functionality is not allowed.

- **IDE running on a deployed platform node**

  When you run the IDE on a deployed platform node, you can only configure multi-galaxy communication for the GR the platform is deployed from. From the platform node, you can connect the IDE to a different GR to perform other configuration operations, but the multi-galaxy configuration dialogs will be disabled.

- **Multiple instances of the IDE on a stand-alone node**

  On a stand-alone node, if you or other users run more than one instance of the IDE, multi-galaxy can only be configured for the GR to which the IDE last connected.

  The IDE node automatically pairs with the GR it connects to. While an IDE is connected to a GR node—and paired with it—if another IDE connects to a different GR on the same computer, the IDE node pairs with the new GR. The previously running IDE will continue to function, but multi-galaxy configuration dialogs will be disabled.

By default, installing Application Server deploys the ASBGRBrowsing service. Other user-defined services are not deployed during installation. When you deploy the platform, the ASBMxDataProvider and the ASBAuthentication services automatically deploy. These are default instances which you can modify. You can configure and deploy new instances of each of these services.

Important: The multi-galaxy environment does not support buffered data.

Alarms in a multi-galaxy environment

All of the available alarm controls are unchanged, and function in a multi-galaxy environment as described in their individual user guides and online help. See the user guides for information on how to configure the various controls, how to query alarms, and how to retrieve alarm information.

- *AVEVA Alarm Client Control User Guide*
- InTouch HMI Alarm Viewer Control online help
- InTouch HMI Alarm DB View Control online help.

The alarm controls offer varied results and functional limits in a multi-galaxy environment.

AVEVA Alarm Control function and results

- Can take several seconds to display the **Alarm Statistics** dialog box with tens of alarm providers. Can take longer with 30–60 alarm providers.
- Can subscribe (query and retrieve) up to 20,000 alarms.
- Has an historical alarms limit for the maximum number of records of 32,766. Can show more than 32,766 in historical mode.
- Consumed high CPU resources at 200 alarms-per-second from four providers. During that time, might not update in a timely manner.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Alarm Viewer Control function and results

- Typically requires one second to display the Alarm Statistics dialog box under the same conditions as the AVEVA Alarm Client Control.
- Can subscribe (query and retrieve) more than 20,000 alarms.
- Can refresh the alarm list every second.

Alarm DB View Control function and results

- Alarm queries are limited to 1,958 characters. Larger queries are truncated.
- Can generate alarms from four providers at 20 alarms per second, then will start to buffer alarms.
- Logger can store 20,000 entries, then the cache is full.

We recommend using the AVEVA Alarm Client Control under the limits specified. In excess of 20,000 alarms, 20 alarm providers, and 100 alarms per second from multiple providers, we recommend using the Alarm Viewer Control.

Enhance multi-galaxy performance

Multi-galaxy communication can compound the volume of run-time subscriptions and data changes, potentially impacting system resources. You can manage potential performance degradation by creating and deploying additional service instances, particularly the ASBMxDataProvider service.

Browsing operations typically place less stress on system resources. Operations tend to be fewer in number than run-time subscriptions and are performed sequentially rather than in volume, in parallel. The ASBGRBrowsing service can be instantiated and deployed to remedy any browsing performance issues.

Operations requiring authentication, such as secured writes, also tend to place less stress on system resources than run-time subscriptions. Performance could become a factor in some scenarios, such as acknowledging a large volume of alarms where the alarm acknowledgements require secured writes. The ASBAuthentication service can be instantiated and deployed to remedy any performance issues with secured writes or other operations requiring authentication.

You can take one of two general approaches to multi-galaxy performance: remedy performance issues after the fact, or plan the multi-galaxy environment before the fact. We recommend scaling the multi-galaxy environment before the fact, particularly in large systems.

Fix performance issues after the fact

Remedying performance issues as they occur involves monitoring subscriptions for data change rates. If your system begins to slow down or starts to show communication failures or bad quality data on subscribed attributes, do the following:

- Create a new ASBMxDataProvider service instance on the publishing side
- Configure the new service instance
- Assign the new service instance to a node and deploy it

  Perform the operation on any node in the multi-galaxy environment exhibiting performance issues.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

In a smaller multi-galaxy environment, or when you are just starting to implement multi-galaxy communication, this approach can have the advantage of saving up front configuration time, and can also serve as a trial period to gather metrics about your system.

The disadvantages of this approach are that you can lose time dealing with a system that is slowing down at run time.

Scale the multi-galaxy environment for optimum performance

We recommend at least general advance planning of service and node deployment.

As a general guideline, a single instance of the ASBMxDataProvider service can subscribe up to 20,000 data changes per second. Beyond that number, you should create and deploy a new ASBMxDataProvider service on the publishing side.

An object running under an AppEngine subscribing to remote attributes or an InTouchViewApp, as a general guideline, can support up to 20,000 data updates per second from remote galaxies. Beyond that number, you should create and deploy a new ASBMxDataProvider service on the publishing side.

You can assign the new service instances to the same node as existing service instances provided each instance has a unique port number.

You can assign new nodes and deploy the new service instances to those nodes. In that scenario, a new service instance can have the same port number as an existing service instance on a different node.

Sett remote galaxy references in the owning object property

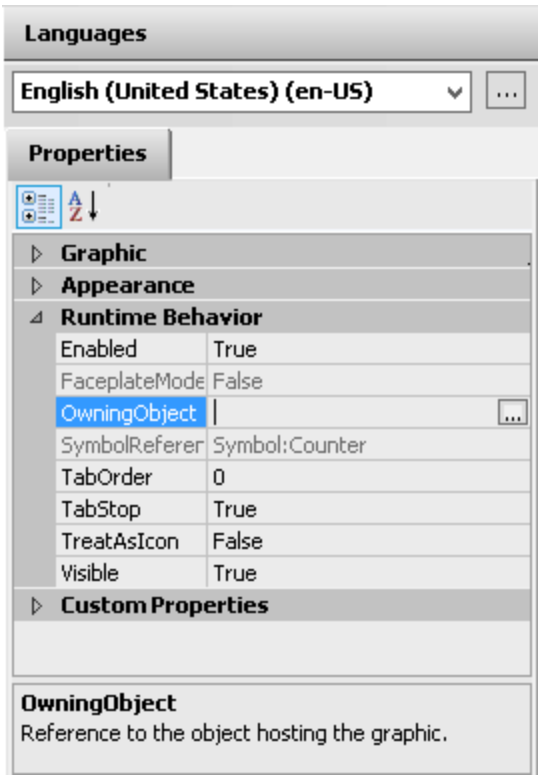
You can assign a remote galaxy reference to the owning object property of an embedded graphic during configuration through the Graphic Editor.

If needed, you can use scripting to change the owning object property during run time.

To configure the owning object property in the Graphic Editor

1. Launch the Graphic editor and embed a graphic. Select the graphic.
2. In the **Properties** pane, under **Runtime Behavior**, select the **Owning Object** property. You can set the owning object property through the Galaxy browser, or by typing the object reference in the field. Object references use the following format: *RemoteGalaxyName:ObjectName*.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.



3. To use the Galaxy Browser to set the owning object (instead of typing the reference), click the **ellipsis** button. This launches the browser.
4. From the Galaxy Browser, select the remote galaxy from the **Namespace** list of galaxy names. The objects contained in the galaxy are listed by tagname.
5. Select the object you want to set as the owing object and click **OK**.

Platform common services

Platform Common Services (PCS) provide tools for multi-galaxy communication scalability and performance. For example, you can create additional instances of the ASBMxDataProvider Service if your run-time attribute subscription volume increases to a point where it begins to impact performance. These services can be viewed when you select the **Galaxy** menu, then select **Configure** > **System** > **Services**.

One default instance of each service of the following services is provided during galaxy creation.

Service Name	Default Instance Name
ASBGRBrowsingService	Default_*<GalaxyName>*_GRBrowse
ASBMXDataProviderService	Default_*<GalaxyName>*_MXDataProvider
ASBAuthenticationService	Default_*<GalaxyName>*_Authentication
OPCUAService	Default_*<GalaxyName>*_OPCUAService
IOMBLService	Default_*<GalaxyName>*_IOMBLSService

Unless otherwise noted, instances of these services should not be renamed or modified.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

About platform common services (PCS)

Platform Common Services comprise a framework that hosts services based on general principles of a service-oriented architecture (SOA).

- SOA provides a mesh of interoperable services that can be used within a single system or extended to external systems based on an established trust relationship between the systems.
- SOA also generally provides a way for services to discover other available SOA-based services.

PCS components are installed with Application Server as part of the System Platform installation. PCS functionality appears as menu items in the IDE. For information about the PCS components in the IDE, see Setting Up a Multi-Galaxy Environment.

PCS components include a set of core run-time services, a catalog of user-configurable services, and a scheme that enables connectivity, interoperability and exchange of data among the services as well as with internal and external applications.

Define service discovery

The discovery servers provide an infrastructure for Platform Common Services to find and communicate with one another, either on the same local node or on a remote node. Multi-Galaxy communication uses three tiers of discovery servers. Note that only two of the tiers are configurable.

- **Local Discovery Server** (not configurable)

  The Local Discovery Server provides discovery for a single, local node. All services register with the Local Discovery Server, and all clients query it.
- **Local Galaxy Server** (user configurable)

  The scope of the Local Galaxy server encompasses the services in an entire galaxy. You can designate primary and secondary servers running on different nodes for redundancy.

  At least one Local Galaxy server must be configured and online to enable multi-galaxy communication.
- **Multi-Galaxy Server** (user configurable)

  The Multi-Galaxy Server encompasses the services available across multiple galaxies. You can designate primary and secondary servers running on different nodes for redundancy. In any multi-galaxy environment, there can be only one designated primary and one designated secondary Cross Galaxy Server.

  The Multi-Galaxy server can reside on any node in the multi-galaxy environment, but all galaxies in the environment must configure the Multi-Galaxy server to run on the same node.

Define user-configurable PCS services

The following user-configurable services are installed with Application Server. These services are a part of the Platform Common Services (PCS) included when you install System Platform. You can begin configuring your PCS services with the default instances of each service provided as part of the installation.

You must configure at least one instance of the OPCUAService to establish OPC UA client/server machine to machine communication within Application Server.

- **ASBMxDataProvider Service**

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

The ASBMxDataProvider service connects to Application Server/System Platform run-time components and provides access to run-time data.

- **ASBGalaxyBrowsing Service**

  The ASBGalaxyBrowsing service connects to the System Platform namespace to enable browsing of System Platform objects and attributes.

- **ASBAuthentication Service**

  Users are authenticated against Active Directory user accounts. You can configure more than one ASBAuthentication service instance to point to different domain servers as needed.

- **OPCUAService**

  The AVEVA OPC UA Service is an OPC UA server service. When you deploy this service and complete additional, required OPC UA configuration tasks, you can use a third-party OPC UA server to browse objects and attributes, and to connect, subscribe to, and retrieve data natively, that is, via an OPC UA Server, without the need for a gateway or other protocol translation mechanism. See Configure the OPC UA server for more information.

The following user-configurable services are not included with Application Server but can be installed separately. Installing these services adds default instances of each service, which you can then configure as needed.

- **Event Service**

  The Event service is installed separately and is used to raise, monitor, and route events. The Event Service is not required to enable or configure multi-galaxy communication.

- **EventHistorization Service**

  The EventHistorization service is installed separately and is used to store events in a configured database. It is not required to enable or configure multi-galaxy communication.

  The EventHistorization service stores events in a configured database. It is not required to enable or configure multi-galaxy communication.

Configure and deploy PCS services

The configuration tool and workflow are the same for all user-configurable PCS services. To configure a PCS service, you will need to perform the following tasks:

1. Create an instance of a PCS service
2. Assign the service instance to a node
3. Deploy the instance

You can make multiple instances of a single service, and deploy them to multiple nodes. You can also deploy a single instance to multiple nodes. This approach helps reduce service management. On the same machine, you can have more than one service simply by creating new instances.

Create an instance of a service

You must create an instance of an AVEVA PCS Service (for example, ASBGRBrowsing Service) to configure the service.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Assign the service instance to a node

After you create an instance of the service, you must assign the instance to the desired node. You must check out the instance before assigning it to a node. You must check in the instance after assigning it to a node.

Note: For the ASBGRBrowsingService, you can also specify the name of the Galaxy to assign an instance of the service.

Deploy the instance

After assigning a service instance to a node, you must deploy the instance. A deployment confirmation message appears.

For information about configuring specific AVEVA PCS Services, see the following topics:

- Configure the ASBGRBrowsing service
- Configure the ASBMxDataProvider service
- Configure the ASBAuthentication service

Configure service TCP ports

Important: As of AVEVA System Platform 2014 R2, there is no need to configure any TCP ports. All Platform Common Services communication is based on the Microsoft WCF shared port 808. This information is retained for existing applications built with a version of AVEVA System Platform prior to the 2014 R2 release that have been assigned custom port numbers.

On a node running an OPC UA server, you need to open two ports, one for the OPC UA server and one for the OPC UA discovery service, if configured. See the following table for more information about OPC UA port configuration.

The ASB Core Services, installed with PCS and started by the Watchdog Service, also use specific ports. The following table lists the default TCP ports assigned to the user-configurable PCS Services and the ports in use for the PCS Core Services.

- The PCS Core Services port numbers should not be used to configure instances of user services.
- The firewall must allow incoming connections to the ports in the following table for their corresponding services to function.

Service	Port Numbers in Use
User-configurable Services	
ASBGRBrowsing Service	7500, 7501 (default) See note following this table.
ASBMxDataProvider Service	3572 (default)
ASBAuthentication Service	7779 (default)

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Service	Port Numbers in Use
Event Service	3575 (default, configurable)
EventHistorization Service	3586 (default, configurable)
OPC UA Service	48031 (default, configurable) Endpoint for Application Server OPC UA Server instance
OPC UA Discovery Service	4840 (OPC Foundation service - not an ASB service)
PCS Core Services-Not configurable	
Local Discovery Server	9111
Primary Local Galaxy Server	9110
Secondary Local Galaxy Server	9210
Primary Cross Galaxy Server	9310
Secondary Cross Galaxy Server	9410
Galaxy Pairing	7085
Configuration Service	6332
Content Provider Service	6011
Deploy Agent Service	6533, 6633
Service Manager Service	6111, 6113
System Authentication Service	9876
aaServiceASBSoftwareUpdate	7587

Important: If a Galaxy Repository (GR) has more than one Galaxy, two additional ports must be opened to enable a remote GR to browse to each additional galaxy. For example, two galaxies would require ports 7500, 7501, 7502, and 7503 to be open. Three galaxies would require ports 7500-7505 to be open.

Configure the ASBGRBrowsing service

The ASBGRBrowsingService enables you to browse Application Server namespaces.

Important: The ASBGRBrowsing service is configured with one default service instance named "Default_<GalaxyName>_GRBrowse". Do not change the name of this default service instance or delete this service instance. The GRBrowsing service will stop working when you undeploy the GR Platform. If the default service instance has been renamed, revert the service instance name back to the default format. If the default

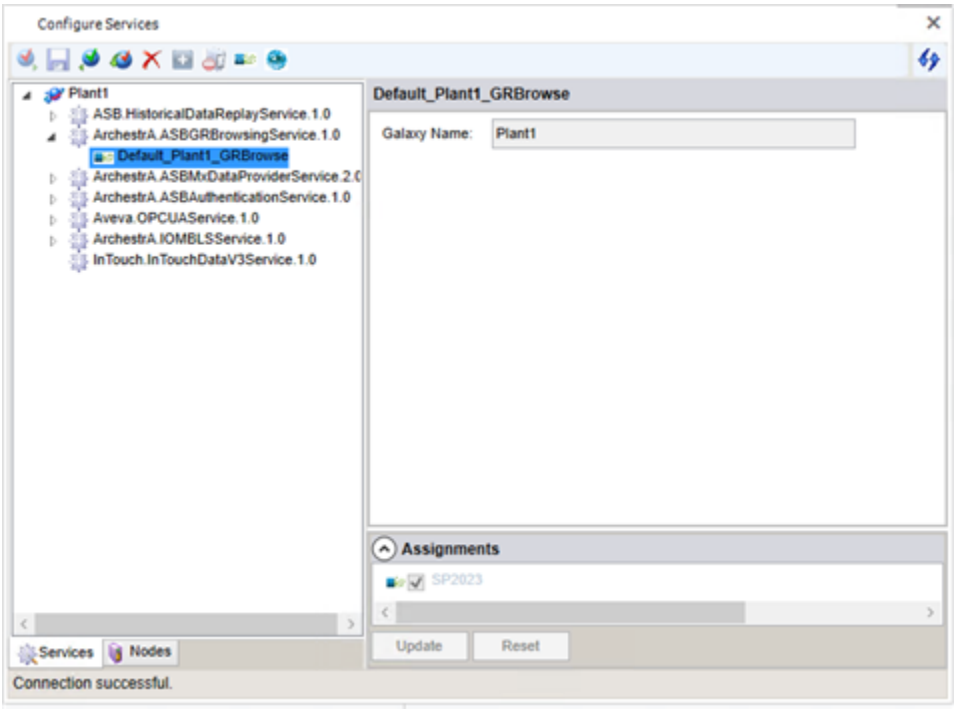
© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

service instance has been deleted, create a new instance with the correctly formatted default name.

To configure the ASBGRBrowsing Service

1. Select the **Galaxy** tab, then select **Configure.**
2. Select **System**, then **Services**.

   The **Configure Services** utility opens.



3. Expand the tree view. Right-click **ASBGRBrowsingService**, and then select **Create.** You can also select **ASBGRBrowsingService**, then type **CTRL+N**. The new instance appears in the tree structure.
4. Right-click the instance name and select **Check-out**
5. Enter the Galaxy name in the **Galaxy Name** box.
6. In the **Assignments** area, select the node you want to assign to the instance, and then select **Update**.

   Note: The **Update** button is enabled only when you select a node. You cannot delete a node that is already assigned to a service instance. To delete a node, you must first unassign the node from the service instance and then delete it.
7. On the left-pane, right-click the instance, and then select **Check-in**.
8. On the left-pane, right-click the instance, and then select **Deploy**. A message appears indicating whether the node is successfully deployed.

Configure the ASBMxDataProvider service

The ASBMxDataProvider Service acts as a gateway providing access to Application Server data.

To configure the ASBMxDataProvider Service

1. On the ribbon, select **Galaxy**, then select **Configure.**
2. Select **System**, then **Services**.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

The **Configure Services** utility opens.

3. Right-click **ASBMxDataProviderService**, and then click **Create.** You can also press CTRL+N. The new instance appears in the tree structure.
4. Right-click the instance name and click **Check-out.**
5. In the **Assignments** area, select the node you want to assign to the instance, and then click **Update**.
6. On the left-pane, right-click the instance, and then click **Check-in**.
7. On the left-pane, right-click the instance, and then select **Deploy**. A message appears indicating whether the node is successfully deployed.

Configure the ASBAuthentication service

The ASBAuthenticationService provides user authentication.

To configure the ASBAuthentication Service

1. On the ribbon, select **Galaxy**, then select **Configure.**
2. Select **System**, then **Services**.

   The **Configure Services** utility opens.
3. Right-click **ASBAuthenticationService**, and then click **Create.** You can also press CTRL+N. The new instance appears in the tree structure.
4. Right-click the instance name and click **Check-out**
5. In the **Assignments** area, select the node you want to assign to the instance, and then click **Update**.
6. On the left-pane, right-click the instance, and then click **Check-in**.
7. On the left-pane, right-click the instance, and then select **Deploy**. A message at the bottom of the window appears indicating whether the node is successfully deployed.

   Note: A runtime node will be offline if its hosted WinPlatform object is undeployed or if there is a pending software upgrade (SUP state). To change the status of the node to online and deploy new services, apply any required software upgrades and redeploy the WinPlatform object.

IOMBLSService

The IOMBLSService provides services for browsing, configuring, and deploying objects and visual elements. All the services provided by IOMBLSService are hosted in a single executable (IOMBLS.exe). IOMBLService is not user-configurable, and runs in the background. Do not delete, duplicate, close, or modify this service in any way.

Configure and deploy the OPC UA service

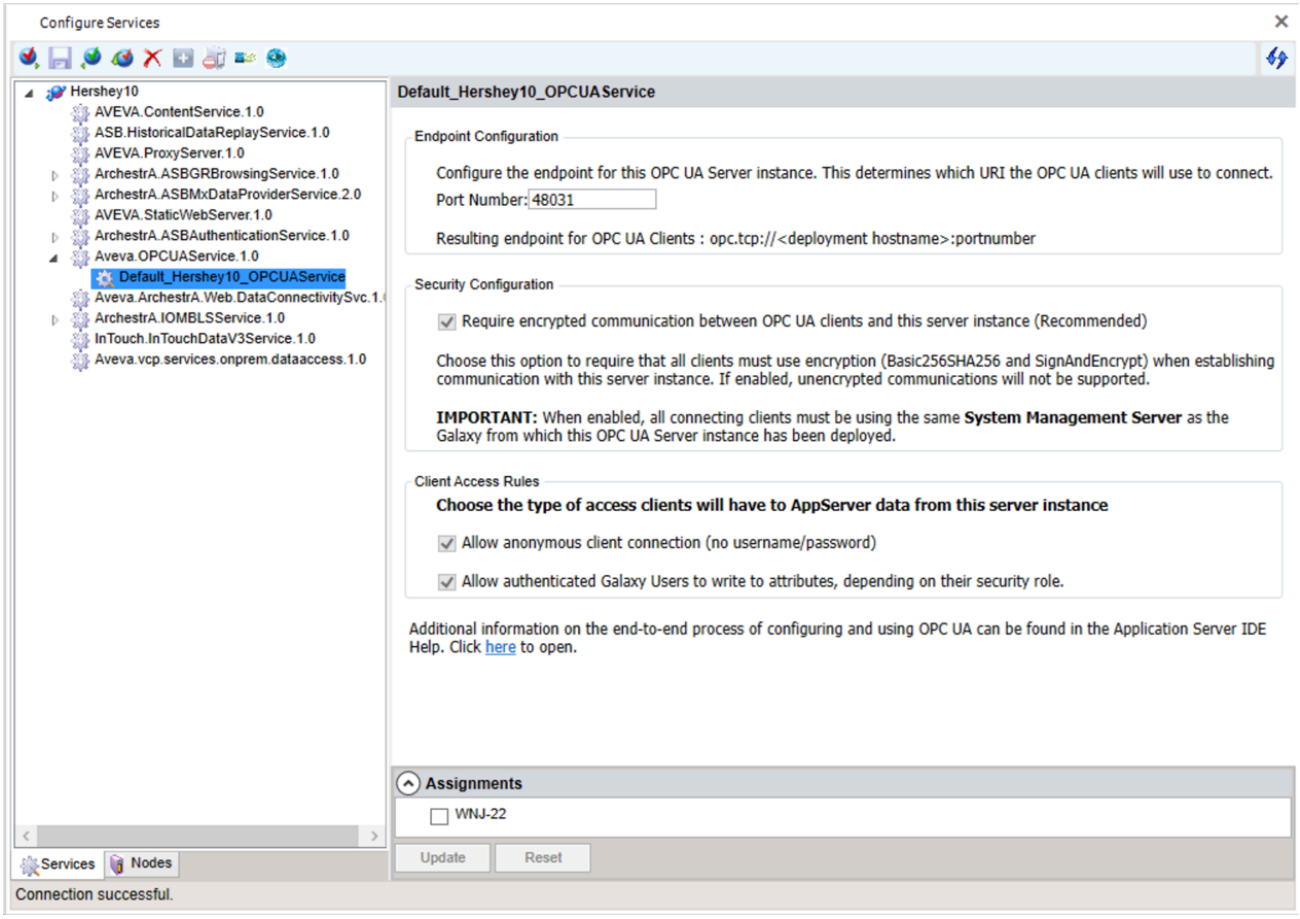
The AVEVA OPC UA Service provides access from an OPC UA client to Application Server data, without the need for the Galaxy Browser, a gateway, or other protocol translation mechanism.

To configure and deploy the OPC UA Server Service

1. On the IDE ribbon, select **Galaxy** then select **Configure**
2. Select **System**, then select **Services**.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

3. The **Configure Services** utility opens.

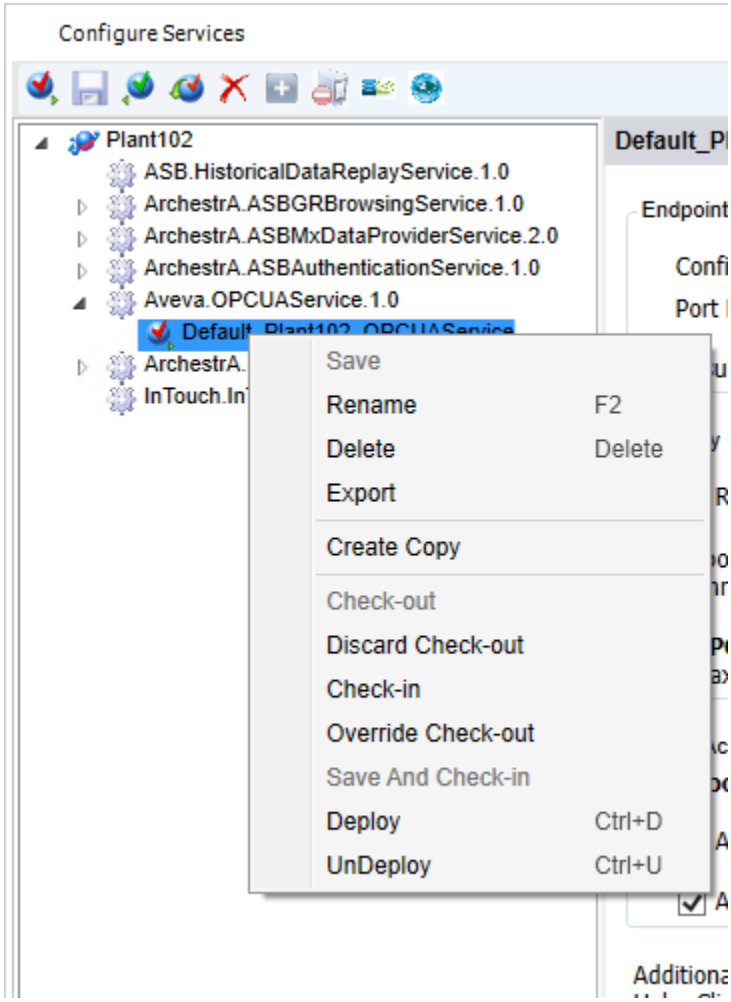


4. Expand the items in the left-hand pane as needed. Right-click the OPC UA service name and select **Check-out** from the context menu.
5. Edit the **port number** for the OPC UA instance if necessary. The default port is **48031**.
6. **Security configuration**
   - **Require encrypted communication:** It is strongly recommended that you enable this option, as this will encrypt the payloads across the connection. Note that the client must match this configuration.

     Important! An OPC UA connection cannot be established if you do not enable this option while OS security is enabled, even if the option "Allow authenticated Galaxy Users to write attributes" (under Step 7) is enabled.
7. **Client access rules**
   - **Anonymous client connection:** Allowing anonymous client connections is recommended ONLY for initial setup configurations and testing. Anonymous client connections should be disabled for production environments. Refer to Client access rules and galaxy security to see the effect that this option has on data access.
   - **Allow authenticated Galaxy Users to write attributes:** Enabling this setting will only take effect when encrypted communication is also enabled (see Step 6, above). The OPC UA client must match this configuration.
8. The **Assignments** section (below the right pane) shows both the local node and deployed remote runtime nodes to which the OPC UA service can be assigned. Select the checkmark next to the runtime node where you want to deploy the OPC Server service, and then click **Update**.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

9. In the left pane, right-click the OPC UA service instance name and select **Check-in** from the context menu.



10. Right-click the instance again to open the context menu and select **Deploy**, or press CTRL+D. A message appears indicating whether the service has been successfully deployed to the OPC UA client node. If deployment is successful, the icon next to the instance name changes to indicate that the instance has deployed.
11. If you have deployed the service to a remote node, you can verify functionality through the Operations Control Management Console.
    a. Under Operations Integration Server Manager, select the remote node name.
    b. Expand Operations Integration Supervisory Servers and select the OPCUA Client (under OI.GATEWAY.3). Check the following settings:
        a. Server Node: localhost
        b. OPCUA Server: opc.tcp://localhost:48031

To add additional OPC UA services

Each OPC UA service is dedicated to a single OPC UA client node. To add additional OPC UA services:

1. Right-click **AVEVA.OPCService**, and then select **Create** from the context menu, or press CTRL+N. The new instance appears in the tree structure.

Note: Each instance must have a unique port number. Enter the port number in the **Base Address** field. The

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

default port number is **48031**. See Configure service TCP ports for a list of port numbers used by ASB services.

2. Rename the OPC UA service as needed. Right-click on the service name and select **Rename** from the context menu, or press F2. Then, enter the new name.
3. Repeat the steps above for configuring and deploying each additional OPC UA service.

To change a deployed OPC UA service

1. Check out the service instance.
2. Make any needed changes.
   - **Port Number:** If you are creating multiple services, each service instance should have a unique port number. If more than one service has the same port number, an error is generated in the logger. Multiple instances of the service can be deployed, as long as each service has a unique port number. A new URI (uniform resource identifier) is automatically generated when a port number is changed.

     Note: You may need to open the inbound port in the firewall to allow communication with the remote node.
   - **Security Configuration:** When enabled (default), communication between OPC UA clients and the OPC UA server is encrypted. This is the recommended setting. If this setting is unchecked (disabled), communication is not encrypted.
   - **Client access rules**

     Allow anonymous client connection (enabled): When enabled (default), an anonymous OPC UA client is allowed to connect to the OPC UA server. This is recommended only for testing and initial set up configurations. Once you have completed configuration and/or testing, be sure to disable this setting to provide protection against possible unwanted intrusions and to ensure that only authenticated users have access. Anonymous client connection should not be enabled in a production environment.

     Galaxy Security settings do not have any affect on these behaviors. See Configure security for more information.

     Allow authenticated Galaxy user to write to attributes (enabled): When enabled (default), an authenticated Galaxy user can change attribute values in runtime, if their security role allows them to do so. See About roles for more information.

     Allow authenticated Galaxy user to write to attributes (disabled): When disabled (unchecked), an authenticated Galaxy user is not permitted to change attribute values in runtime, even if their security role allows them to do so.

     See Client access rules and galaxy security for more information about user permissions for each setting combination.
3. Check in the service or services.
4. Undeploy and then redeploy the service or services.

Client access rules and galaxy security

Client Access Rules configured for the OPC UA Service interact with the Galaxy security authentication mode to allow or deny different levels of access for authorized users.

There are two configurable Client Access Rules in the OPC UA Service dialog. By default, both rules are enabled:

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

- Allow anonymous client connection (no username/password)
- Allow authenticated Galaxy Users to write to attributes, depending on their security role

The following table defines the level of data access users are allowed under different combinations of Client Access Rule configurations, Galaxy security authentication mode, and the type of OPC UA credentials (anonymous or authenticated user with username/password).

If security for the Galaxy is enabled (Galaxy security = **Secured**, column **1**), encrypted communication between the OPC UA clients and OPC UA service must also be enabled. See Configure and deploy the OPC UA service.

Galaxy security	OPC UA	Client access rules		Level of data access		
Authentication mode	Client credentials	Allow anonymous connection	Allow authenticated Galaxy users	Connect	Read	Write (see below)
Secured	Authenticated	Enabled	Enabled	YES	YES	YES
Secured	Authenticated	Enabled	Disabled	YES	YES	NO
Secured	Authenticated	Disabled	Enabled	YES	YES	YES
Secured	Authenticated	Disabled	Disabled	YES	YES	NO
Secured	Anonymous	Enabled	Enabled	YES	YES	NO
Secured	Anonymous	Enabled	Disabled	YES	YES	NO
Secured	Anonymous	Disabled	Enabled	NO	N/A	N/A
Secured	Anonymous	Disabled	Disabled	NO	N/A	N/A
None	Authenticated	Enabled	Enabled	NO	N/A	N/A
None	Authenticated	Enabled	Disabled	NO	N/A	N/A
None	Authenticated	Disabled	Enabled	NO	N/A	N/A
None	Authenticated	Disabled	Disabled	NO	N/A	N/A
None	Anonymous	Enabled	Enabled	YES	YES	NO
None	Anonymous	Enabled	Disabled	YES	YES	NO
None	Anonymous	Disabled	Enabled	NO	N/A	N/A
None	Anonymous	Disabled	Disabled	NO	N/A	N/A

Important: Whenever Client Access Rules and Galaxy Security allow a user to write data, this permission is always conditioned by whether or not the user's configured security role also allows them to write data to a specific attribute. This means that when Galaxy security is enabled, the user's security role must explicitly allow them to write to attributes, regardless of the OPC UA client access rule setting. If their security role does not

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

allow them to write to attributes, they cannot, even if the level of data access in the above table shows that they can.

Manage galaxies in a multi-galaxy environment

The workflow for backing up and restoring a galaxy in a multi-galaxy environment is unchanged from those operations in a single-galaxy environment.

There are some important differences in behavior in galaxy management operations performed in a multi-galaxy environment:

- Uninstall/reinstall Application Server
- Back up a galaxy
- Delete a galaxy
- Restore a galaxy
- Upgrade a galaxy
- Migrate a galaxy
- Upgrade SQL Server

Uninstall/reinstall Application Server

When you uninstall Application Server, the services database is not deleted from the system. This means that any existing galaxy pairings will persist when you reinstall Application Server on the same system. The persistent pairing information is automatically available after the reinstallation.

This pairing information, however, can be out of date and multi-galaxy operations not working as the Discovery Service information is lost in the transition from uninstall to install.

To resolve out-of-date pairing information, you must unpair and re-pair to the required galaxies after performing uninstall and re-install operations on the same node.

For example:

1. Install the latest Application Server software on two nodes, Node1 and Node2.
2. Create a galaxy with a platform, and pair the nodes.
3. On Node1, uninstall Application Server and restart the system.
4. On Node1, reinstall Application Server and connect to the same galaxy.
5. Open the **Multi-Galaxy Configuration** dialog. Notice that all the pairing information has persisted through the uninstall-reinstall operation. Also, notice that multi-galaxy operations are not functioning. You can't browse to or subscribe from Node2 even though the nodes appear to be paired.
6. Unpair and pair Node1 to get the new Discovery Service and pairing information from Node2.

Back up a galaxy

Use the Galaxy Database Manager in the OCMC to perform a galaxy backup. The backup workflow is unchanged. There are minor differences in some details:

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

- The .cab file XML (_GalaxyInfo.txt) has changed to include the backup node name.
- The file repository folder of the .cab file now contains a file, <gr node name><galaxy name>_SR.csv, that lists all nodes and services configured for the galaxy whose backup the .cab file represents. The galaxy restore process uses this .csv file to recreate ASB nodes and services in the target galaxy.

You can back up and restore to a local node. You can back up and restore to a remote node.

Note: Galaxy pairing information is not included in the galaxy backup.

Delete a galaxy

Deleting and restoring a galaxy requires that there be no clients connected to it. In a multi-galaxy environment, any deployed browsing service is considered a connection.

The following preparatory steps occur prior to initiating the galaxy delete operation:

1. Use the **Configure Services** editor to undeploy and delete all service instances. The Service instances created for a galaxy are not deleted when the galaxy is deleted.
2. Use the **Configure Services** editor to undeploy all browsing services associated with the galaxy being deleted. If clients are connected, the delete operation is aborted and all browsing services that were previously undeployed are redeployed.

If the delete operation can proceed, all services, including the browsing services, are deleted.

Restore a galaxy

The galaxy restore workflow remains unchanged. You can do the following restore operations:

- Restore to an existing galaxy
- Restore o a new galaxy

Restore to an existing galaxy

If the existing galaxy to which you are restoring already has deployed browsing services, the browsing services are automatically undeployed as a first step in the restore process.

These browsing services are redeployed if the galaxy restore operation cannot proceed for any reason. Reasons for a galaxy restore operation being aborted:

- There are clients connected to the galaxy.
- The galaxy has a deployed platform.

At the end of the restore process, the service nodes and instances in the existing galaxy are deleted and replaced by new ones defined in the .cab file.

The node corresponding to the GR node in the backup replaces the GR node being restored.

The workflow for restoring Galaxy2 to Galaxy1 remains unchanged. Use the OCMC to restore the galaxy, and target the existing galaxy, Galaxy1, to be replaced. The destination galaxy name will remain unchanged, even though the nodes will be replaced with what is defined in the .cab file.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Restore o a new galaxy

Typically, you use the Galaxy Database Manager in the OCMC to perform a restore operation. Specify the backup .cab file to be restored, and provide a galaxy name. You are restoring a backup, but the result is a new galaxy.

The exception to this workflow is when you must rename a galaxy to ensure that all galaxies in the multi-galaxy environment have unique names. For information about the renaming scenario and procedure, see Galaxy names in a multi-galaxy environment.

Upgrade a galaxy

Back up your galaxies before performing an Application Server software upgrade.

When you back up the upgraded galaxy, the backup .cab file XML will have the node name. The .csv file will contain the list of services and instances as they exist in the upgraded galaxy.

Migrate a galaxy

The migration workflow remains unchanged, but has an additional operation at the end of the migration process specifically for ASB.

Migrating from a non-ASB galaxy to an ASB-enabled galaxy, by default, does not create service instances and nodes during the migration process.

Creation of ASB service instances and nodes is deferred until the end of the migration process.

Upgrade SQL Server

We recommend undeploying Application Server PCS services before you upgrade SQL Server on your computer.

To upgrade SQL Server

1. Undeploy the galaxy.
2. Undeploy the galaxy services:
   a. Select the **Galaxy** from the menu, then **Configure.**
   b. Select **System**, then **Services**. The **Configure Services** window appears.
   c. Select the galaxy name in the left pane. Right-click and select **Undeploy** from the context menu, or click the **Undeploy** icon on the toolbar.
   d. Verify that each service instance is undeployed, then close the configuration window.
3. Follow the relevant Microsoft instructions to install the new version of SQL Server.
4. When complete, you can deploy the services, then deploy the galaxy.

Use the OS Configuration utility to fix communication issues

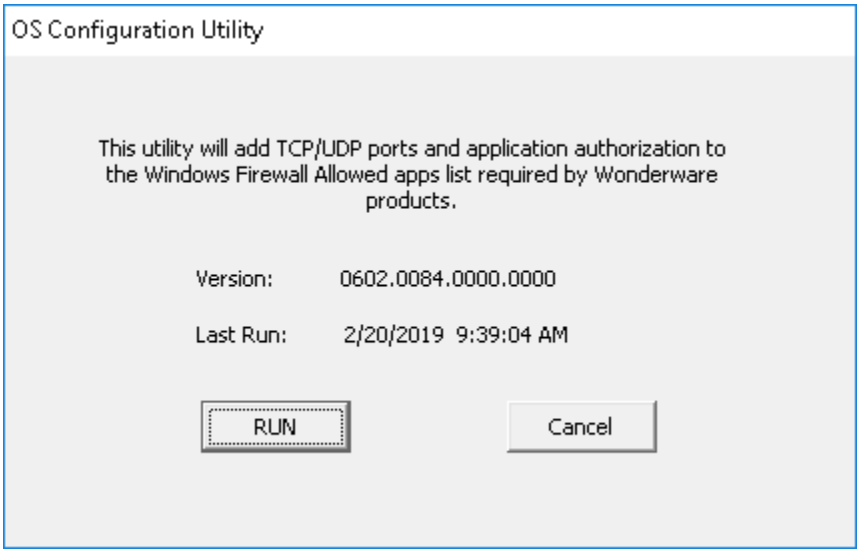
The **OS Configuration Utility** runs automatically as part of the System Platform installation process. In some cases, you may need to run the utility manually to correct communication issues within Application Server. Running the OS Configuration Utility may be necessary if you have problems running the IDE, connecting to a

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

galaxy, or deploying a node. The full pathname of the OS Configuration Utility is (there is no start menu shortcut to the utility):

<Root drive>:\Program Files (x86)\Common Files\ArchestrA\OSConfigurationUtility.exe

The utility will open ports and change registry values to fix any communication issues it finds. Other than starting the utility, all updates are automatic. No user interaction is required or allowed, other than restarting the system after the utility has finished running.



Enable buffered data

Buffered data is data captured and stored locally on a remote device for later transfer to a supervisory system for processing, analysis, and long-term storage. The Buffer property is input-only.

Enabling buffered data alters the run-time behavior of some Application Server features such as History and Alarms. Buffering is essential if the Telemetry Server is installed.

Application Server includes the option to enable buffered data for the entire Galaxy by setting the enable auto-buffering configuration option. See Auto-buffering for additional information.

The buffered data feature works identically for both Galaxy-wide and individually-enabled buffering for attributes. In other words, if you have enabled auto-buffering for the Galaxy, the feature works the same as it does when you enable data buffering for individual attributes.

The buffered data feature enables efficient accumulation and propagation of VTQ (Value, Time, and Quality) data updates, without folding and data loss, to data consumers such as objects, alarms, the Historian, and scripts from field devices that support buffering. If the Telemetry Server is used with Application Server, buffering must be enabled to ensure that all data from the Telemetry Server is received by Application Server, and then sent to the Historian in its entirety.

Enabling the buffered data feature preserves all intermediate data values. Thus, if an attribute receives different values during a single scan cycle, the data is not folded or compressed. Data folding occurs when an attribute value changes multiple times within a single scan, and only the latest value is stored. With buffered data enabled, the entire subset of values is propagated to data subscribers such as the alarm subsystem and the Historian, if the attribute is configured to be alarmed or historized. Scripts can also take advantage of processing buffered data. Since the Telemetry Server only sends a data request periodically, and often only once per day, enabling buffered data ensures that Application Server preserves all intermediate data values.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Note: If multiple attributes with I/O enabled are configured for buffered data from the same data point in the Telemetry Server, only one attribute will receive the buffered data at run time.

Components that use buffered data

The following table summarizes component functions related to buffered data subscription and processing.

Subscribing to the Buffer property	
Attribute	User defined attributes with I/O configured to read or read/write data can support buffered data processing. When buffered data is enabled, the attribute subscribes to the buffer property and receives the buffered data via Message Exchange. See Message exchange and attributes for more information about Message Exchange.
Scripts	A script subscribes to an attribute and receives buffered data from it. A specific syntax is used to properly process buffered data in scripting. Scripts can also read the buffer from an attribute or directly from the DIObject. For more information, see Use scripts to process buffered data.
Processing buffered data	
Communication Drivers	Communication Drivers collect and store VTQ data from field devices. Communication Driver functionality is unchanged by the buffered data feature. Protocol interfaces within the servers allow buffered data from the field devices to be processed. Communication Drivers, in turn, can pass the buffered data as packets to the Application Engine.
Telemetry Server	The Telemetry Server collects and stores VTQ data from RTUs (remote terminal units) connected to field devices. The Telemetry Server typically only sends data to Application Server when polled. Since polling may be performed only periodically, there may be a large dataset of intermediate values that would be lost if buffering is not enabled.
DIObject	For attributes with a buffer property that is being subscribed, the DIObject (Data Integration Object) collects VTQ values from Communication Drivers, and packs the VTQ values into buffer collections, based on scan cycle completion or on end of data received.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

	Typically, the PLCClient object is used for data integration.
Application Engine	The Application Engine hosts and executes the ApplicationObjects' logic.
Data Consumers	When buffered data is enabled, data consumers such as the Historian and alarms subsystems subscribe to the Buffer property of the configured attribute to receive published buffered data. The buffered data, a stream of VTQs, is assembled and appended in caches to avoid data folding, and is then published through Message Exchange to consumers such as the Historian, alarms and scripts. See Message exchange and attributes for more information.

Note: The buffered data feature does not support out-of-sequence data. Out-of-sequence data occurs when the time stamps of the VTQ sequence for a specific reference is not arranged in incremental order from older to newer.

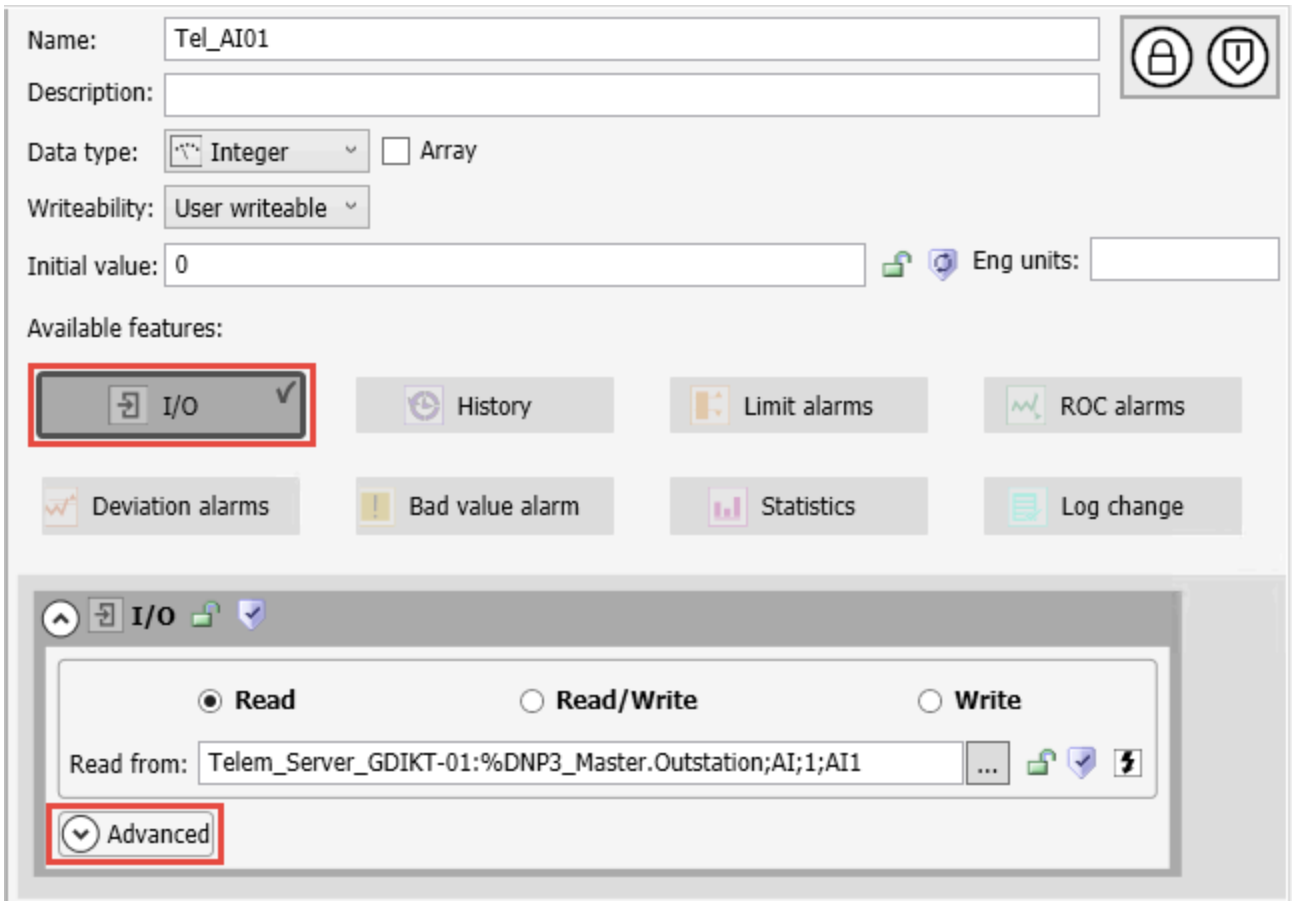
Configure buffered data for an attribute

Note: If auto-configured data buffering is enabled, the buffered setting for individual attributes is ignored. See Auto-buffering for additional information.

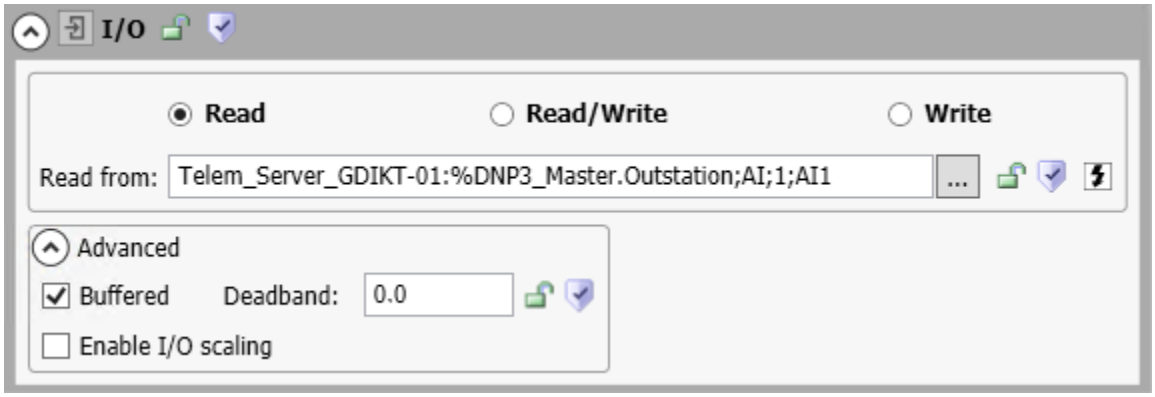
To configure buffered data for an attribute:

1. Open an object for editing and select the **Attributes** page.
2. Select an existing user-defined attribute or add one.
   Buffered data can be configured for all attribute data types except InternationalizedString.
3. Enable the I/O feature for the attribute and select Read or Read/Write, as appropriate.
4. Click the arrow to expand the **Advanced** I/O configuration options.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.



5. Select the checkbox next to **Buffered** to enable data buffering.



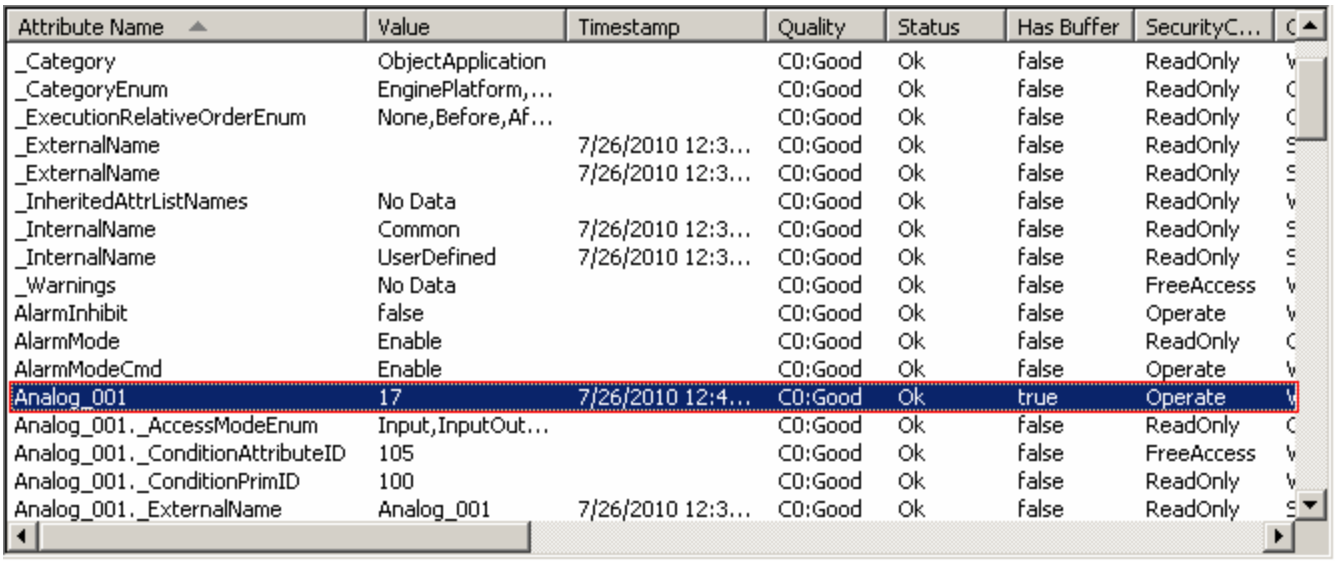
Select and modify other advanced options as needed. See Enable I/O for an attribute for more information.

Use object viewer with buffered data

You can use the Object Viewer utility to view the attribute values of a selected object at run time.

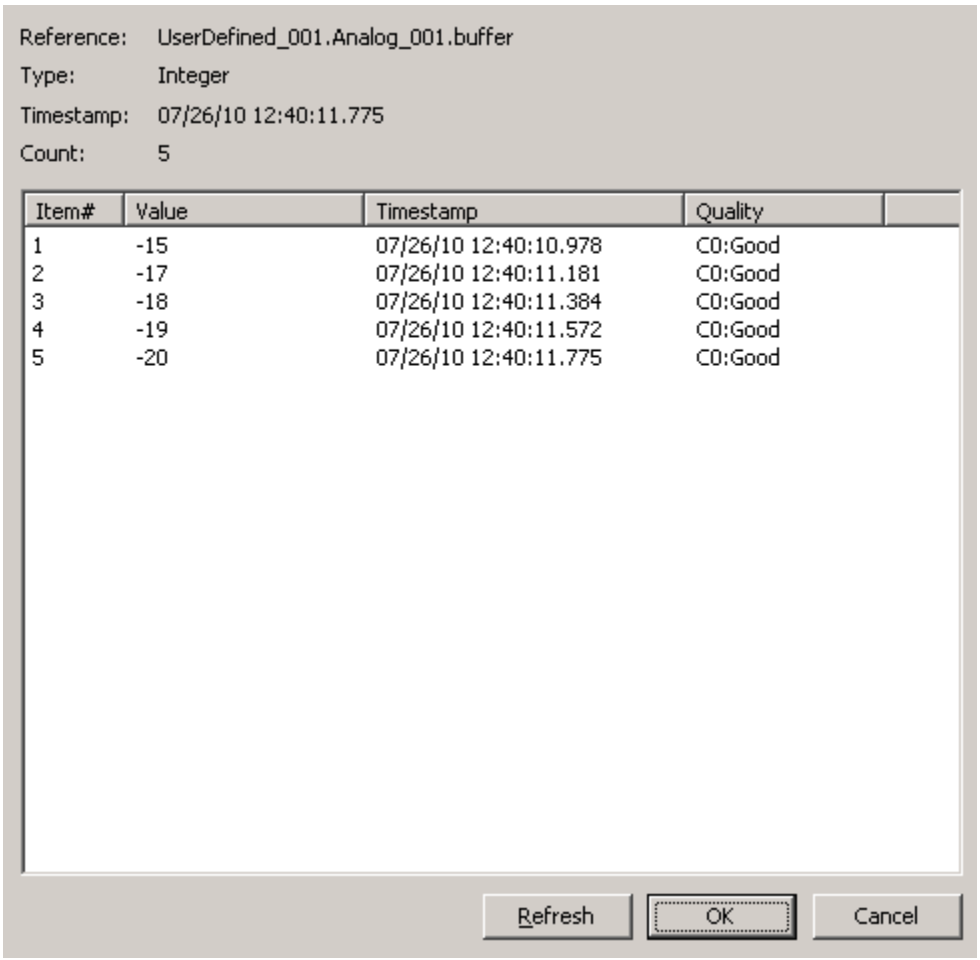
The **Object Viewer Attributes List** displays a **Has Buffer** column to reflect whether an attribute has a buffer. The value displayed in the column is True if the attribute has a buffer and False if the attribute has no buffer, as shown in the following illustration.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.



Attribute Name	Value	Timestamp	Quality	Status	Has Buffer	SecurityC...
_Category	ObjectApplication		C0:Good	Ok	false	ReadOnly
_CategoryEnum	EnginePlatform,...		C0:Good	Ok	false	ReadOnly
_ExecutionRelativeOrderEnum	None,Before,Af...		C0:Good	Ok	false	ReadOnly
_ExternalName		7/26/2010 12:3...	C0:Good	Ok	false	ReadOnly
_ExternalName		7/26/2010 12:3...	C0:Good	Ok	false	ReadOnly
_InheritedAttrListNames	No Data		C0:Good	Ok	false	ReadOnly
_InternalName	Common	7/26/2010 12:3...	C0:Good	Ok	false	ReadOnly
_InternalName	UserDefined	7/26/2010 12:3...	C0:Good	Ok	false	ReadOnly
_Warnings	No Data		C0:Good	Ok	false	FreeAccess
AlarmInhibit	false		C0:Good	Ok	false	Operate
AlarmMode	Enable		C0:Good	Ok	false	ReadOnly
AlarmModeCmd	Enable		C0:Good	Ok	false	Operate
Analog_001	17	7/26/2010 12:4...	C0:Good	Ok	true	Operate
Analog_001._AccessModeEnum	Input,InputOut...		C0:Good	Ok	false	ReadOnly
Analog_001._ConditionAttributeID	105		C0:Good	Ok	false	FreeAccess
Analog_001._ConditionPrimID	100		C0:Good	Ok	false	ReadOnly
Analog_001._ExternalName	Analog_001	7/26/2010 12:3...	C0:Good	Ok	false	ReadOnly

Add the **<attributename>.buffer** property to the **Object Viewer Watch Window**. The **Value** column displays the message <<Double-click to display buffer>>. Double-click the attribute to view the buffer elements. The **View Buffer Elements** dialog box appears.



The **View Buffer Elements** dialog box header displays the Attribute Reference, Type, Timestamp, and Count. A details window displays Value, Timestamp, and Quality (VTQ) columns for each item. The data is displayed as a snapshot. Refresh the window to retrieve another snapshot.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

The source attribute must also have buffered data enabled. If buffered data is not enabled on the source attribute, the data quality will return as bad, and a configuration error message is displayed.

See Has buffer in the *Object Viewer User Guide* for more information.

Use scripts to process buffered data

Scripts can be synchronous or asynchronous. The synchronous mode is the default choice and represents serial script execution by the AppEngine in the course of calling the Execute method of all ApplicationObjects that are on-scan in the AppEngine.

Synchronous mode requires that all scripts execute deterministically and quickly enough to prevent an AppEngine over-scan condition.

Important: For scripts that process buffered data, it is strongly recommended that you configure scripts as asynchronous execution type. Processing buffered data may be time consuming and exceed the script execution time limit. See Scripting tips and best practices for more information.

Script basic functions

The following is a collection of basic functions used to read buffer data from the script accompanied by script snippets.

Declaration	
	`dim vtq as ValueTimeQuality;`
To access samples in each buffer	
	`FOR EACH vtq in me.U1.buffer`
	Note: The only syntax that can be used to process buffers in scripts is FOR EACH.
To write the buffer to text file	

	Declaration:
	`dim strWriter as System.IO.StreamWriter;`
	Usage:
	`strWriter = System.IO.File.AppendText("C:\\sample.txt");`

To write the buffer to Logger (or OCMC)	
	LogMessage();

Sample script and output

Read buffered data from the VTQ buffer

```
dim vtq as ValueTimeQuality;
```

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

```
dim strWriter as System.IO.StreamWriter;
LogMessage("Script Start");
   strWriter = System.IO.File.AppendText("C:\\sample.txt");
FOR EACH vtq in me.UD1.buffer
   strWriter.WriteLine("DAS value is "+vtq.value);
   strWriter.WriteLine("Time stamp of the value is "+vtq.time);
   strWriter.WriteLine("Quality of the value is "+vtq.quality);
   strWriter.WriteLine("-------------------------------------");
   ‘Below messages will be logged to the Logviewer (or OCMC)
   LogMessage("DAS value is "+vtq.value);
   LogMessage("Time stamp of the value is "+vtq.time);
   LogMessage("Quality of the value is "+vtq.quality);
next;
   strWriter.Flush();
   strWriter.Close();
LogMessage("Script End");
```

For each VTQ in this script sample, Value, Timestamp, and Quality are written to the sample.txt file.

For purposes of illustration, UD1 in the script sample is a buffer-enabled Integer attribute with input extended to a DDE item.

Sample output text written to text file

```
DAS value is 2
Time stamp of the value is 7/15/2010 11:28:54.880 AM
Quality of the value is 192
```

Sample output text written to logger

```
Info ScriptRuntime UD1.S1: Script Start
Info ScriptRuntime UD1.S1: DAS value is 2
Info ScriptRuntime UD1.S1: Time stamp of the value is 7/15/2010 11:28:54.880 AM
Info ScriptRuntime UD1.S1: Quality of the value is 192
Info ScriptRuntime UD1.S1: Script End
```

Scripting tips and best practices

The topics in this section give some suggestions to help you get the most out of using scripts to process your buffered data.

Use asynchronous scripts for buffered data

Synchronous scripts may read buffered data. However, if the buffer is expected to take too much time for processing it would be better to mark the script as asynchronous.

Buffered data script syntax and configuration

Scripts can process buffered data by using the FOR EACH syntax to iterate over an attribute’s Buffer property.

To process all the buffers without any loss, asynchronous scripts need to have the following configuration:

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

- Execution type: Execute
- Trigger type: While True
- Expression: True
- Time period: 00:00:00.0000000

BindTo calls for buffered data

When working with indirect scripts to read the .buffer property of an attribute, the BindTo call needs to be used in startup instead of using it in execute. Using BindTo in execute can lead to losing the data in alternate buffers.

- Using artificial delays in the script is not recommended as this may result in script timeouts and would in turn affect UI refresh. For example, do not use a while loop that increments a variable from 1 to <60000. This could cause the ViewApp screen to go blank or become unresponsive while the script is executing.
- Instead of artificial delays, allow the whole script to continue execution across multiple cycles, without any artificial delays in the script, and set the script's expression to false once the quality of all the attributes is good.

When binding to an on-engine attribute, the whole script could be put in Execute. The following snippets show an example of using an indirect to process a buffer. In this example, the BindTo call is in Startup and FOR EACH is in Execute.

```
In Declarations:
dim x as indirect;
In Startup:
x.BindTo("me.attribute1.buffer");
In Execute:
dim vtq as ValueTimeQuality;
for each vtq in x
   LogMessage(vtq.value);
next;
```

When binding to an off-engine attribute, it may take more than one scan cycle to bind and the quality of the Indirect variable must be inspected before attempting its use. When binding to an off-engine attribute, declare the reference as indirect, and test the off-engine reference quality in the body of the script. You can use a WHILE-triggered script to ensure execution, illustrated in the following example. For more information, see Bind to off-engine attributes.

```
In Declarations:
dim x as indirect;
dim y as boolean;

In Execute (while true: me.z):

if not y then
   'This could also be done in the startup script
   x.BindTo("Object1.attribute1");
   y = true;
endif;

if IsGood(x) then
   LogMessage(x);
   me.z = false;
endif;
```

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Buffered data run-time behavior

This following table describes how enabling buffered data affects the run-time behavior of the Scan Group, Inputs, and Redundancy for all phases:

- On Startup
- On Scan
- Execute
- Off Scan
- Shutdown

History run-time behavior and Data and Alarms run-time behavior also change when buffered data is enabled. The run-time behavior of those features is described in About buffered data and history and About buffered data and alarms and events.

Important: Buffered data is not supported in a multi-galaxy environment.

Feature	Phase	Behavior
Scan Group		When the buffer property of a scan group's dynamic attribute is subscribed to at run time, the buffer data support is enabled. The scan group accumulates all of that item's updates into a buffer of VTQs. The scan group supports buffering up to 10,000 VTQs per item per scan. Higher data throughput rates are not supported
	On Scan	When the object is placed On Scan, the system enables all available items and resets item error count.
	Off Scan	When the Scan Group goes Off Scan, it will clear all buffers associated with each item in the Scan Group. When buffered data is enabled and the host object goes Off Scan, a VTQ with Bad Quality is added to the buffer. On a subsequent scan, the buffer property of the **Value** attribute is cleared.
Inputs	On Startup	When buffer is enabled on the attribute, buffer support for the

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Feature	Phase	Behavior
		input is enabled. No data is processed during this phase.
	Execute	The buffered attribute is registered, but no data is processed or stored until the buffer reference finishes initializing.
	Shutdown	Interfaces are unregistered. (No change in behavior for buffered data-enabled.)
Redundancy		**Application Engine** Buffered attributes are not synchronized with a redundant partner. When a failover occurs, unprocessed buffered data is lost.
		Redundant DIObject When buffering is enabled, the RedundantDIObject (RDI) subscribes to the buffer property of its DI source items. On failover, all advised items are moved to the backup node, whether or not buffered data is enabled. Buffered attributes are not synchronized with a redundant partner. When a failover occurs, unprocessed buffered data is lost.

About buffered data and history

When an ApplicationObject starts up, if configured to save historical data, it registers its configuration data with the Historian using a Historian supplied interface.

The following table describes buffered data-enabled history behavior.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

History function	Behavior
Buffer Monitoring	The ApplicationObject monitors and historizes the attribute value. If the attribute being historized has a buffer property, the ApplicationObject monitors the attribute's buffer instead of its value, and historizes each VTQ in the buffer.
Attribute Registration with the Historian	Attributes that are buffer-enabled are registered with the Historian in the same manner as non-buffered attributes are registered.
Push Attribute Changes to the Historian	After successful registration of an attribute with the Historian, the buffer-enabled attribute pushes the data changes to the Historian. If the buffer-enabled attribute reads an empty buffer, and if this is the first scan cycle after the attribute was registered, it gets the attribute's Value, Time, and Quality properties, and if Quality is Good, sends this VTQ to the Historian. If the buffer-enabled attribute reads an empty buffer, and if this is not the first scan cycle after the attribute was registered, it does nothing. For more information, see Configure historization.

About buffered data and alarms and events

Application Server's alarm and event capabilities automate the detection, notification, historization, and viewing of either application alarms and events or system alarms and events.

An attribute, when configured for alarming and buffer-enabled, can receive a packet of VTQs in a single scan. This packet can contain values that will trigger an alarm condition. Alarm conditions will be generated based on the VTQs in the buffer.

Important: Alarms are not supported in this release of Telemetry Server.

AppEngine reconnect configuration for objects linked to the telemetry server

If an object with an attribute linked to a Telemetry Server is undeployed and then redeployed, you may see a gap in the timestamps that are historized by the Historian for the data collected between the undeployment and redeployment times. This can happen, even when buffering is enabled. Note that all data values are accurately preserved, but the timestamps for the values will not be accurate. This is a function of the behavior of the Historian and the default configuration of the AppEngine.

When an AppEngine or objects connected to it are undeployed and then redeployed, all values from the Telemetry Server are collected and sent to the Historian upon redeployment. The Historian gives the current timestamp to the first value received after redeployment. This leaves a gap between undeployment and redeployment times. The Historian adds 5 ms from the time of redeployment to all timestamps for values (after

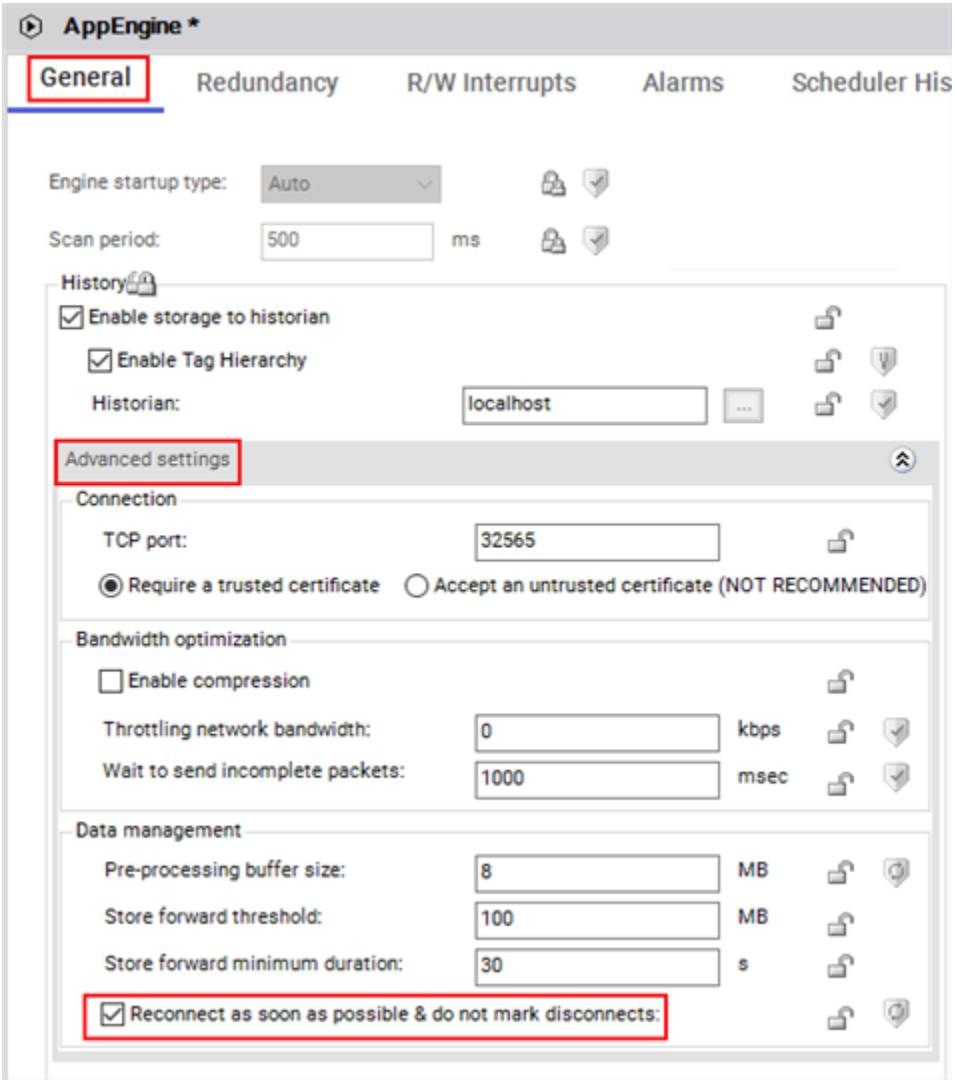
© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

the first value) that were collected during the undeployed time. Thus, subsequent values have timestamps of redeploy time + 5 ms, redeploy time + 10 ms, etc.

To eliminate this gap in timestamp values for data collected during undeployment, configure the **Reconnect** setting under **Data management** in the AppEngine that contains the object. When the Reconnect setting is enabled, all values will be historized with the correct timestamp.

To configure the AppEngine Reconnect setting

1. Locate the AppEngine that contains the object with the Telemetry Server-connected attribute.
2. Open the AppEngine for editing.
3. In the **General** tab, expand **Advanced Settings**.



4. Under **Data management**, enable the setting, **Reconnect as soon as possible & do not mark disconnects**.
5. Save and close the AppEngine object.

Alarm functions and buffered data

The attribute, if configured for alarms, detects the alarm condition and sets an associated alarm condition

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

attribute according to the base attribute being monitored. If the base attribute is configured with buffered data enabled, the condition attribute will also be enabled with buffered data support.

At run time, the condition attribute's buffer property will contain a VTQ buffer of true or false values representing alarm conditions corresponding to the base attribute buffer contents.

Only the most current alarm condition can be acknowledged.

For more information, see Alarms and events.

The following table describes buffer-enabled behavior for alarm functions.

Alarm function	Behavior
Detection	With buffered data enabled for the condition attribute, the alarm condition change is detected using the buffer property of the condition attributes. Multiple alarm messages can be generated based on the buffer contents. The alarm is time-stamped with the time the alarm was recorded at the field device, not the time at which the values were received and processed within the object. A set of alarm messages corresponding to the alarm state transitions are sent to its Notification Distributor. The TimeAlarmOff attribute’s value is set to the time of the last off transition, and TimeAlarmOn to the last on transition. The InAlarm value, time and quality properties are set to the last alarm transition’s state, time and quality.
Notification	When the Notification Distributor receives a set of alarm messages, it updates the state of the record for the corresponding alarm. The set of alarm messages are forwarded to all registered alarm clients. The Notification Distributor does not send a throttling signal as a result of receiving multiple alarm messages. When the Notification Distributor receives a set of event messages from ApplicationObjects as the result of processing an attribute’s buffer, the set of event messages are forwarded to all registered event clients.
Throttling	Alarm throttling settings are ignored for alarm messages generated based on buffered data. Alarms will be generated for all alarm conditions in the buffers. If a potential throttling point is reached, the current version of Application Server provides a functionality for all alarms to be processed without system overload.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Alarm function	Behavior
Acknowledgement	If an alarm is associated with buffered data, acknowledging an alarm only acknowledges the last alarm, and not all alarms contained in the buffer.
Disabled Alarms	If an alarm is disabled for a buffered attribute, the alarm's Buffer property is not processed. Alarm buffers for disabled alarms are cleared. If an alarm is silenced for a buffered attribute, the alarm's Buffer property is processed. Alarms will be generated but will not be displayed by the alarm client.
Outages	Buffered alarms and events can be lost when there is an outage because the Notification Distributor will only keep and re-send the last on-alarm message and the last off-alarm message.

Alarm and event types and buffered data

The following table describes buffered data-enabled behavior of alarm and event types.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Alarm/event type	Behavior
Data Change Event	Application (data change) Events have the following fields: • Event type – data change (historized) • Timestamp – date/time of event (historized) When Log Event is checked for the buffered data-enabled attribute, a data change event will be reported for every VTQ in the attribute's buffer property.
Limit Alarm	The limit alarm can provide a buffer of alarms on an attribute with buffered data enabled. If the base attribute has an associated Buffer property, then the limit alarm calculation is performed for each VTQ in the buffer. This results in a buffer of Boolean alarm conditions with associated time of the VTQ matching the times of the VTQ in input buffer, which is stored in the respective condition attribute's buffer property. The order of the condition buffer matches the order of the input buffer. The respective condition value and time reflects the last VTQ in the input buffer. **Value Deadband:** If value deadband is configured for limit alarms, then the alarm condition is calculated using VTQs in the input buffer. All VTQs in the buffer are processed. Value deadband is used during execution. **Time Deadband:** If time deadband is configured for limit alarms, when the alarm condition is met, the duration of the condition is calculated using the VTQs in the input buffer. If the last condition change remains undecided until the end of processing the current buffer, then a late decision will be made when the next input buffer is processed. If no more input buffer is received, the last alarm condition change is reported when the time deadband has elapsed as engine time. The time deadband is applied only to the On Rise event of the alarm state. It is not applied when the alarm state falls outside the limit condition.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Alarm/event type	Behavior
State Alarm	The state alarm can provide a buffer of alarms on a Boolean attribute when buffered data is enabled. If the base attribute has an associated Buffer property, then the state alarm calculation is performed for each VTQ in the input buffer. This results in a buffer of Boolean alarm conditions with associated time of the VTQ matching the times of the VTQ in input buffer, which is stored in the respective Condition attribute's buffer property. The order of the condition buffer matches the order of the input buffer. The respective condition value and time reflects the last VTQ in the input buffer.
Rate of Change Alarm	If the input has an associated Buffer property, then the Rate of Change (ROC) calculation is performed for each VTQ in the input buffer. Interval is calculated using the difference of two successive VTQs. These calculations result in an array of Boolean alarm conditions with associated timestamps which are stored in the buffer property of the respective condition alarm attribute. The condition value reflects the last VTQ in the respective condition buffer.
Deviation Alarm	If buffered data is enabled for a deviation alarm's attributes, buffered data is also enabled on the configured condition attributes Dev.Minor.Condition or Dev.Major.Condition or both. When buffered data is enabled on a deviation alarm's attributes and the AppEngine goes Off Scan, the buffers for the configured condition attributes, Dev.Minor.Condition or Dev.Major.Condition or both, are cleared of all VTQs. If the input has an associated Buffer property, then the deviation calculation is performed for each VTQ in the input buffer. These calculations result in an array of Boolean alarm conditions and associated timestamps which are stored in the buffer property of the respective condition alarm attribute. The condition value reflects the last VTQ in the respective Condition buffer.

Work with references

This section describes the concept of references and how to use reference strings in creating your Application

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Server application.

Message exchange and attributes

References allow identification and communication between objects in the System Platform environment. Every object, every attribute, and every property can be uniquely referenced. Those references are communicated over the messaging system. The Message Exchange is the object-to-object communications protocol used by Application Server, OMI, InTouch, and other System Platform components.

All object attributes have properties, such as Value and Quality. Any data read or written to or from these attributes over Message Exchange is tracked. If an operation cannot be performed, the requesting client is notified.

Message Exchange provides the following features and information:

- Guaranteed response
- Name signatures
- Status and data quality
- Message order preservation within a priority system
- AppEngine-to-AppEngine buffering
- Publish-subscribe heartbeats

Reference strings

Reference strings refer to an object or to data within an object's attributes. A reference string consists of an object's reference string plus an attribute's reference string.

```
object Reference + Attribute Reference
```

A reference string is the object name plus the attribute name: `ObjectName.AttributeName`.

In a multi-galaxy environment, a reference string to a remote galaxy is the galaxy name plus the object name plus the attribute name:

```
GalaxyName:ObjectName.AttributeName.
```

Use references and reference strings in a multi-galaxy environment as you would in a single-galaxy environment, with the exception of the galaxy name prefix, <GalaxyName>, followed by a colon as in the preceding example. For more information about using references and reference strings in a multi-galaxy environment, see Access multiple galaxies.

In `TIC101.PV`, `TIC101` is the object reference and `PV` is the attribute reference. The `AttributeName` can be omitted in a reference string, `PV` being assumed in such cases.

Note: Some objects have a `PV` attribute, while others do not.

Reference strings are concatenated substrings, each no more than 32 characters separated by periods. A substring cannot contain a period. Mathematical operator characters are not allowed. At least one character in each substring must be non-numeric.

Avoid assigning objects and attributes names such that the same reference string can refer to two different things. For example, you have two objects named A1 and B2, and inside A1 you create an attribute named B2 with a data type of Float. A1.B2 refers to the attribute Float named B2. If you then assign object B2 so that A1 is a container of object B2, the reference A1.B2 could refer either to the object B2 or the attribute B2 Float.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Important: The Galaxy resolves reference strings. If the GR is not available, resolution is done on a peer-to-peer level. After initial resolution, an object is provided an alias that handles references to its location across your network. If an object is relocated or renamed, the reference string resolution is repeated and a new alias provided.

Relative references

References that go up the hierarchy to parent objects are called relative references. For more information about hierarchy, see ApplicationObject containment.

Relative references, such as `Me`, are valid reference strings. A valid reference string must always contain at least a relative reference or one substring.

The following are valid relative references that refer to the current object:

- Me
- MyContainer
- MyArea
- MyPlatform
- MyEngine
- MyHost

Relative references are especially useful in templates because absolute references typically do not apply or make sense.

When you use relative references, like MyContainer, you can refer to contained objects within that container. For example, a reference to `MyContainer.InletValve.PV` is equivalent to `Tank1.InletValve.PV` in the following hierarchy:

`Tank1`	Cannot reference at this level because this is not contained
`Inlet Valve (InletValve)`	Can reference at this level because this object is contained
`Outlet Valve (OutletValve)`	Can reference at this level because this object is contained

Property references

Certain property names are reserved by Application Server. However, you are not blocked from using a string that is a reserved property name,. The `PROPERTY` keyword must be part of the string, for example, `PROPERTY(propertyName)`. In all other cases, the case insensitive `PROPERTY` keyword is not required.

The Value property is assumed if no property reference is specified.

The following are property references:

- .Name
- .Value
- .Type
- .Quality

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

- .Time

syntax:

```
obj.int.PROPERTY(quality)
```

where:

obj = object specifier

int = attribute

PROPERTY = keyword

(quality) = property specifier

For example, you can address the time of an attribute in a scan group for a DIObject from within an InTouch application, as follows:

```
Galaxy:"<DIObject>.<scangroup>.<attribute>.Property(Time)"
```

This is the same as if you used .Time:

```
Galaxy:"<DIObject>.<scangroup>.Attribute(<attribute>).Time"
```

You can directly address an item without having an attribute in the scan group. For this example, the item is MB1:

```
Galaxy:"<DIObject>.<scangroup>.MB1.Property(Time)"
```

and

```
Galaxy:"<DIObject>.<scangroup>.Attribute(MB1).Time"
```

For objects with a default scan group, you must refer to the .Time, .Value, and .Quality properties using the .Property(time), .Property(value), and .Property(quality) notation.

The following is an example of the correct use of property:

```
   LogMessage(ATTRIBUTE("MyEngine.tagname.PROPERTY(securityclassification)"))
```

The following is an example of the incorrect use of property:

```
   Logmessage(abtcpplc5_001.fast.changingpoint.property(quality));
```

Handle time zones with the time property

If you need to share time stamp values across different time zones (Platforms), use the Time data type in every time zone location. However, if you need to share it as string, remember that when converting the Time data type to a string (for example, in a script), it is automatically converted to local time, so you lose the ability to adjust it in a different time zone.

For example, to convert the Time property to a string GMT:

```
Dim localDateTime As System.DateTime;
localDateTime = System.DateTime.Parse( obj.attr.Time );
obj.udStringGMTfromLocalTime= localDateTime.ToUniversalTime().ToString();
```

To convert the string GMT to a string of local time:

```
Dim univDateTime As System.DateTime;
univDateTime = System.DateTime.Parse( obj.udStringGMTfromLocalTime );
Obj.udStringLocalTimeFromGMT = univDateTime.ToLocalTime().ToString();
```

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Preserve time stamps from the publishing source

In the following cases, if you want to pass only the time stamp, the subscriber gets the time stamp as converted to the local time zone of the publisher and not the time zone of the data source.

Examples of configurations that do not preserve the original time zone are as follows.

In this configuration, GalaxyB:Object1.TimeAttr shows the time adjusted to the local time zone of the GalaxyA Gateway Communication Driver and not the time zone of the PLC:

PLC.Item <= GalaxyA Object1.IntAttr.Time <= GatewayCommDriver <= GalaxyB OPCClient <= Object1.TimeAttr

In this configuration, GalaxyB:Object1.TimeAttr shows the time adjusted to the local time zone of the InTouch application and not the time zone of the PLC:

PLC.Item <= GalaxyA Object1.IntAttr.Time <= InTouch App I/O Message Tag <= GalaxyB InTouchProxy <= Object1.TimeAttr

To avoid these problems, subscribe to the GalaxyA:Object1.IntAttr value property. This way, both the value and time stamp propagate to GalaxyB:Object1.IntAttr. You can then use the GalaxyB:Object1.IntAttr.Time. For example:

PLC.Item <= GalaxyA Object1.IntAttr <= GatewayCommDriver <= GalaxyB OPCClient <= Object1.IntAttr

PLC.Item <= GalaxyA Object1.IntAttr <= InTouch App I/O Integer Tag <= GalaxyB InTouchProxy <= Object1.IntAttr

In this configuration, the time property propagates from InTouch to Object.IntAttr.Time:

PLC.Item <= InTouch I/O Integer Tag <= Galaxy InTouchProxy <= Object.IntAttr

Arrays

A reference string can also refer to the Value property of an array attribute with an optional Array Element Reference that includes up to one dimension:

- [i] – individual element
- [] – entire array

The letter `i` represents an integer constant.

Format reference strings

These symbols apply to the reference strings that follow:

This...	means...
::=	can be replaced by
|	or
[]	contents optional
{}	contents can be left out, used one time or repeated

Quotation marks are not allowed in tag names, feature names, or attribute names.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Literals

Items outside of angle brackets "<>" are literals. For example:

- reference_string ::= <Automation_object_reference><attribute_reference> | <tag_name>
- Automation_object_reference ::= <absolute_reference>|<relative_reference>
- absolute_reference ::= <tag_name>{.<contained_name>}
- tag_name ::= <identifier>
- contained_name ::= <identifier>
- relative_reference ::= <relative_name> | <relative_contained_reference>
- relative_contained_reference ::= MyContainer.<contained_name> | MyArea.<contained_name>
- relative_name ::= Me | MyContainer | MyArea | MyHost | MyEngine | MyPlatform
- attribute_reference ::= <value_ref>|<property_ref>
- whole_attribute_ref ::= [.<feature>][.<attribute>] | [.<feature>][.ATTRIBUTE(attribute)]
- value_ref ::= <whole_attribute_ref>[<array_index>]
- array_index ::= <open_bracket> {<index>} <close_bracket> [<open_bracket><index><close_bracket>][<open_bracket><index><close_bracket>]
- property_ref ::= <whole_attribute_ref>.<property>
- property ::= Value|Type|Quality|BitField|Dimension1|SecurityClassification|Locked|Category | propertyref
- propertyref ::= PROPERTY(Value|Type|Quality|BitField|Dimension1|SecurityClassification|Locked|Category)
- BitField ::= .00, .01, .02, …, .31 (valid ONLY for attributes of type MxInteger; otherwise Configuration error occurs at run time)
- attribute ::= <static_attribute>|<dynamic_attribute>
- static_attribute ::= [<static_attribute>.]<identifier>
- <dynamic_attribute> ::= <any_char_but>{<any_char_but>}
- feature ::= [<feature>.]<identifier>
- identifier ::= <valid_char>{<valid_char>}
- valid_char ::= <letter>|<digit>|<special_character>
- letter ::= any letter in alphabet of any language
- digit ::= any numerical character
- special_character ::= any graphics char, except the following:
  . + - * / \ = ( ) ` ~ ! % ^ & @ [ ] { } | : ; ’ , < > ? " whitespace
- whitespace ::= CR, LF, Tab, Space, FF, as returned by iswspace()
- any_char_but ::= any character except whitespace
- open_bracket := [
- close_bracket :=]
- Galaxy_identifier ::= <letter> | <digit>

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Notes

- <tag_name> is an object's unique name.
- <contained_name> is an object's optional contained name. It can be specified in a reference when an object is referred to as a contained child of another object.
- <index> is –1 or a positive integer from 1 to 32767.
- <identifier> is limited to a maximum of 32 characters.
- An <attribute> name or <feature> name can contain several <identifier> parts. The length of each <identifier> part can be up to 32 characters. Each <identifier> part is separated by a period. The maximum total length of the <attribute> name is 329. This name length applies to both static and dynamic attribute names. The maximum total length of the <feature> name is 329.
- <relative_name> and <property> replacements are case insensitive, including `PROPERTY()`.
- If no attribute reference is specified, `.PV` is assumed. If PV is an attribute of type array, the resulting reference is invalid. For arrays, the `.PV[]` part must be explicitly supplied.

  The exception to this rule is a reference that is preceded with an @ sign. This reference refers to the object itself and not any particular attribute or property. Currently, this reference string format is used only in the **Execution Order** group on the **Object Information** page of the Object Editor.
- Do not use Property Names or InTouch Pseudo-Property Names for the names of features or attributes when enhancing an object's functionality on the **Scripts**, **Attributes** and **Field Attributes** pages.

  Property Names include: Locked, Category, HasruntimeSetHandler, Name, Type, Quality, Dimension1, Value, SecurityClassification, 00, 01, 02, 03, 04, 05, 06, 07, 08, 09, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30 and 31.

  InTouch Pseudo-Property Names include: #VString, #VString1, #VString2, #VString3, #VString4, #EnumOrdinal, #ReadSts, #WriteSts and #QString. For more information on InTouch Pseudo-Property Names, see the *InTouch® HMI Data Management Guide*.
- If the object is a DIObject, use `obj.Attribute(attribute)[x]` rather than `obj.Attribute[x]` to avoid having to resolve the reference at the GR. Resolving an object reference at the GR can negatively impact performance.

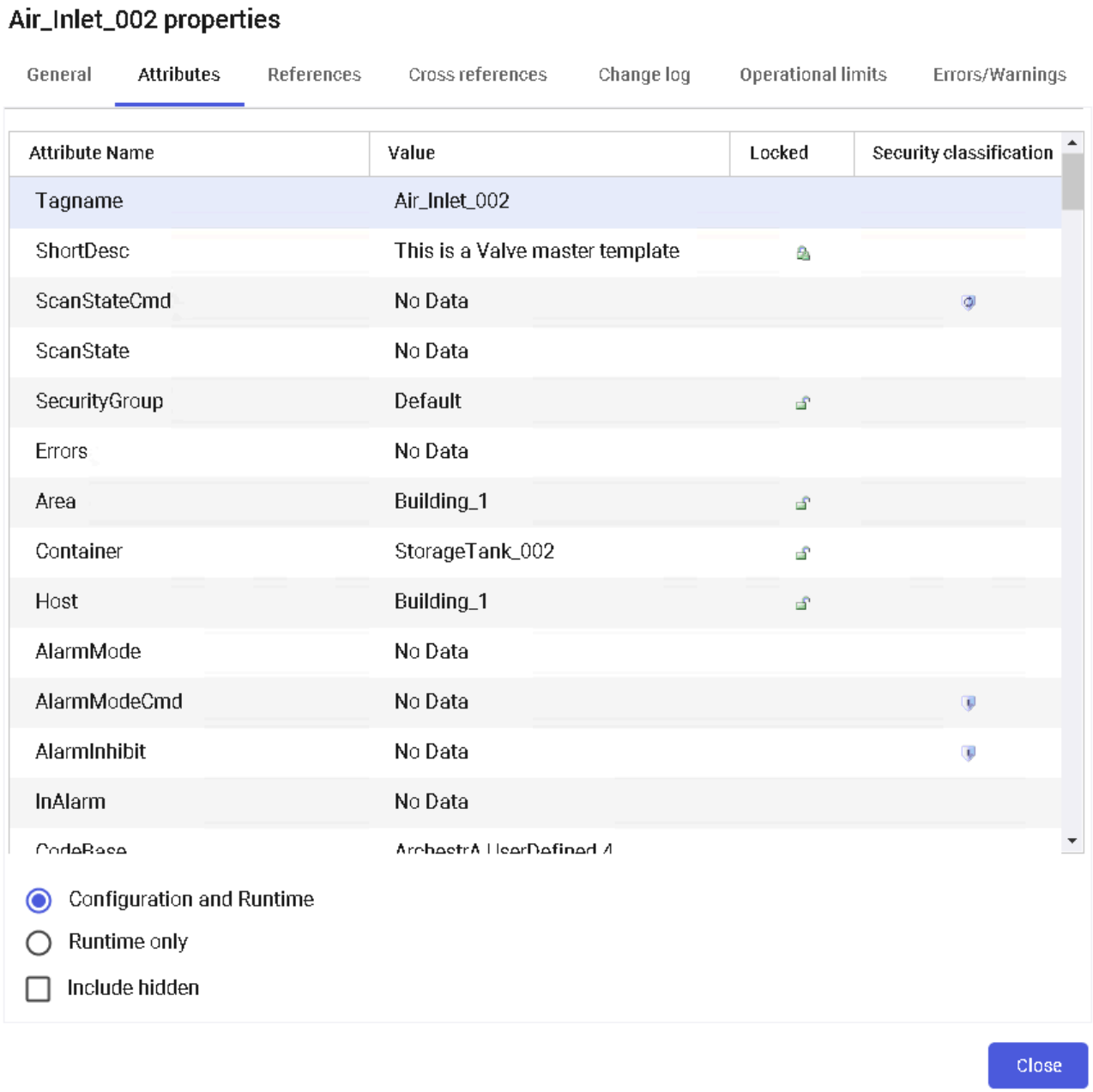
View object attributes in the IDE

Within the System Platform IDE, you can view the attributes in an object. This lets you see what attributes are available.

To view the attributes in a selected object

1. Select an object.
2. On the **Home** ribbon, in the **Edit** area, select **Properties**.
3. To see the references for the selected object, click the **Attributes** tab.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.



This page shows the selected attributes for the object, the current attribute values, and locked and security status. The information shown depends on whether **Configuration and Runtime** or **Runtime only** is selected.:

AttributeName

The name of the attribute.

Value

Shows the value of the attribute.

Locked/Unlocked

Shows if the attribute is locked or unlocked

Security

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

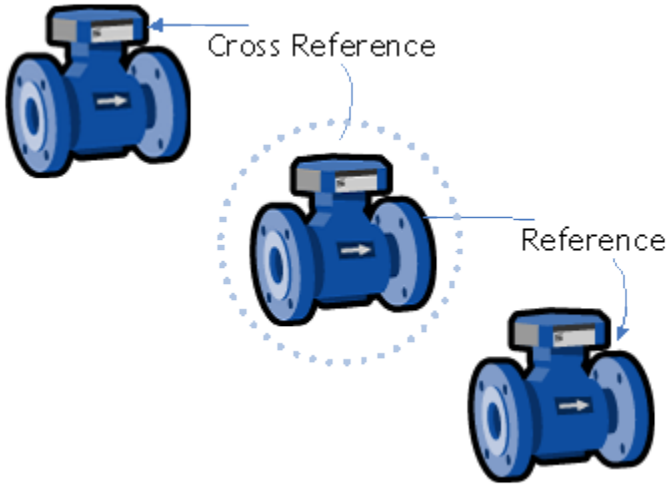
Shows the current security setting, if any.

4. To filter the view, select one or more of the following:
   - **Configuration and Runtime:** Select to show both configuration and run-time attributes for the selected object.
   - **Runtime only:** Select to show only run-time attributes for the selected object.
   - **Include hidden:** Select to show hidden attributes for the selected object.
5. When you are done, click **Close**.

View references and cross references

Objects have references and cross references.

- References are the objects the selected object is looking for.
- Cross-references are the objects looking for the selected object.



You can view references and cross-references for a selected object.

Some attributes are dynamic attributes. These are attributes that get created during run time and exist only in run-time. A device integration (DI) reference to a hardware register is a good example.

The attribute referencing a hardware register does not exist at configuration time by default. DI instances create the attribute dynamically during run time if the hardware register exists in the target device.

Note: References and cross-references shown in the **Properties** dialog box only refer to interobject communications. Area associations, containment, or host assignments are not shown.

To view references and cross-references

1. Select an object.
2. On the **Home** ribbon, in the **Edit** area, select **Properties**.
3. To see the references for the selected object, click the **References** tab. You see:

   Source Attribute

   The attribute in the selected object referencing an attribute in another object in the application Galaxy.

   Attribute Reference

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

The reference string within the **Source Attribute**. This is either an absolute reference or a relative reference.

Target Attribute

The absolute reference of the **Attribute Reference**. If this is a reference to a dynamic attribute, the **Target Attribute** only lists the Tagname name of the instance.

4. To see the cross references for the selected object, click the **Cross References** tab. You see:

   Target Attribute

   The absolute reference of the **Attribute Reference**. If this is a reference to a dynamic attribute, the **Target Attribute** only lists the Tagname of the instance.

   Attribute Reference

   The reference string within the **Source Attribute**. This is an absolute reference or a relative reference.

   Source Attribute

   The attribute in the selected object that is referencing an attribute in another object in the application Galaxy.

5. When you are done, click **Close**.

Find objects

Your Galaxy can get very large and can include many objects. It can become difficult to find a specific object. You can search for templates or instances, and you can search by part or all of a tag name or hierarchical name. You can limit a search by platform and engine. Other search options are:

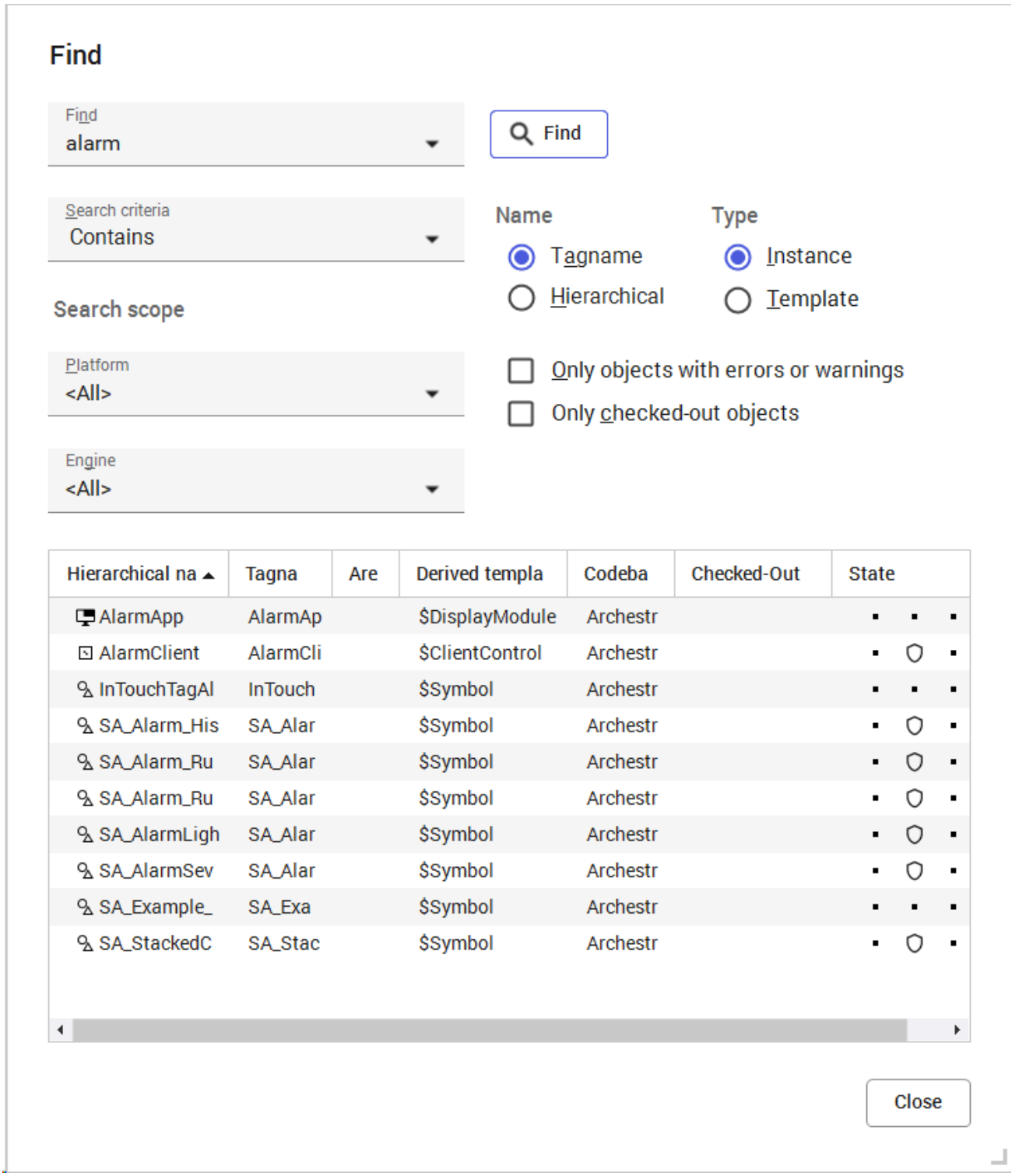
- Find objects with errors or warnings
- Find checked-out objects

To search for objects

1. On the **Home** ribbon, in the **Find** area, select **Find**.

   The **Find** dialog box appears.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.



2. Do some or all of the following:
   - In the **Find** box, type some or all of the name of the object.
   - In the **Name** area, select either **Tagname** or **Hierarchical name** as the type of name you entered in the **Find** box.
   - In the **Type** area, specify if you are looking for an **Instance** or a **Template**. If you select **Template**, the **Name** and **Search scope** groups are unavailable.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

- In the **Search criteria** field, select how to search for the name in the **Find what** box. The options are: **Contains, Exact match, Starts with** or **Ends with**.
- Search for objects in an error or warning state by selecting **Only objects with errors or warnings**.
- Search for objects that are checked out by selecting **Only Checked-out objects**.
- Limit the search scope by selecting from the **Search scope** lists.

3. When you are done specifying the search criteria, click the **Find** button. The search results appear in the bottom pane. The results pane includes the following information about each each object:
   - **Hierarchical name**
   - **Tagname**
   - **Area**, if any, that contains the object
   - **Derived template name** (which template the object is derived from)
   - **Codebase** name
   - Whether or not the object is **checked out**
   - **State** of the object Validation status, Editing status, Deployment status). See Application object icons for details.
4. Double-click an object in the results pane. The object is located and selected for you in the **Application views** area.

   If you double-click a Backup AppEngine, the IDE opens the **Deployment** view and the object is selected there. See Working with AppEngine Redundancy for more information.

Use galaxy references in InTouch

You can use Galaxy references in InTouch. Use the InTouch Tag Browser in unlimited selection mode to browse and include references from an Application Server Galaxy in InTouch applications you are developing.

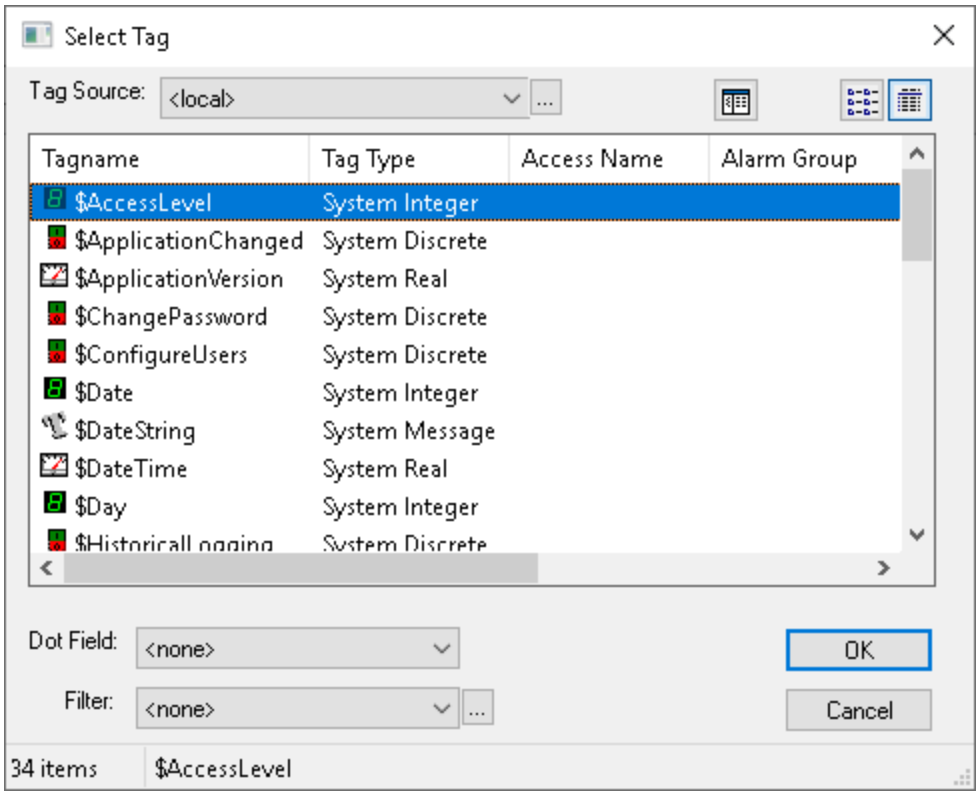
You also can use Galaxy references in InTouch in a multi-galaxy environment by using both the Galaxy: access name and the remote GalaxyName as part of the item name. For more information, see Use InTouch with multiple galaxies.

Note: You must install the Bootstrap and IDE on the InTouch node to create the TagSource to browse Galaxy objects.

The following lists the primary ways you can open the Tag Browser in InTouch to see unlimited selection mode:

- Double-click an animation link tagname or expression input box.
- Double-click an ActiveX or wizard tagname or expression input box.
- Double-click an empty area in any InTouch QuickScript window.
- In the InTouch QuickScript editor, select **Tagname** on the **Insert** menu.
- Press the **Alt+N** keys in the InTouch QuickScript editor.
- Double-click a blank **New Name** box in the **Substitute Tagnames** dialog box.
- Double-click the **Tagname** input box in the SQL Access **Bind List Configuration** dialog box.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.



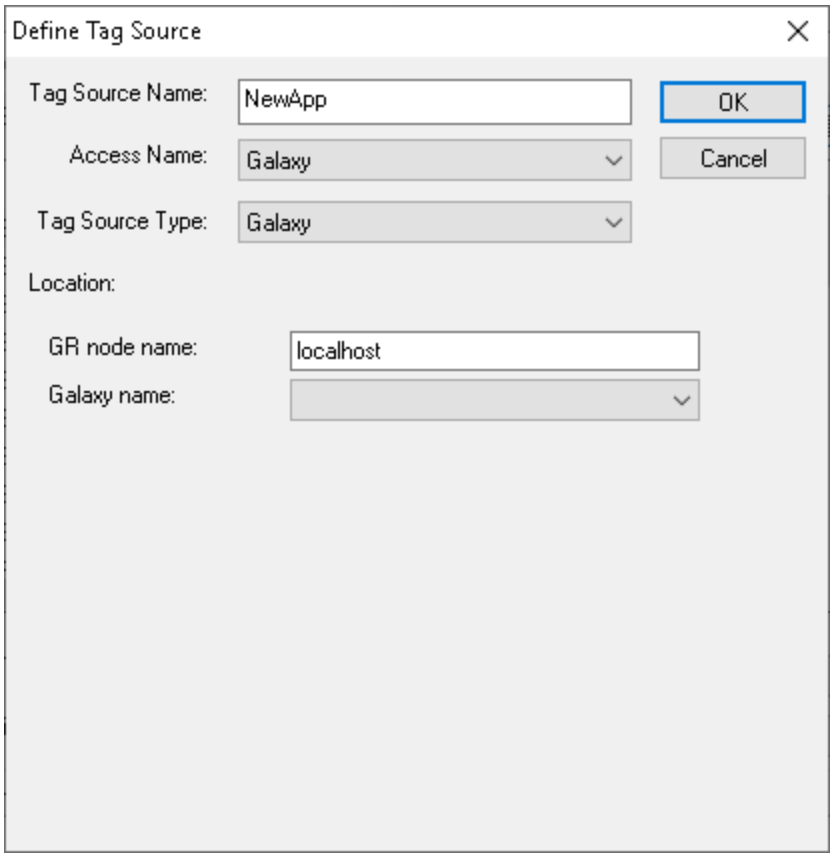
For complete information about using the InTouch Tag Browser, see the InTouch documentation.

Before you can browse Galaxy references, you must define a new tag source.

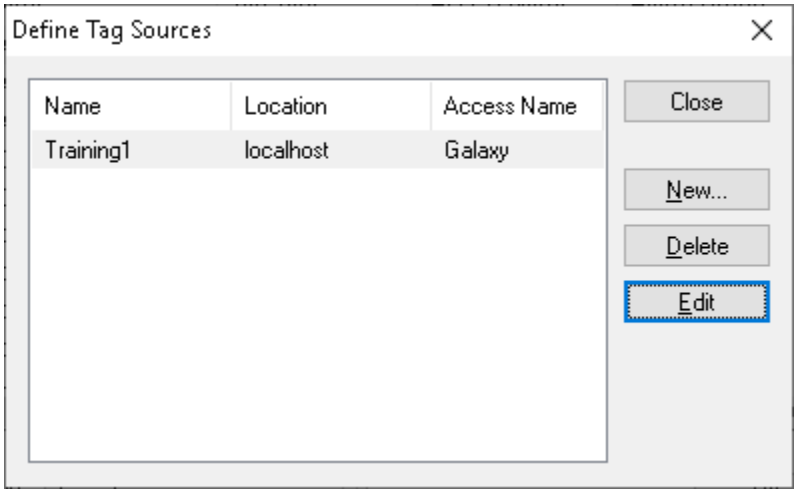
To define a new tag source

1. In the InTouch HMI **Tag Browser**, click the **Browse** button to the right of the **Tag Source** list. The **Define Tag Sources** dialog box appears.
2. Click **New** to open the **Define Tag Source** dialog box.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.



3. Type a **Tag Source Name**. This name appears in the **Tag Source** box of the Tag Browser. For example: Training1.
4. Select Galaxy from the list for **Access Name** and **Tag Source Type**.
5. In the **GR node name** box, type the Host Name of the computer on which the Galaxy is located. If the Galaxy is located on the same computer, use `localhost`.
6. In the **Galaxy Name** list, select the name of your Galaxy. For example: Training. Select **OK**. Assuming the examples given in steps 3 and 5, the **Define Tag Sources** dialog box appears as follows.



7. Click **Close**. The Tag Browser appears. Now you can browse the TagNames.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

To browse attribute references in a Galaxy

1. Open the **InTouch Tag Browser** in unlimited selection mode.
2. In the **Tag Source** box, select the tag source you created in the previous steps (Training1 in the example). The **Attribute Browser** appears.
3. Select the object and attribute you want to reference in your InTouch application and click **OK**.

Note: The next time you open the Tag Browser, the **Attribute** Browser automatically opens. To change that, exit the **Attribute** Browser without selecting anything by clicking the blue arrow at top right. The Tag Browser appears and it defaults to the InTouch Tagname Dictionary.

For more information about using the Attribute Browser, see Use the galaxy browser to reference objects .

Telemetry server data references

The Telemetry Server interfaces with Application Server to link data from widely dispersed equipment, using DNP3, IEC-60870, or Modbus protocols. This section describes how to create dynamic and static data references to the Telemetry Server, and is organized as follows:

- Telemetry server dynamic tags
- DNP3 data references
- IEC 60870-5 dynamic data references
- Modbus data references
- Telemetry server static data references

Note: The IEC-60870 protocol is not supported in AVEVA Telemetry Server version 1.0.

Dynamic data references to the Telemetry Server use the following syntax:

Telem_Server_<ScopeName>:%<OutstationFullName>;<PointType>;<SyntaxSpecificItems>;[<PointName>]

- The <space> character is valid in Telemetry Server references.
- <ScopeName> is not case sensitive. However, the rest of the reference string is case-sensitive.
- The percent sign (%) that precedes the OutstationFullName indicates that the reference is dynamic. Static data references do not include the percent sign. See Telemetry server static data references for more information.
- If [<PointName>] is specified, it must be enclosed in brackets.

Note that **Telem_Server_** is always prepended to the ScopeName (computer name) at the beginning of the reference string. Dynamic tags automatically populate some of the required fields in the Telemetry Server dashboard. This helps to reduce the amount of duplicated configuration data needed for each point. For information about creating dynamic tags, see Telemetry server dynamic tags.

Using attributes on UserDefined objects (UDOs) is a recommended best practice to create the I/O interface to the Telemetry Server. Once you create the attribute and enable I/O, create the I/O reference using the applicable protocol and syntax. See Enable I/O for an attribute for more information about enabling I/O.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Point Name Format

When creating a reference and you include the optional parameter, "Point Name," the Point Name MUST be enclosed in [brackets] as the following example shows:

Telem_Server_Node-12:%Group.DNP3_Outstation;AI;100;[Point_Name]

Invalid Characters

The following are invalid characters that cannot be used within Telemetry Server OutStationFullNames or PointNames:
< > # % \ " ; : ? . @ & = + $ { } | ^ [ ]

The % . [ ]and : characters are used in creating Telemetry Server references.

- The "%" character designates the reference as a dynamic tag.
- The "." character is used as a separator between the GroupName and OutstationName within the OutStationFullName, and can be used as part of a Point Name when creating a point in a different group than the Outstation. The syntax for this is <GroupName>.<PointName>.
- The ":" character is used to separate the ScopeName from the Telemetry Server tag information.
- The square bracket characters "[ ]" are used to enclose the Point Name, when the Point Name is specified and cannot be used anywhere else in the reference.

Telemetry server dynamic tags

Dynamic tags (data references) automatically populate some of the required fields in the Telemetry Server dashboard. This helps to reduce the amount of duplication when creating tags for the Telemetry Server. A dynamic tag takes the information entered as an I/O reference in Application Server, and creates or renames a point on the Telemetry Server.

Include a percent sign (%) in the reference string to designate a dynamic tags. The % character is inserted after the Scope Name and separator character ("Telem_Server_<ScopeName>:") in the reference string.

Note that **Telem_Server_** is always prepended to the Scope Name (computer name) at the beginning of the reference string.

Dynamic data references vs. static data references

Both dynamic data references (tags) and static data references can be used to access the Telemetry Server.

Dynamic data references are used to create the Telemetry Server points/tags remotely from Application Server. Use the reference syntax that corresponds with the protocol you are using. See Telemetry server dynamic tags for more information.

Static data references are used when the points/tags (discrete, analog, strings) already exist in the Telemetry Server. As such, the reference syntax is specific to the pre-existing tag. Use a static data reference Syntax to reference the properties of any object in the database (outstations, channels, points, etc) that already exist in Telemetry Server, or to invoke methods on these objects. See Telemetry server static data references for more information.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Dynamic tag prerequisites

Before a dynamic tag entered in Application Server can function and automatically create a point in the Telemetry Server, the following items must be defined through the Telemetry Server dashboard:

- Group
- Channel
- Outstation Set
- Outstation

Note: *Group* is a Telemetry Server-specific term that refers to a folder in the configuration hierarchy. An outstation object is contained within a group. In the Telemetry Server, the configuration hierarchy is the tree at the left side of the dashboard.

Naming rules

When entering data references for an I/O attribute in Application Server, the following naming rules are used to create or rename points in the Telemetry Server:

- If a point with the specified name and address does not exist in the specified group and outstation, a new point is created with specified name. If a point name is not specified in the reference, the default naming convention is used to generate the point name. Typically, this is <PointType>_<Address>.
- If the point name is specified in the data reference, the point name must be enclosed in square brackets [ ].
- If a point with the specified name and address already exists in the specified group and outstation, the existing point is used.
- If a point with the specified address but a different name is found in the specified group and outstation, the reference generates a configuration error. The point with specified address is not changed and a new point is not created. See Configuration errors for more information.
- If a point with the specified name exists in the specified group, but is attached to a different outstation, the reference generates a configuration error. The named point is not changed and a new point is not created. See Configuration errors for more information.
- If a point with the specified name exists in the specified group and outstation but has a different point address, the reference generates a configuration error. The named point is not changed and a new point is not created. See Configuration errors for more information.
- If any other database object with the specified name exists in the specified group, the reference generates a configuration error. A new point is not created. See Configuration errors for more information.
- If the point name is invalid, for example, if it contains invalid characters, is too long, or has some other error, the reference generates a configuration error and a new point is not created. See Configuration errors for more information.

Configuration errors

To resolve configuration errors, refer to the *Telemetry Server User Guide* for detailed information. The basic way to resolve an error is to make one or more of the following changes, as applicable:

- Check that the <PointName> in the dynamic tag reference is valid to allow the point to be created in the

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Telemetry Server database.

- Check that the <PointType> and *address* in the dynamic tag reference are valid for the protocol and correct for the particular outstation. For some point types, such as Modbus digital points, you must additional items, such as bit offset and bit count.
  - In DNP3, address refers to PointNumber.
  - In IEC 60870, address refers to IOA.
  - In Modbus, address refers to PointAddress
- Check that the <OutstationFullName> in the dynamic tag reference exactly matches the name of the outstation in the Telemetry Server database and that this is the correct outstation. The name is case sensitive.
- Rename, or delete, the object that has the same name in the group, or move the object to a different group.

DNP3 data references

A DNP3 point is defined by its outstation name, point type, point number, and optionally, a point name. When adding points to an outstation, use the following general guidelines:

- All references begin with "Telem_Server_" prepended to the Scope Name.
- Point numbers are unique within each point type. For example “Analog Input 1” (AI_1) and “Analog Output 1” (AO_1) are different points. In the case of binary output points, which contains three subtypes, the subtype does not provide differentiation. Thus, “Digital Output 1”, “Pulse (NULL) 1” and “Pulse Trip-Close 1” all refer to the same binary output point. For example “Digital Output 1”, “Pulse (NULL) 1” and “Pulse Trip-Close 1” all refer to the same binary output point.
- Point numbers must be unique within each data type. The range of valid point numbers may vary between outstations. Point numbers do not need to be contiguous.
- The point numbers (in an outstation) are configurable. Refer to the outstation configuration to determine which point numbers are valid for a particular outstation.
- The valid range of DNP3 point numbers is 0 to 65535. The Telemetry Server does not limit this range, however, values are device-specific and many devices will impose a much smaller number than 65535 for their upper limit.

Point types

A point in DNP3 can be defined as any of the following data types:

Point type	I/O type	Details
AI	Analog Input	DNP3 analog input and output points
AO	Analog Output	DNP3 analog input and output points
DI	Binary Input	DNP3 binary input and output points

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Point type	I/O type	Details
DO	Binary Output	DNP3 binary input and output points
DBI	Double Binary Input	DNP3 double-bit binary input points
RC	Counter	DNP3 counter input points
OS	String	DNP3 string input points
PN	Pulse (NULL)	DNP3 Pulse Output Points
PTC	Pulse (Trip/Close)	DNP3 Pulse Output Points

Reference syntax

When creating a Telemetry Server reference, adding a percent sign (%) before the full outstation name makes the reference a dynamic reference. The full outstation name includes the group name.

The general syntax for dynamic references is as follows :

Telem_Server_*<ScopeName>:%<OutstationFullName>;<PointType>;<PointNumber>;*[*<PointName>*]

To create the point in a different group, add the group name to the PointName with a "." separator, as follows:

Telem_Server_*<ScopeName>:%<OutstationFullName>;<PointType>;<PointNumber>;[<GroupName>.<PointName>*]

Note that all point types contain all the elements shown in the general syntax, but some point types contain additional parameters. For example, the syntax for Pulse data types is as follows:

Telem_Server_*<ScopeName>:%<OutstationFullName>;PTC;<PointNumber1>;<PointNumber2>;*[*<PointName>*]

When the [*<PointName>*] parameter is not specified and the point does not already exist, a default name is automatically assigned to the point. The name is in the format "*<PointType>_<PointNumber>*."

If you specify the PointName, enclose the name in brackets.

DNP3 analog input and output points

DNP3 analog point syntax

Telem_Server_<ScopeName>:%<OutstationFullName>;<PointType>;<PointNumber>{;ZSValue}{;FSValue}{;DBValue}{%}{;[P

where items enclosed in { curly brackets } are optional.

- **PointType:** AI or AO
- **PointNumber:** a decimal number from 0 to 65535 that maps to the "Point Number" field in the Telemetry Server dashboard. The maximum point number is device-dependent.
- **Optional fields:** ZSValue, FSValue, DBValue, and PointName. You can enter these fields in any order. If no value is entered, the default values for these fields are used.
  - **ZSValue:** This sets the zero scale limit of the tag in the Telemetry Server, and uses the format "ZS<value>", where <value> is a double data-type value and includes both positive and negative

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

numbers. If <value> is out of range, point creation fails.

- **FSValue:** This sets the full scale limit of the tag in the Telemetry Server. It uses the format "FS<value>", where <value> is a double data-type value and includes both positive and negative numbers. If the value given is out of the range, point creation fails.
- **DBValue:** This sets the Significant Change option (deadband) of the tag in the Telemetry Server. You can enter the deadband as an absolute or percentage value, as follows:
  - DB<value> sets the **Significant Change** field to **Absolute,** and sets the **Deadband** field to <value>.
  - DB<value>% sets the **Significant Change** field to **Percentage of Change**, and sets the **Deadband** field to <value>.
  - The <value> specified for both **Absolute** and **Percentage of Change** must be a positive double data-type value.
  - When **Percentage of Change** is used, the maximum value is 100.
- **PointName:** a descriptive name for the point, for example, [LiftStationPump_01]. If specified, the PointName must be enclosed in brackets.
  - If PointName is not specified, the point is assigned a default name that uses point type and number, in the format **PointType_PointNumber**, for example, AI_100.
  - If you want to create the new point in a different group than the Outstation group, include the group name: **GroupName.PointName.**

Analog reference examples

Without optional parameters:

Telem_Server_Node-12:%Group.DNP3_Outstation;AI;100

References analog input point 100, which is given the name **AI_100**

With the optional zero scale, full scale, and deadband parameters:

Telem_Server_Node-12:%Group.DNP3_Outstation;AI;100;ZS-10;FS250;DB25%

References the same analog input point as above, but defines the zero scale limit as -10, the full scale limit as 250, and the deadband as 25%

With the optional point name parameter that includes group name (include the group name when creating the point in a different group than the Outstation):

Telem_Server_Node-12:%Group.DNP3_Outstation;AI;100;[Group2.Point_Name]

References analog input point 100, which is named **Point_Name** and is created in Group2.

DNP3 binary input and output points

DNP3 binary point syntax

Telem_Server_<ScopeName>:%<OutstationFullName>;<PointType>;<PointNumber>{;[<PointName>]}

where items enclosed in { curly brackets } are optional.

- **PointType:** DI or DO

  Note: When referencing a binary output point using the DO point type, the Telemetry Server uses the "latch on/off" operation type.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

- **PointNumber:** a decimal number from 0 to 65535 that maps to the "Point Number" field in the Telemetry Server dashboard. The maximum point number is on the device-dependent.
- **PointName (optional):** a descriptive name for the point, for example, LiftStationPump_01. If specified, the PointName must be enclosed in brackets.
  - If PointName is not specified, the point is assigned a default name that uses point type and number, in the format **PointType_PointNumber**, for example, DI_5.
  - If you want to create the new point in a different group than the Outstation group, include the group name: **GroupName.PointName.**

Binary reference examples

Without optional PointName:

Telem_Server_Node-12:%Group.DNP3_Outstation;DI;5

References binary input point 5, which is given the name **DI_5**

With optional PointName:

Telem_Server_Node-12:%Group.DNP3_Outstation;DI;5;[Tag_Name]

References binary input point 5, which is named **Tag_Name**

DNP3 double-bit binary input points

DNP3 double-bit binary point syntax

Telem_Server_<ScopeName>:%<OutstationFullName>;<PointType>;<PointNumber>{;[<PointName>]}

where items enclosed in { curly brackets } are optional.

- **Point type:** DBI
- **Point number:** a decimal number from 0 to 65535 that maps to the "Point Number" field in the Telemetry Server dashboard. The maximum point number is device-dependent.
- **PointName (optional):** a descriptive name for the point, for example, LiftStationPump_01. If specified, the PointName must be enclosed in brackets.
  - If PointName is not specified, the point is assigned a default name that uses point type and number, in the format **PointType_PointNumber**, for example, DBI_5.
  - If you want to create the new point in a different group than the Outstation group, include the group name: **GroupName.PointName.**

Double-bit binary reference examples

Without optional PointName:

Telem_Server_Node-12:%Group.DNP3_Outstation;Group.DBI;5

References double-bit binary point 5, which is given the name **DBI_5**

With optional PointName:

Telem_Server_Node-12:%Group.DNP3_Outstation;DBI;5;[Tag_Name]

References double-bit binary point 5, which is named **Tag_Name**

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

DNP3 counter input points

DNP3 running counter point syntax

Telem_Server_<ScopeName>:%<GroupName.OutstationName>;<PointType>;<PointNumber>{;[<PointName>]}

- **PointType:** RC
- **PointNumber:** a decimal number from 0 to 65535 that maps to the "Point Number" field in the Telemetry Server dashboard. The maximum point number is device-dependent.
- **PointName (optional):** a descriptive name for the point, for example, LiftStationPump_01. If specified, the PointName must be enclosed in brackets.
  - If PointName is not specified, the point is assigned a default name that uses point type and number, in the format **PointType_PointNumber**, for example, RC_3.
  - If you want to create the new point in a different group than the Outstation group, include the group name: **GroupName.PointName.**

Counter reference examples

Without optional PointName:

Telem_Server_Node-12:%Group.DNP3_Outstation;RC;3

References running counter point 3, which is given the name **RC_3**

With optional PointName:

Telem_Server_Node-12:%Group.DNP3_Outstation;RC;3;[Tag_Name]

References running counter point 3, which is named **Tag_Name**

DNP3 string input points

DNP3 octet string point syntax

Telem_Server_<ScopeName>:%<OutstationFullName>;<PointType>;<PointNumber>{;[<PointName>]}

- **PointType:** OS
- **PointNumber:** a decimal number from 0 to 65535 that maps to the "Point Number" field in the Telemetry Server dashboard. The maximum point number is device-dependent.
- **PointName (optional):** a descriptive name for the point, for example, LiftStationPump_01. If specified, the PointName must be enclosed in brackets.
  - If PointName is not specified, the point is assigned a default name that uses point type and number, in the format **PointType_PointNumber**, for example, OS_6.
  - If you want to create the new point in a different group than the Outstation group, include the group name: **GroupName.PointName.**

String reference examples

Without optional PointName:

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Telem_Server_Node-12:%Group.DNP3_Outstation;OS;6

References octet string point 6, which is given the name **OS_6**

With optional PointName:

Telem_Server_Node-12:%Group.DNP3_Outstation;OS;6;[Tag_Name]

References octet string point 6, which is named **Tag_Name**

DNP3 Pulse Output Points

DNP3 pulse-null point syntax

Telem_Server_<ScopeName>:%<OutstationFullName>;<PointType>;<PointNumber>{;[<PointName>]}

- **PointType:** PN
- **PointNumber:** a decimal number from 0 to 65535 that maps to the "Point Number" field in the Telemetry Server dashboard. The maximum point number is device-dependent.
- **PointName (optional):** a descriptive name for the point, for example, LiftStationPump_01. If specified, the PointName must be enclosed in brackets.
  - If PointName is not specified, the point is assigned a default name that uses point type and number, in the format **PointType_PointNumber**, for example, PN_12.
  - If you want to create the new point in a different group than the Outstation group, include the group name: **GroupName.PointName.**

Pulse reference example

Without optional PointName:

Telem_Server_Node-12:%Group.DNP3_Outstation;PN;12

References binary output point 12 using "pulse-on" operator type and "NULL" trip-close code, which is given the name **PN_12**

With optional PointName:

Telem_Server_Node-12:%Group.DNP3_Outstation;PN;12;[Tag_Name]

References pulse (null) point 12 using "pulse-on" operator type and "NULL" trip-close code, which is named **Tag_Name**

DNP3 pulse (trip/close) output points

DNP3 trip-close pulse point syntax

Telem_Server_<ScopeName>:%<OutstationFullName>;<PointType>;<PointNumber1>;<PointNumber2>{;[<PointName>]}

- **PointType:** PTC
- **PointNumber:** a decimal number from 0 to 65535 that maps to the "Point Number" field in the Telemetry Server dashboard. The actual range of valid point numbers is dependent on the specific outstation.

  For the Trip-Close Pulse function, two point numbers are defined, one for each action (that is, one point

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

number for **trip** and the other for **close**). Generally, these two numbers are the same, however, some non-standard outstations require different point numbers.

- **PointName (optional):** a descriptive name for the point, for example, LiftStationPump_01. If specified, the PointName must be enclosed in brackets.
  - If PointName is not specified, the point is assigned a default name that uses point type and point numbers in the format **PointType_PointNumber1_PointNumber2**, for example, P_5_5. Note that both point numbers are used for automatic naming.
  - If you want to create the new point in a different group than the Outstation group, include the group name: **GroupName.PointName.**

Pulse (trip-close) reference example

Without optional PointName:

Telem_Server_Node-12:%Group.DNP3_Outstation;PTC;5;5

Reference binary output point 5 using "pulse-on" operator type and "trip-close" codes, which is given the name **PTC_5_5**

With optional PointName:

Telem_Server_Node-12:%Group.DNP3_Outstation;PTC;5;5;[Tag_Name]

References binary output point 5 using "pulse-on" operator type and "trip-close" codes, which is named **Tag_Name**

IEC 60870-5 dynamic data references

An IEC60870-5 point is defined by its outstation name, point type, Information Object Address (ioa), and optionally, a point name. When adding points to an outstation, use the following general guidelines:

- All references begin with "Telem_Server_" prepended to the Scope Name.
- Each point type (AI, AO, etc.) has its own unique set of point numbers. Note that AI_1 is different than AO_1.
- Information Object Addresses (IOAs) must be unique across the outstation.
- The valid range of IEC 60870 IOAs is 1 to 16,777,216. The Telemetry Server does not limit this range, however, values are device-specific and many devices will impose a much smaller number than 16,777,216 for their upper limit.

Point types

A point in IEC 60870-5 can be defined as any of the following data types:

Point type	I/O type	Details
AI	Analog Input	IEC60870 analog input and output points
AO	Analog Output	IEC60870 analog input and output points

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Point type	I/O type	Details
DI	Digital Input	IEC60870 digital input and output points
DO	Digital Output	IEC60870 digital input and output points

Reference syntax

The general syntax for references is as follows, but there are variations between analog and digital point types:

Telem_Server_*<ScopeName>:%<OutstationFullName>;<PointType>;<IOA>;[<PointName>]*

If you specify the PointName, enclose the name in brackets.

When the optional *<PointName>* parameter is not specified, a default name is automatically assigned to the point. The name is in the format "*<PointType>_<IOA>*."

To create the point in a different group, add the group name to the PointName with a "." separator, as follows:

Telem_Server_*<ScopeName>:%<OutstationFullName>;<PointType>;<IOA>;[<GroupName>.<PointName>]*

IEC60870 analog input and output points

IEC 60870 analog point syntax

Telem_Server_<ScopeName>:%<OutstationFullName>;<PointType>;<IOA>{;ZSValue}{;FSValue}{;DBValue}{%}{;[PointName

- **PointType:** AI or AO
- **IOA:** Information Object Address. This is a decimal number from 1 to 16,777,216, and maps to the "IOA" field in the Telemetry Server dashboard. The actual range of valid IOA numbers is dependent on the specific outstation.
- **Optional fields:** ZSValue, FSValue, DBValue, and PointName. You can enter these fields in any order. If no value is entered, the default values for these fields are used.
  - **ZSValue:** This sets the zero scale limit of the tag in the Telemetry Server, and uses the format "ZS<value>", where <value> is a double data-type value and includes both positive and negative numbers. If <value> is out of range, point creation fails.
  - **FSValue:** This sets the full scale limit of the tag in the Telemetry Server. It uses the format "FS<value>", where <value> is a double data-type value and includes both positive and negative numbers. If the value given is out of the range, point creation fails.
  - **DBValue:** This sets the Significant Change option (deadband) of the tag in the Telemetry Server. You can enter the deadband as an absolute or percentage value, as follows:
    - DB<value> sets the **Significant Change** field to **Absolute,** and sets the **Deadband** field to <value>.
    - DB<value>% sets the **Significant Change** field to **Percentage of Change**, and sets the **Deadband** field to <value>.
    - The <value> specified for both **Absolute** and **Percentage of Change** must be a positive double data-type value.
    - When **Percentage of Change** is used, the maximum value is 100.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

- **PointName:** a descriptive name for the point, for example, [LiftStationPump_01]. If specified, the PointName must be enclosed in brackets.
  - If PointName is not specified, the point is assigned a default name that uses point type and number, in the format **PointType_PointNumber**, for example, AI_100.
  - If you want to create the new point in a different group than the Outstation group, include the group name: **GroupName.PointName.**

Analog reference examples

Without optional parameters:

Telem_Server_Node12:%Group.IEC60870_Outstation;AI;100

References analog input point 100, which is given the name **AI_100**

With the optional zero scale, full scale, and deadband parameters:

Telem_Server_Node12:%Group.IEC60870_Outstation;AI;100;ZS-10;FS250;DB25%

References the same analog input point as above, but defines the zero scale limit as -10, the full scale limit as 250, and the deadband as 25%

With the optional point name parameter that includes group name (include the group name when creating the point in a different group than the Outstation):

Telem_Server_Node12:%Group.IEC60870_Outstation;AI;100;[Group2.Point_Name]

References analog input point 100, which is given the name **Point_Name** and is created in Group2.

IEC60870 digital input and output points

IEC 60870 digital point syntax

Telem_Server_<ScopeName>:%<OutstationFullName>;<PointType>;<IOA>;<BitCount>{;[<PointName>]}

- **PointType:** DI or DO
- **IOA:** Information Object Address. This is a decimal number from 1 to 16,777,216, and maps to **Information Object Address** in the Telemetry Server dashboard. The maximum IOA number is dependent on the outstation.
- **BitCount:** a decimal number between 1 and 3. This number maps to **Number of Bits** in the Telemetry Server dashboard.
- **PointName (optional):** a descriptive name for the point, for example, LiftStationPump_01.
  - If PointName is not specified, the point is assigned a default name that uses point type and number, in the format **PointType_PointNumber**, for example, DI_5.
  - If you want to create the new point in a different group than the Outstation group, include the group name: **GroupName.PointName.**

Digital reference examples

Without optional PointName:

Telem_Server_Node12:%Group.IEC60870_Outstation;DI;5;1

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

References 1-bit digital input point 5, which is given the name **DI_5**

With optional PointName:

Telem_Server_Node12:%Group.IEC60870_Outstation;DI;5;1;[Tag_Name]

References 1-bit digital input point 5, which is named **Tag_Name**

Modbus data references

A Modbus point is defined by its outstation name, point type, point address, and optionally, point name. When adding points to an outstation, use the following general guidelines:

- All references begin with "Telem_Server_" prepended to the Scope Name.
- Modbus point addresses have 5 or 6 digits, depending on the Point Address Range setting of the outstation. The default setting is 5 digits. The range of valid addresses vary by point type.

Point types

A point in Modbus can be defined as any of the following types:

Point type	I/O type	Details
AI	Analog Input	Modbus analog input and output points
AO	Analog Output	Modbus analog input and output points
DI	Digital Input	Modbus digital input and output points
DO	Digital Output	Modbus digital input and output points
STR	String	Modbus string points

Reference syntax

The general syntax for references is as follows, but there are variations between analog, digital and string point types:

Telem_Server_<*ScopeName*>%<*OutstationFullName*>;<*PointType*>;<*PointAddress*>;<*DataType*>;[<*PointName*>]

When the optional <*PointName*> parameter is not specified, a default name is automatically assigned to the point. The name is in the format "<*PointType*>_<*PointAddress*>."

If you specify the PointName, enclose the name in brackets.

To create the point in a different group, add the group name to the PointName with a "." separator, as follows:

Telem_Server_<*ScopeName*>%<*OutstationFullName*>;<*PointType*>;<*PointAddress*>;<*DataType*>;[<GroupName>.<*PointNa*

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Modbus analog input and output points

Modbus analog point syntax

Telem_Server_<ScopeName>:%<GroupName.OutstationName>;<PointType>;<PointAddress>;<DataType>{;ZSValue}{;FSVa

- **PointType:** AI or AO
- **PointAddress:** a 5 or 6 digit decimal number, depending on the Point Address Range setting of the outstation. The default setting is 5 digits. The range of valid addresses vary by point type and maps to the "Address" field in the Telemetry Server dashboard.
- **DataType:** one of the values listed in the following table, for example UINT16.

Data type	Definition
UINT16	Unsigned 16-bit integer
UINT32	Unsigned 32-bit integer
INT16	Signed 16-bit integer
INT32	Signed 32-bit integer
SINGLE	Single precision float
DOUBLE	Double precision float

- **Optional fields:** ZSValue, FSValue, DBValue, and PointName. You can enter these fields in any order. If no value is entered, the default values for these fields are used.
  - **ZSValue:** This sets the zero scale limit of the tag in the Telemetry Server, and uses the format "ZS<value>", where <value> is a double data-type value and includes both positive and negative numbers. If <value> is out of range, point creation fails.
  - **FSValue:** This sets the full scale limit of the tag in the Telemetry Server. It uses the format "FS<value>", where <value> is a double data-type value and includes both positive and negative numbers. If the value given is out of the range, point creation fails.
  - **DBValue:** This sets the Significant Change option (deadband) of the tag in the Telemetry Server. You can enter the deadband as an absolute or percentage value, as follows:
    - DB<value> sets the **Significant Change** field to **Absolute,** and sets the **Deadband** field to <value>.
    - DB<value>% sets the **Significant Change** field to **Percentage of Change**, and sets the **Deadband** field to <value>.
    - The <value> specified for both **Absolute** and **Percentage of Change** must be a positive double data-type value.
    - When **Percentage of Change** is used, the maximum value is 100.
  - **PointName:** a descriptive name for the point, for example, [LiftStationPump_01]. If specified, the PointName must be enclosed in brackets.
    - If PointName is not specified, the point is assigned a default name that uses point type and number, in the format **PointType_PointNumber**, for example, AI_100.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

- If you want to create the new point in a different group than the Outstation group, include the group name: **GroupName.PointName.**

Analog reference examples

Without optional parameters:

Telem_Server_Node-12:%Group.Modbus_RTU;AI;40001;UINT16

References the analog input point mapped to holding register 40001with UINT16 data type, which is given the name **AI_40001**

With the optional zero scale, full scale, and deadband parameters:

Telem_Server_Node-12:%Group.Modbus_RTU;AI;40001;UINT16;ZS-10;FS250;DB25%

References the same analog input point as above, but defines the zero scale limit as -10, the full scale limit as 250, and the deadband as 25%

With the optional point name parameter (include the group name when creating the point in a different group than the Outstation)::

Telem_Server_Node-12:%Group.Modbus_RTU;AI;40001;UINT16;[Group2.Point_Name]

References the analog input point mapped to holding register 0001 with UINT16 data type, which is named **Point_Name** and is created in Group2.

Modbus digital input and output points

Modbus digital point syntax

Telem_Server_<ScopeName>:%<OutstationFullName>;<PointType>;<PointAddress>;<BitOffset>;<BitCount>{;[<PointName

- **PointType:** DI or DO
- **PointAddress:** a 5 or 6 digit decimal number, depending on the Point Address Range setting of the outstation, which is set through the Telemetry Server dashboard. The default setting is 5 digits. The range of valid addresses vary by point type and maps to the "Address" field in the Telemetry Server dashboard.
- **BitOffset:** a decimal number between 0 and 15. This number maps to the "BitOffset" field in the Telemetry Server dashboard. The maximum offset varies and depends on the bit count.
- **BitCount:** a decimal number between 1 and 3. This number maps to the "BitCount" field in the Telemetry Server dashboard.
- **PointName (optional):** a descriptive name for the point, for example, LiftStationPump_01. If specified, the PointName must be enclosed in brackets. If point name is not specified, the point is assigned a default name that uses point type and point address, in the format **PointType_PointAddress**, for example, DI_30001.
  - If PointName is not specified, the point is assigned a default name that uses point type and number, in the format **PointType_PointAddress**, for example, DI_30001.
  - If you want to create the new point in a different group than the Outstation group, include the group name: **GroupName.PointName.**

Digital reference examples

Without optional PointName:

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Telem_Server_<*ScopeName*>:%Group.Modbus_RTU;DI;30001;0;1

References the 1-bit digital input point mapped to the least significant bit of input register 30001, which is given the name **DI_30001**

With optional PointName:

Telem_Server_<*ScopeName*>:%Group.Modbus_RTU;DI;30001;0;1;[Tag_Name]

References the 1-bit digital input point mapped to the least significant bit of input register 30001, which is named **Tag_Name**

Modbus string points

Modbus string points can function as either input (read) only or input/output (read/write) points. This is set by checking the "Write" checkbox in the Telemetry Server dashboard.

Modbus string point syntax

Telem_Server_<ScopeName>:%<OutstationFullName>;<PointType>;<PointAddress>;<Length>;<StringType>{;[<PointName

- **PointType:** STR
- **PointAddress:** a 5 or 6 digit decimal number, depending on the Point Address Range setting of the outstation. The default setting is 5 digits. The range of valid addresses vary by point type and maps to the "Address" field in the Telemetry Server dashboard.
- **Length:** a decimal number from 1 to 247 that sets the maximum string length and maps to **Requested String's Length** in the Telemetry Server dashboard.
- **String type:** one of the values listed in the following table, for example, ASCIILOW. This value maps to the "StringType" field in the Telemetry Server database.

String Type	Definition
ASCIILOW	Single byte ASCII, first character in low byte
ASCIIHIGH	Single byte ASCII, first character in high byte
UNICODE	Two byte Unicode

- **PointName (optional):** a descriptive name for the point, for example, LiftStationPump_01. If specified, the PointName must be enclosed in brackets.
  - If PointName is not specified, the point is assigned a default name that uses point type and point address, in the format **PointType_PointAddress**, for example, STR_30002.
  - If you want to create the new point in a different group than the Outstation group, include the group name: **GroupName.PointName.**

String reference example

Without optional PointName:

Telem_Server_Node-12:%Group.Modbus_RTU;STR;30002;20;ASCIILOW

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

References the 20-character ASCII string point mapped to a block of input registers starting at 30002, which is given the name **STR_30002**

With optional PointName:

Telem_Server_Node-12:%Group.Modbus_RTU;STR;30002;20;ASCIILOW;[Tag_Name]

References the 20-character ASCII string point mapped to a block of input registers starting at 30002, which is named **Tag_Name**

Telemetry server static data references

Static data references are used when the objects have already been defined in the Telemetry Server. Static data reference syntax is common to all protocols in Telemetry Server, whereas dynamic data reference syntax varies by protocol. Static data references can use any of the following syntaxes:

Telem_Server_<ScopeName>:<Item>

Telem_Server_<ScopeName>:<Item>.<Property>

Telem_Server_<ScopeName>:<Item>.<DatabaseAggregate>

Telem_Server_<ScopeName>:<Item>.<DatabaseAggregate>.<Property>

Telem_Server_<ScopeName>:<Item>.<Method>

Telem_Server_<ScopeName>:<Item>.<DatabaseAggregate>.<Method>

Where

- <ScopeName> is by default the name of the computer where the Telemetry Server instance is installed. You can customize the "ScopeName" field in the Telemetry Server to accommodate redundancy.
- <Item> is the location of the database item.
- <Property> is the name of the property tag that you want to reference. If you omit <Property>, the tag uses the default property for the defined database item or aggregate. The default property for an item or aggregate varies according to the type of item or aggregate. For example, the default property for a point is CurrentValue.
- <Database Aggregate> is the name of the database aggregate that has the required property. Database Aggregates are extensions to a table in the database and they have their own name, table, properties and methods.
- <Method> is the name of the method that you want to reference. Method static tags are write-only. Attempting to read a method tag generates an error.
  - Method static tags are write only. Writing to a method tag invokes the method, and thus triggers the associated action. Attempting to read a method tag generates an error.
  - Only methods with zero or one arguments are supported in static tags. When you write to a method tag with zero arguments, the value being written is discarded. When you write to a method tag with one argument, the value being written is passed as the argument to the method. Therefore, the argument must match the value.

Static data reference examples

Example of the static data reference syntax:

Telem_Server_Node12:Group01.Group02.Tank_Level

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

You can trigger the Telemetry Server to call a dial-up outstation by invoking the one shot method. Writing to this tag invokes the “OneShot” method on the “PSTN” database aggregate of an outstation called “Group.Outstation.”

Telem_Server_XXXXX:Group.Outstation.PSTN.OneShot

You can apply an override to an analog point by invoking the override method. Writing a value to this tag invokes the “Override” method on the point “Group.AI_1” to change the point to the specified value. Overriding must be enabled on the point:

Telem_Server_XXXXX:Group.AI_1.Override

Manage galaxies

You can back up and restore Galaxies, change the Galaxy you are working with, delete a Galaxy, and export and import all or part of a Galaxy.

If you want to create a Galaxy, see Create a Galaxy.

Back up and restore galaxies

Periodically, you should back up your Galaxy. Backing up your Galaxy helps if you have a computer failure or other problem. If there is a failure, you can restore your Galaxy from the backup.

Use the Galaxy Database Manager to back up and restore your Galaxy. The Galaxy Database Manager is part of the suite of AVEVA System Management Console utilities.

The aaConfigSQL utility is another utility that you may need to use when restoring Galaxies. aaConfigSQL is used to set System Platform user permissions for SQL Server. In most cases, the default level of permissions is adequate for restoring Galaxies.

However, if you are restoring a Galaxy created with an older version of Application Server, you may need to change the permissions level.

See the section about SQL Server Rights Requirements in the *System Platform Installation Guide* for additional information about the aaConfigSQL utility.

To open the Galaxy Database Manager

- From the **Start** menu, navigate to the **AVEVA** folder, and then select **Operations Control Management Console**.

  See the *Galaxy Database Manager* documentation for more information about backing up or restoring your Galaxy.

Change galaxies

You can have many Galaxies in the Galaxy Repository. If you are a systems integrator, you have at least one Galaxy for every client.

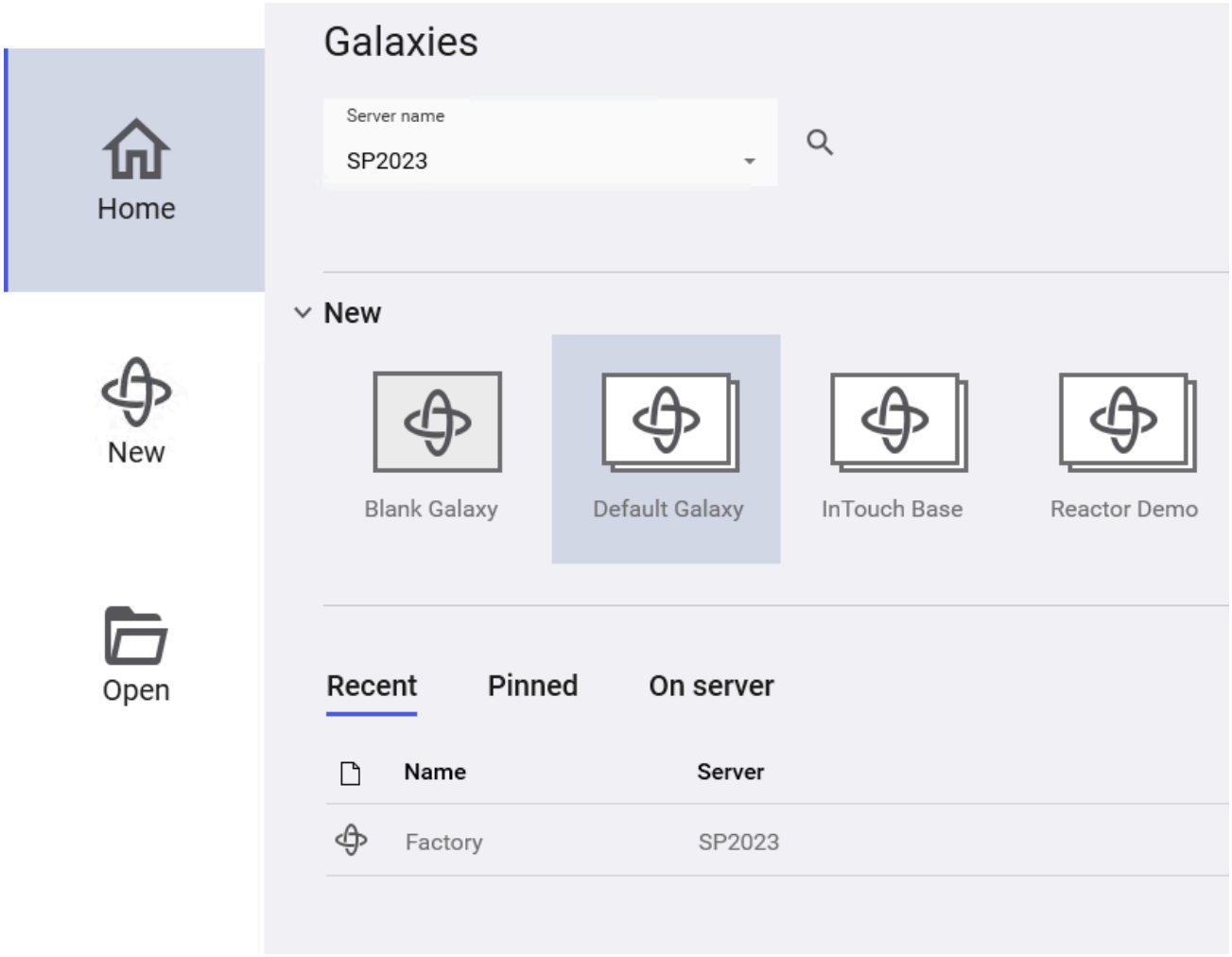
diagram of GR node and the galaxies it contains

For a Galaxy to deploy objects to other computers in the Galaxy, the Galaxy Repository (GR) must host a platform defined in that Galaxy. Because of this, you can only deploy one Galaxy from the Galaxy Repository at a time to the computers on your network. For more information about deploying, see Deploy objects.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

To change from one Galaxy to another

1. On the ribbon. select **Galaxy.** The **Galaxies** page appears.



2. Follow the instructions in Connect to a Galaxy.

Delete a galaxy

You can delete a Galaxy. Deleting a Galaxy removes all of the Galaxy information from your computer.

Before you delete a Galaxy, you must undeploy all objects within it. For more information about undeploying objects, see Undeploy objects.

Make sure you select the right Galaxy to delete. After you delete a Galaxy, you cannot undelete it. You can only recreate it by restoring from a backup. For more information, see Back up and restore galaxies.

To delete a galaxy

1. Undeploy all objects from the Galaxy you want to delete.
2. Close any Galaxy and connected IDE you have open.
3. Start the IDE, or if you are connected to a different galaxy, select **Galaxy** from the ribbon.. The **Galaxies** page appears.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

4. Select the Galaxy you want to delete, except the connected Galaxy.
5. Click the **Delete** icon next to the galaxy name.
6. At the prompt, click **Yes**.
7. When the Galaxy is deleted, click **Close**.

Galaxy object components synchronization

The Galaxy Object Components Synchronization feature provides a detection and recovery mechanism when the client is out-of-sync with the server. Synchronization means synchronizing the installed object components on the client node with respect to the connected GR.

Synchronization of object components as defined in this section is fundamental to the subsequent functionality. For that reason, it is unnecessary to synchronize script libraries, graphics or InTouchViewApp-related files, client controls, IDE extensions, run-time components, security, or history data.

In the case of a run time out-of-synchronization condition, recovery is accomplished by redeploying the run-time platform.

Definitions

The following terms are important in understanding the Galaxy Object Components Synchronization feature:

- **Object Component:** Base template-dependent files
- **Server:** The GR node, which maintains the object component information
- **Client:** The IDE

Typical out-of-sync scenarios

The GR and the client nodes can become out-of-sync in a number of ways. The following are the typical causes:

- Re-imaging the client or server node or both
- Restoring a Galaxy from a backup
- Deleting and creating a Galaxy on the server node

Use the synchronization feature

The synchronization detection and resolution process is the same for all scenarios. Out-of-sync conditions between a client and a Galaxy are detected and resolved at connection time.

If the system recognizes that the minimum connectivity requirements are met, and there are object versioning conflicts, then you are prompted about the out-of-sync state with the option either to synchronize or to reject the synchronization operation.

Select the option you prefer. If you select to synchronize, the synchronization operation will run without further prompts.

A warning message is placed in the Logger for any out-of-sync detection and synchronization issues.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Exportg instances to a CSV file

You can export some or all instances and their configuration data to a comma-separated values (CSV) file. This file has a .csv extension and is sometimes called a galaxy dump file. After you export the instances, you can edit the CSV file with Microsoft Excel. This makes it very easy to do large editing changes, such as search and replace.

This feature exports only instances. You cannot export templates to a CSV file.

If you have an instance checked out when you export it, your checked-out version is exported to the CSV file. Otherwise, the checked-in version of each instance is exported, even if an instance is checked out by another user.

The CSV file contains attributes that are unlocked and configuration time-writeable, one column per attribute. If the Object Wizard was used to configure the instance, there are also columns for each Object Wizard choice and option.

The following are not exported:

- Script libraries. Scripts within an object are exported.
- Attributes that are not text-based. For example, type QualifiedStruct is not exported.
- Custom object online help files.

Note: I/O Auto Assignment information is not included in the CSV file. For information about manually adding I/O Auto Assignment information to a CSV file for inclusion so it can be imported with the instance, see Enable I/O auto assignment in the exported CSV file.

See About the CSV file structure for specific information about the structure of the .csv file. Before you start, make sure you know what instances you want to export to a file.

To export objects to a CSV file

1. In the **Application** views area, select one or more instances. **Shift+click** to select multiple instances. You can export all instances derived from a template by selecting the template.
2. From the ribbon, select **Galaxy**, then select **Export.**
3. Select **Objects,**. then **Selected as CSV**. The **Export selected objects as CSV** dialog box appears.
4. Browse to the folder where you want to create the `.csv` file. Type the name of the file. Select **Save**.
5. When the Galaxy export process is done, click **Close**. A `.csv` file is created containing the selected objects and configuration data.

Edit the exported instance data

You can open the .csv file containing the exported instances in a text editor such as Notepad or in Microsoft Excel. When you have finished editing the file, save it as a plain text .csv (comma-separated value) file to import back into the Galaxy.

About the CSV file structure

The Galaxy CSV file has a specific structure. This section describes that structure.

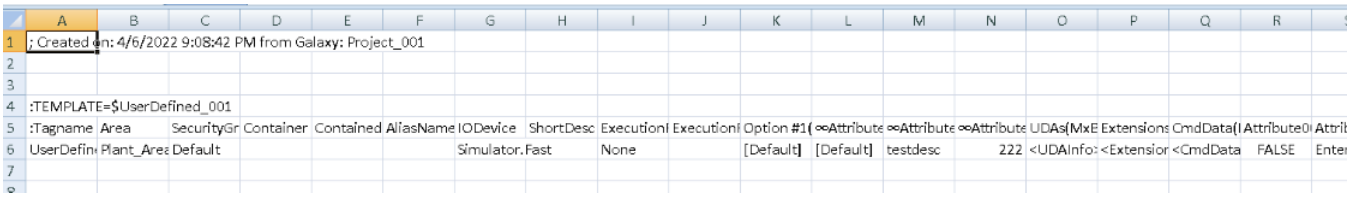
Objects are grouped in the `.csv` file based on the template each is derived from. A header row per template

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

indicates what is in each column for those instances. After columns containing general information about the instance—such as its security group, area, I/O device, and scan group—the rest of the columns contain attribute information.

The headings of columns containing Object Wizard selections show the choice group or option prompt. A column header that starts with an infinity symbol (¥) indicates an attribute value that was set by the Object Wizard.

The column structure makes it easy to read the information after opening the `.csv` file in a text editor or Microsoft Excel. You can then carefully make any changes before re-importing the file.



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
1	; Created on: 4/6/2022 9:08:42 PM from Galaxy: Project_001																	
2																		
3																		
4	:TEMPLATE=$UserDefined_001																	
5	:Tagname	Area	SecurityGr	Container	Contained	AliasName	IODevice	ShortDesc	ExecutionI	ExecutionI	Option #1(	∞Attribute	∞Attribute	∞Attribute	UDAs(MxE	Extensions	CmdData(I	Attribute0
6	UserDefin	Plant_Area	Default				Simulator.	Fast	None		[Default]	[Default]	testdesc	222	<UDAInfo:	<Extensior	<CmdData	FALSE
7																		

You can add comments by adding a line with a semi-colon as the first character.

Host attributes

Exported CSV files contain a column for the Host attribute of the objects being exported. For Platform objects, Host is always the name of the Galaxy from which the object is being exported.

This data is ignored in subsequent import operations because the Host for Platform objects is automatically the name of the Galaxy into which it is being imported, regardless of the name of the Galaxy from which it was exported.

Using a text editor, you can delete the Host attribute column, like any other data in the CSV file. This has no effect on Platform objects in subsequent import operations because they take the Galaxy name as their Host.

About quotation marks and carriage returns

Carriage returns in scripts associated with exported objects are replaced with \n in the `.csv` file. If you edit the CSV file, **do not** delete the \n characters. If you edit scripts in the file, use \n for a carriage return. This character set is interpreted as a carriage return when the CSV file is used in an object import.

When editing a script in an exported file, use \\n if you want to include the backslash character (\) followed by the letter n. This character set is not converted to a carriage return during an import.

Be careful when adding or editing quotation marks in the CSV file. Type all single quotation marks as two single quotation marks and surround the entire string with opening and closing quotation marks.

Make sure the string contains an even number of quotation marks. When the object is loaded during an object import, the extra quotation marks are stripped from the string.

For example, if you want to enter `3"Pipe` as a Short Description, add a second quotation mark (`3""Pipe`) and then surrounding quotation marks (`"3""Pipe"`).

Time formats in Excel

If you edit an exported CSV file in Microsoft Excel, be careful typing time entries. Excel can change the time format and the resulting entries do not work when you import the changed objects.

The import objects from CSV feature accepts two time formats:

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

- DAYS HH:MM:SS:SSSSSS - the number of DAYS is followed by a space.
- HH:MM:SS:SSSSSS - Excel automatically changes the entry to an incompatible format.

When you type time entries in Excel, use the following format:
DAYS HH:MM:SS:SSSSSS.

For example:
0000 01:02:12.123: 1
04:03:06.12:120
22:66:88:123456

Enable I/O auto assignment in the exported CSV file

You can use exporting to a CSV file to convert a Galaxy that contains traditional I/O hard coding to use I/O auto assignment instead. If you are planning to expand an existing Galaxy by adding many objects, or if you are upgrading PLCs, converting hard coded I/O references to auto assignment can save you time. To convert the references, edit the CSV file as described below. This bulk edit is faster than changing each object individually in the IDE object editor.

The exported CSV file displays the entry "---Auto---", for input and output attributes that use I/O auto assignment. This entry indicates that I/O auto assignment is enabled for the attribute. The actual I/O auto assignment to a DI Object and scan group is not preserved in the exported file. If you want to preserve existing I/O auto assignment settings, export objects to a package files instead of to a CSV file. See Export objects for additional information.

To convert hard coded I/O references, edit the CSV file to replace the hard coded reference with "---Auto---" for each <attribute>.InputSource and/or <attribute>.Output.Dest. Import the edited objects back into the Galaxy, and then assign the objects in the **IO Devices** view of the IDE to a DI Object and scan group.

For more information about I/O auto assignment, see I/O auto assignment.

To convert hard coded I/O references to I/O auto assignment

1. Export the objects with hard coded I/O references to a CSV file.
2. Open the exported .CSV file in a text editor or Excel.
3. In the data row of the Attribute.InputSource column, delete the hard coded I/O assignment. Enter "---Auto---" in the data row.
4. Repeat for each object that you want to enable for I/O auto assignment.
5. In the data row of the Attribute.OutputDest column, delete the hard coded I/O assignment. Enter "---Auto---" in the data row.
6. Repeat for each object that you want to enable for I/O auto assignment.
7. Save your changes.

To complete the process of activating I/O auto assignment

1. Import the objects in the edited CSV file. See Import objects from a CSV file for additional information.

   Once imported, the objects will appear in the **IO Devices** view, under the Unassigned IO Device folder.
2. Drag and drop objects in the **IO Devices** view to link the objects to the applicable DI Object and scan group. See I/O auto assignment for additional information.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Import objects from a CSV file

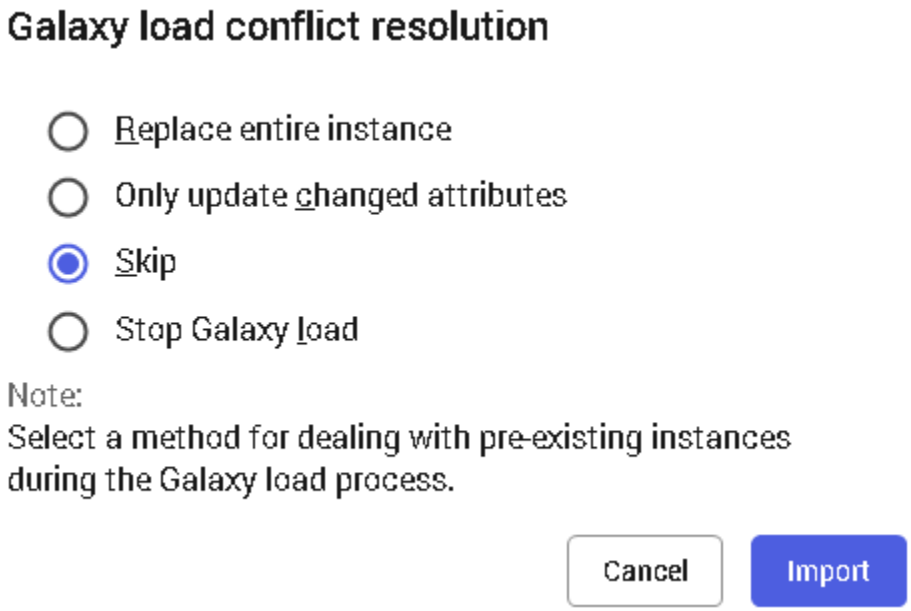
After you are done editing a CSV file, you can import the objects in it back into your Galaxy. This is also called loading the objects into the Galaxy.

An exported CSV file contains only instances. Templates cannot be exported to or imported from a CSV file See About the CSV file structure for more information about the contents of the CSV file.

Note: A comment line in a CSV file created in Microsoft Excel can create an unintended default-value object. To avoid this, open the CSV file in Notepad to make sure the comment line does not contain quotation marks.

To import a .csv file

1. On the ribbon, select **Galaxy**, then select **Import**.
2. Select **Objects**, then **Objects from CSV**. The **Importing automation object(s) from csv** dialog box appears.
3. Browse to the .csv file that contains the objects you want to import. Select the file and click **Open**.
4. The **Galaxy Load Conflict Resolution** dialog box appears. Use it to resolve conflicts that occur if objects that you want to import already exist in the Galaxy.



5. Select one of the following:
   - **Replace entire instance** if an instance of an object with the same name already exists and you want to replace it entirely with the object in the import file.
   - **Only update changed attributes** if an instance with the same name already exists and you want to replace only the attributes of the object where the values are different.
   - **Skip** if an instance with the same name already exists and you want to keep the version already in the Galaxy.
   - **Stop Galaxy load** if an instance with the same name already exists and you want to cancel the object import.
6. Click **Import**. A progress box appears showing the import progress.

   When the import is done, all objects changed or created during the import process are checked in.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Synchronizing time across a galaxy

Certain Application Server functions like scripting, alarming, and historizing require that all member computers of a galaxy are synchronized to the same time. A time master is a network-time-protocol server that provides a time to which other nodes on your network can synchronize.

The System Management Server node should be designated as the time master. The Application Server nodes in the galaxy periodically synchronize their clocks to the time master.

Time synchronization in Windows domains

The system administrator of the Windows domain may have time synchronization configured already. If this is the case, configuring a galaxy time master is unnecessary and can conflict with the existing time synchronization.

Application Server does not implement its own time synchronization algorithm, but supports time synchronization for your galaxy through windows time service. If needed, you can configure a galaxy time master to utilize the Windows time service on a node that you designate as the time master, preferably the System Management Server node. If a node in your galaxy has a system time that is outside of a predetermined amount (for example, five minutes), certain operations such as deployment may fail.

For information about configuring the Windows time server, refer to the following Microsoft documentation:

https://learn.microsoft.com/en-us/windows-server/networking/windows-time-service/configuring-systems-for-high-accuracy

Synchronization schedule

The designated node clock serves as the master clock for all timestamping functions. Time synchronization is based on the Microsoft Windows time service.

All WinPlatforms begin synchronizing the time on their node when they are deployed.

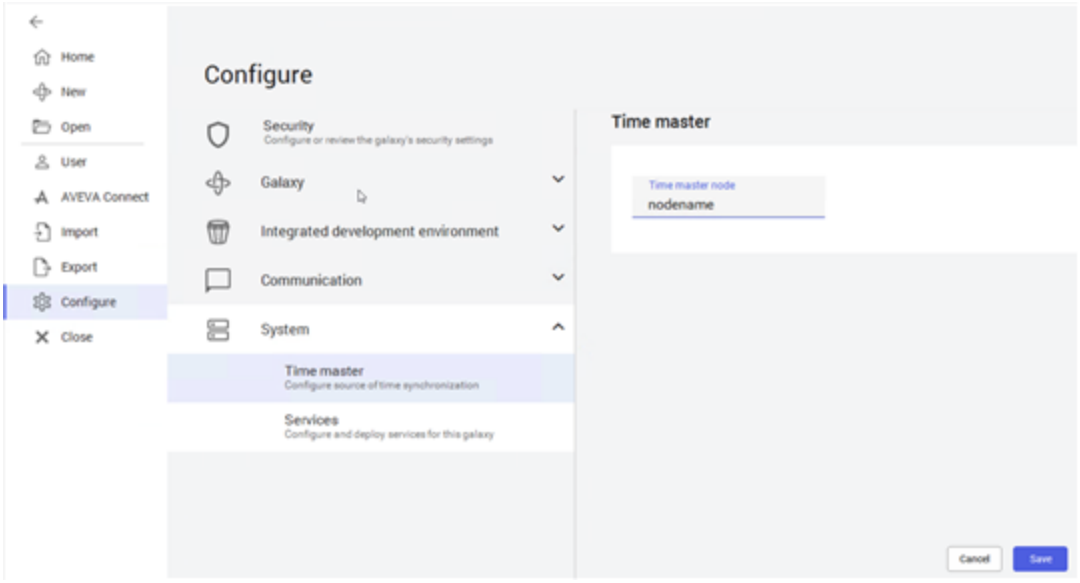
You can specify a time master node in another time zone. The time on each Application Server node is set to the time specified on the node in the other time zone.

Configure the time master

To configure a time master node

1. Select the **Galaxy** backstage, then navigate to **Configure** and then expand the **System** option.
2. Select **Time master**. The **Time master** dialog box appears.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.



3. Enter the node name in the **Time master** dialog. We recommend that you use the System Management Server node as the time master.
4. Click **Save**.

Host multiple galaxies in a galaxy repository

You can create and configure multiple Galaxies in a single Galaxy Repository on the same computer. You can configure one Galaxy from an IDE and then change Galaxies and configure the second one using the same IDE.

Note: If you try to deploy objects from the second Galaxy to a computer that hosts deployed objects from the first Galaxy, the deploy fails.

Disk space requirements

After Application Server is installed, certain operations require at least 100 MB of available disk space. These operations include

- Creating a Galaxy
- Deploying objects
- Importing and exporting objects
- Loading and dumping a Galaxy
- Restoring and backing up a Galaxy.

This minimum requirement applies to the Galaxy node as well as any remote IDE nodes.

If your computer has less than 100 MB of available hard disk space, running any of these operations can result in erratic behavior.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Manage communication between galaxy nodes

You can use Application Server in a distributed environment. All computers with System Platform software installed must be able to communicate with each other.

Two items must be considered to ensure communication between nodes:

- Network user accounts (initially created during System Platform installation)
- Multiple network interface cards in computers

All Platforms communicate with several attributes on the GR Platform to detect configuration changes. This communication sends NMX heartbeats from each Platform to the GR Platform. For example, the attributes include "time of last deploy," and "time of last configuration change."

In addition to NMX heartbeats, all Bootstraps on each Platform in the Galaxy send each other heartbeats. These heartbeats are for Platform status information, occur every two seconds, and are configurable.

Map network drives with user account control enabled

Mapping network drives with Windows User Account Control (UAC) enabled can create issues with subsequent access to those drives. Windows considers mapping a network drive and later access to that drive, to import objects from a network location, for example, as separate logon contexts, and issues separate credentials. Without correct credentials, the mapped network drives can become inaccessible.

You can ensure that no issues occur with mapped drives by using one of two methods to map the drives:

1. Map drives from the command prompt as an administrator.

   From Windows **Start/All Programs**, click **Accessories**, then right-click **Command Prompt** and select **Run as administrator** on the context menu.

   At the command prompt, type the "net use" command, using the following format:

   ```
   net use \\<computername>\<sharename> /user:<username>
   ```

   Example:

   ```
   net use z: \\MainComputer\ObjectFiles /user:DRoberts
   ```

   where the drive being mapped is z:, the computer name is MainComputer, the share name is ObjectFiles, and the user name is DRoberts.
2. Map drives from the IDE using elevated privileges.

   For example, you can map drives from the **Import Object(s)** dialog.

About the System Platform network account

Communication occurs with a System Platform Network Account set up during the initial installation of each System Platform component on each computer, including the IDE. See Change network account utility for additional information.

WARNING! The Network Account that is used for System Platform communication is a standard Windows operating system account located on the local computer or on a domain. Do not delete this account with operating system account management tools. If you do, the IDE stops functioning.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

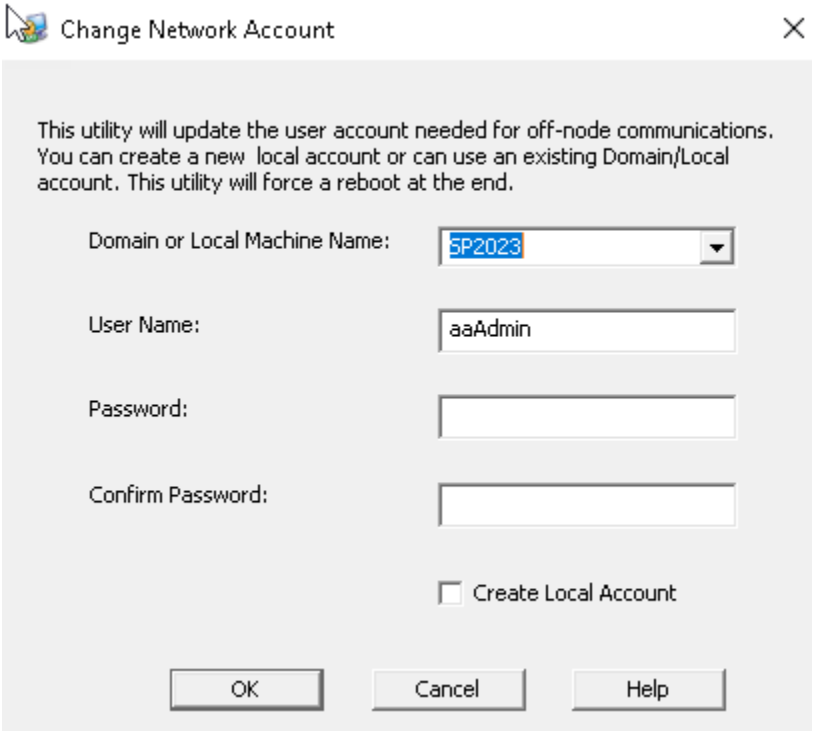
If you delete the Network Account on an IDE node, you must recreate it with the Change Network Account utility. You must have Administrator privileges on the computer to make changes with the Change Network Account utility.

Before you start, find out the user name and password that is created on all computers with System Platform software installed.

Important! Completing the account change requires a system restart. Once you press OK, you cannot cancel the account name change or the system restart.

To recreate the Network Account

1. From the **Start** menu, navigate to the **AVEVA** folder, and then select **Change Network Account.**



2. The domain name or local machine name for the old Network Account is shown. If you need to change to a different domain, use the short domain name. Do not use a fully qualified domain name (FQDN). For example, use "DomainName" and not "DomainName.com" or "DomainName.local."
3. Do one of the following:
   - In a single-node System Platform installation, enter a new or existing account name.
   - In a multi-node System Platform installation, create the same user account with the same user name and password on all other Systtem Platform computers.

   If you are using an existing account name or a domain account, leave the "Create Local Account" checkbox unchecked.
4. Click **OK** or **Cancel**. If you click OK, the system will warn you to shut down all open applications and that the computer will be restarted. You cannot cancel the account change or system restart at this point. After you recreate the Network Account, the Microsoft Windows security component on your computer can take several minutes to update this information on the Application Server Galaxy node. Until that happens, your IDE might not function properly. Restarting the Galaxy node applies this update immediately.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Multiple network interface cards

System Platform nodes can be configured to use two or more network interface cards (NIC) for redundancy or other requirements. Each NIC must be configured to allow communication with other System Platform nodes.

If a node contains multiple NICs but is using only one of the cards, disable all cards except the one in use, that is, the NIC for the supervisory network.

Define the NIC order

For any System Platform node with multiple NICs to communicate with other Galaxy nodes, you must define the correct order of network connections in the network services of the computer.

Before you start configuring multiple network cards, you need the IP address for the second and subsequent nodes.

See Configure multiple NICs for detailed instructions.

Configure IP address and DNS settings

After you specify the order of the network cards, you are ready to configure several other parameters to ensure successful node-to-node communications in the System Platform environment.

You must configure the IP address and DNS settings as follows for each network card to function properly.

To configure the IP address and DNS settings

1. In the **Network Connections** dialog box, right-click the network connection and click **Properties** in the context menu. The **Properties** dialog box for this connection appears.
2. In the list of components used by this connection, select **Internet Protocol (TCP/IP)** and click **Properties**. The **Internet Protocol (TCP/IP) Properties** dialog box appears.
3. For the supervisory network, select **Obtain an IP address automatically**.

   For the other network connections, select **Use the following IP address**. Type the correct IP address for the secondary and further cards. See your network administrator for the proper settings for the remainder of the parameters in this group.
4. Click **Advanced**. The **Advanced TCP/IP Settings** dialog box appears. Click the **DNS** tab.
5. For the supervisory network, select **Register this connection's addresses in DNS**.

   For the other network connections, clear this check box.
6. Click **OK**.

Configure multiple NICs

Network profiles are identified and assigned during computer startup and each time a connection changes. The three profiles currently in use in the supported Windows operating systems are:

- **Domain profile:** Active only when the computer can authenticate with a domain controller on all active interfaces (for example, LAN, wireless, and VPN). The domain profile may be more or less restrictive than the other two profiles depending on network security policies.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

- **Private profile:** Active whenever the network type for all active network connections on the computer are identified as private networks. The private profile typically is used in a more trusted environment and is less restrictive than the public profile to allow for network discovery.
- **Public profile:** Active in all other circumstances. The public profile typically is more restrictive than the private profile because the computer often is connected to the Internet in an insecure location. Network discovery and remote access are disabled rather than explicitly blocking specific traffic.

All programs allowed to communicate through the firewall are blocked if one or more network interface cards (NICs) are configured as "Public Network." When the IP address of a network card changes, the NIC is automatically updated to "Public Network." If even one NIC in a computer with multiple NICs is configured as "Public Network", the firewall exceptions are disabled.

This is particularly important for computers configured to run as a redundant pair. If the TCP/IP properties for a network card are set to obtain an IP address automatically, there is a possibility that the network address will change when the computer restarts. The computer may suddenly change from having a "Private Network" to having a "Public Network," and potentially block the programs that were previously allowed to communicate through the firewall.

Note: An exception to this would occur if the NICs are configured and set to the Public Profile before Application Server is installed. The Application Server installation would allow the necessary programs to communicate through the firewall in the current (Public) profile.

The following describes the key NIC settings for a redundant pair. You configure these settings for both the primary and backup computers. These settings apply if more than two NICs are installed for redundancy (RMC) or other networking purposes, such as networking PLCs.

Important! When configuring NICs for a redundant pair, speed and duplex settings for the standby NIC must match the settings for the primary NIC.

For detailed instructions on how to configure multiple network interface card binding order settings, see Define the NIC order and Configure IP address and DNS settings.

To configure multiple NICs

1. Under **Internet Protocol Version 4 (TCP/IPv4) Properties**, select the **Use the following IP address option** and provide an IP address, subnet mask, and default gateway.
   a. For the primary computer, the default gateway should be the IP address that you configure for the backup computer. For example:

      IP address: 100.100.100.94

      Subnet mask: 255.255.255.0

      Default gateway: 100.100.100.95

   b. For the backup computer the default gateway should be the IP address that you configure for the primary computer. For example:

      IP address: 100.100.100.95

      Subnet mask: 255.255.255.0

      Default gateway: 100.100.100.94

   c. A NIC acting as Redundancy Message Control (RMC) does not require a default gateway. The IP address and Subnet mask can be configured as described for the primary and backup computers. The IP address must be configured for a different network than the primary and backup computers. Windows will identify this NIC and assign it a private profile. For example:

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

IP address: 192.168.1.100

Subnet mask: 255.255.255.0

2. Under **Advanced Settings**, check that the **Register this connection's address in DNS option** is not selected.

To set the binding order in Windows 8.1 and Windows Server 2012 / 2012 R2

1. Make sure that the primary network adapter is first in the binding order. This places it above the RMC network in the binding order.
2. Make sure that ALL network cards are configured as "Private Network."
3. Configure the RMC NIC to have a persistent (across reboots) Private profile.
   a. Open a command window and enter the command `secpol.msc`. The **Security Policy** window appears.
   b. Select **Network List Manager Policies**.
   c. Select **Unidentified Network**.
   d. Right click, then select **Properties**, and then change the **Location** type from **Not configured** to **Private**.
   e. Close the command window.

It may be necessary to restart the computer for the changes to take effect.

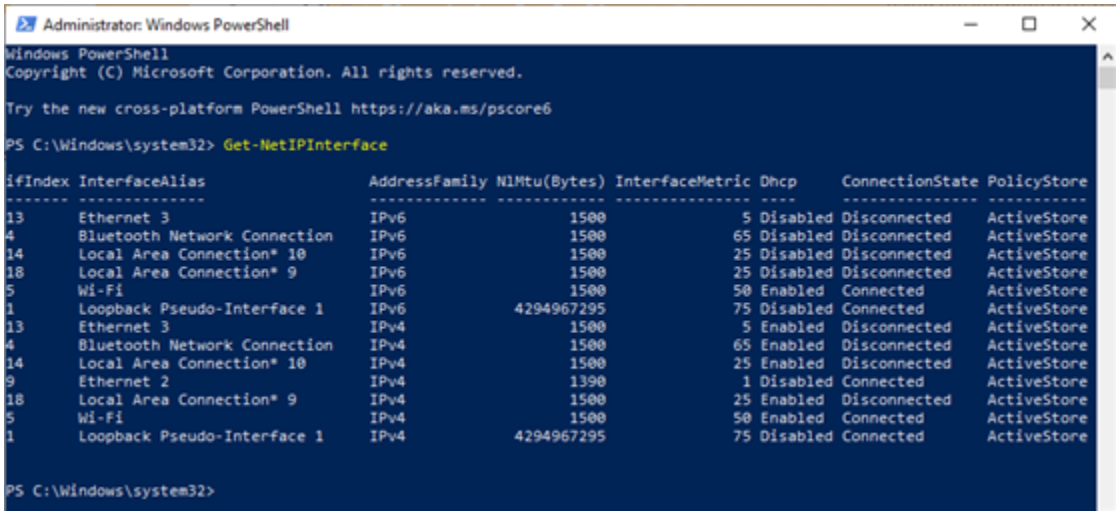
To set the binding order in Windows 10 and Windows Server 2016

Microsoft has changed how network protocol bindings are handled in its newest operating systems. Windows now uses an Interface Metric number to determine priority.

To check the existing priorities of your network cards:

1. Launch the Windows PowerShell as administrator.
2. To view the current priority, enter the following cmdlet:

   Get-NetIPInterface



   This displays the Interface List. An interface metric number is listed for each network interface card/adapter. Note that the lowest number has the highest priority.

3. Identify the NIC cards and define the binding order (priority). Check the InterfaceIndex numbers for each NIC. It could be that your cards already have the correct priorities (higher InterfaceIndex numbers have lower priority).

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

4. If you need to change the priority of your interface cards, use the following cmdlet to change the InterfaceIndex for one or more network cards. For example:

   Set-NetIPInterface -InterfaceIndex 2 -InterfaceMetric 100 (lower priority)

   Set-NetIPInterface -InterfaceIndex 5 -InterfaceMetric 10 (higher priority)

5. Close the PowerShell.

If the two computers are set up properly, you can install Application Server and other System Platform software, and the OSConfigurationUtility sets up the firewall exceptions appropriately.

If you install Application Software and other System Platform software before configuring the NICs properly, configure the NICs as described in this section and then run the OSConfigurationUtility again. To do this, run the OSConfigurationUtility.exe, located in the **C:\Program Files (x86)\Common Files\ArchestrA** folder.

To verify that the firewall exceptions are set, open **Control Panel** and then open the **Windows Firewall** application. Verify that the executables for the System Platform products you have installed are allowed to communicate through the firewall.

Redundancy

You can set up and run Application Server in a redundant environment. For more information about redundancy, see the AVEVA Software Global Customer Support website.

Redundancy overview

A failover is the condition during which run-time operations are moved from one critical component to another. Failover can occur due to failure conditions or it can be forced manually, called a forced failover.

Application Server provides redundancy in two critical areas:

- **AppEngine redundancy:** You must configure two AppEngine objects, one as primary and one as backup, and two WinPlatform objects to host them.

  Note: When you enable redundancy for the initial (primary) AppEngine, the backup AppEngine is automatically created.

- **Data acquisition redundancy:** You must configure two DIObjects (data sources) and a RedundantDIObject.

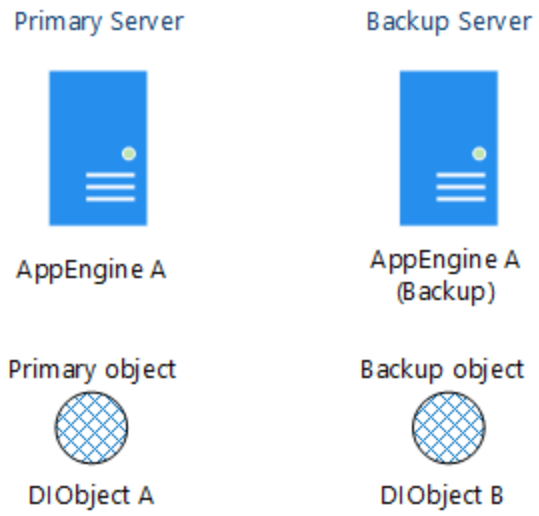
The Primary/Backup AppEngines form a redundant pair. The Application Server infrastructure will generate the backup AppEngine automatically when redundancy is enabled for an AppEngine. Hierarchy of ApplicationObjects can only be assigned to the primary AppEngine. The primary and backup engines need to be assigned to redundancy enabled platforms, and they can be deployed separately.

For data acquisition, the Primary/Backup DIObjects (the data sources) must be separately created, configured, and deployed. Also, you must create, configure, and deploy a RedundantDIObject to control failovers between the two data source objects.

In a redundant system, install and configure the primary OPC server on the backup engine node.

When you configure redundancy, you configure the Primary object and the Backup object.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.



- **Primary object**: The main or central object that provides the functionality during run time. For AppEngines, this is the object you enable for redundancy. For data acquisition, this is the DIObject you intend to use first as your data source in run time.
- **Backup object**: The object that provides the functionality of the Primary object when the Primary object fails. For AppEngines, this is the object created by the Application Server infrastructure when the Primary object is enabled for redundancy. For data acquisition, this is the DIObject you do not intend to use first as your data source in the run-time.

Configure AppEngine redundancy

Important! Every redundant Application Server run-time node must be configured to use the System Management Server if data is being historized. Redundant nodes have an instance of HCAP running, which is used to synchronize tags and store-and-forward data between redundant AppEngines. With the release of System Platform 2023 R2, secure communication is required for HCAP, and thus, redundant nodes will not function without the SMS.

Data loss will occur if the SMS is not configured, and warnings and error messages will be generated.

See Common Platform for detailed instructions about SMS configuration.

The Primary/Backup AppEngines form a redundant pair. Application Server will generate the backup AppEngine automatically when redundancy is enabled for an AppEngine. Hierarchy of ApplicationObjects can only be assigned to the primary AppEngine. The primary and backup engines need to be assigned to redundancy enabled platforms, and they can be deployed separately.

For data acquisition, the Primary/Backup DIObjects (the data sources) must be separately created, configured, and deployed. Also, you must create, configure, and deploy a RedundantDIObject to control failovers between the two data source objects.

In a redundant system, install and configure the primary OPC server on the backup engine node.

When you configure redundancy, you configure the Primary object and the Backup object.

Diagram of primary and backup objects

- **Primary object**: The main or central object that provides the functionality during run time. For AppEngines, this is the object you enable for redundancy. For data acquisition, this is the DIObject you intend to use first

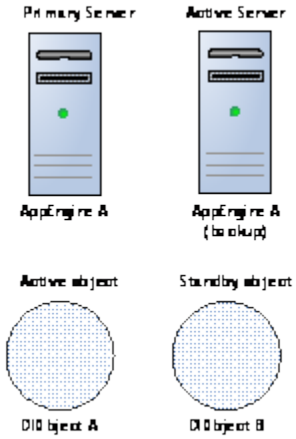
© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

as your data source in run time.

- **Backup object**: The object that provides the functionality of the Primary object when the Primary object fails. For AppEngines, this is the object created by Application Server when the Primary object is enabled for redundancy. For data acquisition, this is the DIObject you do not intend to use first as your data source in the run-time.

Redundancy during run time

When you deploy these objects to a run-time environment, the configuration objects become the Active object and the Standby object.



- **Active object**: The object that is currently executing functions. For AppEngines, it is the object that is hosting and executing ApplicationObjects. For data acquisition, it is the object that is providing field device data through the RedundantDIObject.
- **Standby object**: The object that is waiting for a failure in the Active object or for a force-failover. For AppEngines, it is the object that monitors the status of the Active AppEngine. For data acquisition, it is the object that is not providing field device data through the RedundantDIObject.

In the AppEngine redundancy environment, the Active and Standby objects monitor each other's statuses and switch during a failover situation.

During a failover, when an Active engine fails, it restarts and the Standby engine becomes active. The Active engine then turns to the standby mode. In the data acquisition environment, the RedundantDIObject monitors the status of the two DIObject data sources, and handles the switching from Active to Standby objects.

The relationship between the configuration time (Primary/Backup) and run-time (Active/Standby) object pairs is not static. In the run time, either the Primary or Backup object can be the Active object at any particular time. Whenever one becomes the Active object, the other automatically becomes the Standby.

Engine restart after failure

An AppEngine will restart automatically after an engine failure such as a crash or hang. If the engine is redundant, it will synchronize with its redundant partner, and running applications will resume.

Engine restart on failure is automatic and cannot be disabled.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

CPU load balancing

CPU load balancing for redundant AppEngines can be used to maintain system stability by limiting high CPU usage during AppEngine startup after failover. When CPU usage by a core is close to 100%, other processes can be starved and startup times can extend as a result. When CPU load balancing is enabled, objects are loaded in discrete chunks, thus improving CPU utilization.

Application Server includes a registry key that you can use to configure CPU load balancing for redundant engines. CPU load balancing is disabled by default, and the configuration of each node is independent. That is, you can enable CPU load balancing on some nodes and not on others, and have different settings for each node.

Note: CPU load balancing will only work on run-time nodes with redundant engines.

- Registry Subtree – 64-bit:
  `HKEY_LOCAL_MACHINE\SOFTWARE\Wow6432Node\ArchestrA\Framework`
- Registry Subtree – 32-bit:
  `HKEY_LOCAL_MACHINE\SOFTWARE\ArchestrA\Framework`
- Registry Key:
  `EngineFailoverStartup`

Subkey entries used for enabling and tuning CPU load balancing are:

Subkey name	Default value	Description
EnableCPUBalancing	0	Turns CPU load balancing on or off (0 = OFF, 1 = ON)
LoadObjectsNumberPerGroup	50	Specifies the number of objects loaded at a time during startup. If set to 0, this function is disabled.
LoadObjectsTimeToSleep	200	Specifies in milliseconds, the time between loading groups of objects (end of load to start of next load). If LoadObjectsNumberPerGroup is 0, sleep is disabled.
FirstScanCycleNumberPerGroup	25	Specifies the number of objects to be initialized during the first scan cycle after failover. If set to 0, this function is disabled.
FirstScanCycleTimeToSleep	100	Specifies in milliseconds, the time between initializing groups of objects (end of first cycle to start of second). If FirstScanCycleNumberPerGroup is 0, sleep is disabled.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

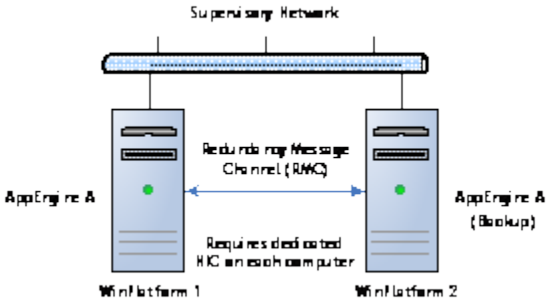
AppEngine Redundancy

You enable AppEngine redundancy in the Primary AppEngine. You must also configure two WinPlatforms for redundancy, one to host the Primary AppEngine and one to host the Backup AppEngine.

Important: While it is not a requirement to use the same operating system for WinPlatforms that host redundancy-enabled AppEngines, is recommended to do so since mixed operating system scenarios are untested.

The configuration of both WinPlatforms should be the same. At a minimum, the store-forward directory configurations must be common to both WinPlatforms:

During configuration, you can assign Primary and Backup AppEngines to the same WinPlatform. To deploy, you must assign the Primary and Backup AppEngines to different WinPlatforms.



Each production computer hosting a redundancy-enabled AppEngine must have a minimum of two network cards. One NIC is for the supervisory network and PLC network, if the computer has only two network cards. The other must be for a dedicated Ethernet connection between computers for the redundancy message channel (RMC).

For information about distributed networks, see Multiple network interface cards.

The RMC handles redundancy monitoring, message handling and data synchronization between redundant pairs.

Configure the redundancy message channel

For the redundant pair of AppEngines to successfully communicate with each other, you must define the correct order of network connections in the network services of each computer.

We recommend that you name each card with a clearly identifiable function (for example, "Supervisory Net" and "Redundant Message Channel").

You must configure at least the following parameters:

- Redundancy Message Channel IP Address
- Redundancy Message Channel Port
- Primary Message Channel

To configure network cards

1. Open the **Network Connections** dialog box in Windows. The specifics about opening this dialog box varies, depending on the version of Windows you are working on.
2. Click **Advanced Settings**. If the computer name is used in the platform's "node name" box, the first

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

connection in the list must be the supervisory network card. Use the up and down arrows to define the correct order.

3. **Connections**.

   If a computer contains more than two network cards, the supervisory net must be listed first and the RMC connection can be listed in any other position. For example, your computer might have a supervisory connection, a field device connection, and a RMC connection.

4. Configure the DNS settings for the supervisory network card to function properly.
   a. On the **DNS** page of the **Advanced TCP/IP Settings** dialog box, select the **Register this connection's addresses in DNS** check box.
   b. For the RMC network card to function properly, clear the **Register this connection's addresses in DNS** check box. For more details on these settings, see Multiple network interface cards.
5. In Application Server, open the WinPlatform in the Object Editor for the computer you just configured.
6. Change the **Redundancy Message Channel IP Address** to the IP address of the RMC connection. See the WinPlatform Help for more information about these parameters.
7. Save and close.

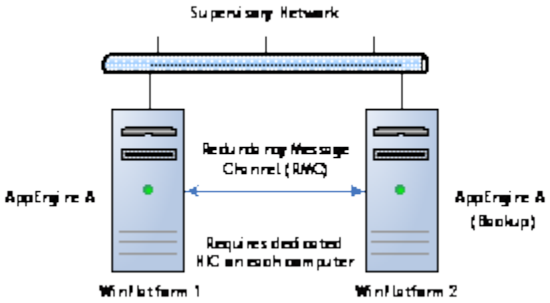
Enable AppEngine redundancy

You enable AppEngine redundancy in the Primary AppEngine. You must also configure two WinPlatforms for redundancy, one to host the Primary AppEngine and one to host the Backup AppEngine.

Note: WinPlatforms that host redundancy-enabled AppEngines do not have to be deployed on the same operating system. Using the same operating system is recommended, however, since mixed operating system scenarios are untested.

The configuration of both WinPlatforms should be the same. At a minimum, the store-forward directory configurations must be common to both WinPlatforms:

During configuration, you can assign Primary and Backup AppEngines to the same WinPlatform. To deploy, you must assign the Primary and Backup AppEngines to different WinPlatforms.



Each production computer hosting a redundancy-enabled AppEngine must have a minimum of two network cards. One NIC is for the supervisory network and PLC network, if the computer has only two network cards. The other must be for a dedicated Ethernet connection between computers for the redundancy message channel (RMC). The RMC handles redundancy monitoring, message handling and data synchronization between redundant pairs. For more information about the RMC, see Configure the redundancy message channel .

For information about distributed networks, see Multiple network interface cards.

You configure redundancy in AppEngine/WinPlatforms objects using their Object Editors.

The redundancy-related parameters in the AppEngine Object Editor are located on the **Redundancy** and **General**

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

tabs.

An AppEngine that is part of a redundancy pair has a deployment status indicating its own status and that of its partner object. These statuses are visually indicated in the **Application** views.

You can set the status of a redundancy pair to one of the following states:

Pair Deployed	Both Primary and Backup AppEngines are deployed.
Pair Undeployed	Both Primary and Backup AppEngines are undeployed.
Partial Deployed	Either the Primary or Backup AppEngine is deployed and its partner is not deployed. If an AppEngine has a Partial Deployed status, its partner has a Partial Undeployed status.
Partial Undeployed	Either the Primary or Backup AppEngine is undeployed and its partner is deployed. If an AppEngine has a Partial Undeployed status, its partner has a Partial Deployed status.

To create AppEngine redundancy

1. On the **Redundancy** tab, for the Primary AppEngine, select the **Enable Redundancy** check box.
2. On the same tab, select the **Enable Warm Redundancy** check box to improve failover performance.
   - When **Warm Redundancy** is **enabled**, failover from the active to standby engine occurs much faster than when it is disabled. Warm redundancy starts up and initializes the Standby engine concurrently with the Active engine. DI client objects subscribe to both the Active and Standby engines, even though the Standby engine is running offscan until failover occurs. The standby engine transitions to onscan at failover without requiring the DI Objects to restart and subscribe, thus greatly reducing the time needed for the standby engine to become active. This checkbox will be checked by default for new redundant pairs (Warm Redundancy enabled).

     Note: The behavior of redundant engines that use Startup or Undeploy scripts may differ between Warm Redundancy and Legacy Redundancy. For existing redundant engines that use Startup or Undeploy scripts, test engine behavior before enabling Warm Redundancy in the production environment. See Script execution types for additional information.
   - When **Warm Redundancy** is **disabled** (legacy behavior) the Standby engine does not perform start up and initialization until failover occurs, and DI objects subscribe to only the Active engine. After initialization, the engine then transitions to onscan and becomes active, and DI Objects must restart and subscribe to the now-active (formerly standby) engine. The additional time needed for start up, initialization, and subscription increases the time needed for failover to complete. This checkbox will be unchecked for migrated redundant pairs (legacy behavior; Warm Redundancy is disabled).

   Default redundancy behavior:

   - New Redundant Pairs: Run Warm is enabled by default.
   - Redundant Pairs migrated from System Platform 2020 R2 or prior release: Run Warm is disabled by default (uses legacy behavior).
3. Configure the remaining redundancy parameters as needed. See the help file for the AppEngine for specific information about these parameters.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

4. On the **General** tab, set the **Engine Failure Timeout** option to 2000 milliseconds. You may need to tune this parameter between 2000 ms and the default 10000 ms, depending on the requirements of your application.

   When the Active engine fails during a failover, it restarts automatically and becomes the Standby engine.

   Note: The actual engine failure time-out is three times the value of this parameter. If you set the parameter to 2000 ms (2 seconds), a failover occurs if the AppEngine fails to communicate with the computer's Bootstrap for 6 seconds. A setting of 10000 ms (10 seconds) can be too long a wait period (30 seconds) for a well-functioning redundancy operation.

After you save the configuration of the object and check it into the Galaxy, the icon for the object changes. A Backup AppEngine is created with the same configuration as the Primary object.

Icon	Object
	Primary AppEngine
	Backup AppEngine

Configure redundancy in the AppEngine template

You can create a redundancy-enabled AppEngine template. When you create an instance of the template, both the Primary and Backup instances are created.

The Backup AppEngine is hosted by the Unassigned Host or the default WinPlatform if you specified one in the **Configure User Information** dialog box.

If you change the configuration and check in the Primary AppEngine, the Backup AppEngine:

- Is checked out in the background
- Updates the configuration
- Is checked in without notification to connected clients

Delete redundant AppEngines

You can disable redundancy in an AppEngine after Application Server creates the Backup object. If you disable redundancy and check in the Primary AppEngine, the Backup is deleted from the Galaxy. The exception is when the Backup is already deployed. In that case the newly configured Primary object cannot be checked in.

To delete a Backup AppEngine from the Galaxy, you must undeploy it first.

There is no redundancy-related configuration required on ApplicationObjects hosted by an AppEngine configured for redundancy. When a system failure occurs, the ApplicationObjects and their attribute values are duplicated on the Standby AppEngine, which becomes the Active AppEngine. For more about failover functionality, see Redundant AppEngines during deployment.

If the Platform hosting the redundant AppEngine is shutdown in OCMC, the AppEngine cannot be flagged as "On failure mark as undeployed" when you undeploy the AppEngine. You see an engine communication error in the **Deploy** dialog box.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Deploy AppEngine objects

Primary and Backup AppEngines can be deployed together or individually.

- When they are deployed together, regardless of which object is actually selected for deployment, the Primary always becomes the Active and the Backup becomes the Standby.
- When they are deployed individually, the first one deployed becomes the Active.

Hosted ApplicationObjects are always deployed to the Active AppEngine. When deploying the first of a redundant pair of AppEngines, you can cascade deploy all objects it hosts. This operation can be paired with deploying both the Primary and Backup AppEngines at the same time.

Important: If you deploy the Backup AppEngine first and then deploy hosted objects to that AppEngine, make sure the network communication to both target computers is good before deploying the Primary AppEngine. Otherwise, errors occur.

In the run-time environment, either the Primary or Backup AppEngine can become the Active or Standby depending upon failure conditions on either computer.

Configuration requirements

Before deploying the Primary and Backup AppEngines, all configuration requirements must be met.

- Each AppEngine must be assigned to a separate WinPlatform.
- A valid redundancy message channel (RMC) must be configured for each WinPlatform.
- To deploy the Primary and Backup together, select **Include Redundant Partner** in the **Deploy** dialog box. This option is not available when doing the following operations:
  - Cascade deploy from the Galaxy
  - Multiple object selection deploy
  - Deploying the WinPlatform that hosts the Primary or Backup AppEngine

The following table shows a matrix of allowed operations based on specific conditions.

Condition	Deploy both primary and backup objects	Cascade deploy allowed
Backup AppEngine's host WinPlatform configured for failover and deployed	Yes	Yes
Backup AppEngine in error state	Yes	Yes
Backup AppEngine's host WinPlatform not deployed	No	Yes
Backup AppEngine's host WinPlatform not configured for failover and deployed	Yes	Yes

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Condition	Deploy both primary and backup objects	Cascade deploy allowed
Deploy from Galaxy node	No	Yes
Deploy from WinPlatform hosting Primary AppEngine	No	Yes
Multiple object selection deploy	No	No
Backup AppEngine's host WinPlatform not configured for failover and not deployed	No	Yes
Deploy from Backup AppEngine	Yes	Yes
Deploy from Primary AppEngine	Yes	Yes

Undeploy AppEngine objects

Undeploying redundant pairs of AppEngines is just like undeploying any regular object.

You can undeploy the Active and Backup AppEngines separately or as a pair.

Important: Undeploying any objects, including redundant AppEngine pairs, does not uninstall code modules for that object from the hosting computer. Code modules are uninstalled only when the WinPlatform is undeployed.

To undeploy as a pair

1. Select one of the objects in an Application view.
2. On the **Object** menu, click **Undeploy**. The **Undeploy** dialog box appears.
3. Select **Include Redundant Partner** in the **Undeploy** dialog box and click **OK**.

Redundant AppEngines during deployment

During initial deployment of a redundant pair of AppEngines, files are deployed in the following order:

- Code modules and other files for the Primary AppEngine are deployed
- Those files for its assigned ApplicationObjects
- All of these files are deployed to the Standby AppEngine by the Active engine’s WinPlatform using the redundancy message channel (RMC)

  Note: If some or all of these files already exist on the Standby AppEngine’s WinPlatform, perhaps, assigned to another AppEngine on that platform, only the delta files are deployed to the Standby AppEngine.

In the configuration environment, objects are always assigned to the Primary AppEngine.

During run time, objects are always deployed to the Active AppEngine whether or not it was initially configured as the Primary object.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

All files are deployed by the Active AppEngine's WinPlatform to the Backup AppEngine using the RMC.

Objects at run time

Objects are always assigned to the Primary AppEngine in the configuration environment.

During run time, objects are always deployed to the Active AppEngine whether or not it was initially configured as the Primary object.

All files are deployed by the Active AppEngine's WinPlatform to the Backup AppEngine as described in the previous section.

Redundant AppEngines during run time

In the run-time environment, the Active and Standby AppEngines first attempt to establish communication across the RMC. This occurs when an AppEngine belonging to a redundant pair first starts. Therefore, if one AppEngine is relocated later to a different WinPlatform, this communication between AppEngines can be reestablished.

During run time, the Active and Standby engines communicate with each other and monitor each other's status.

In the case of a hardware or software failure on the Active computer, the Standby AppEngine becomes the Active one. To move the new Standby AppEngine from its hosting computer, undeploy this AppEngine by selecting the **On failure mark as undeployed** option on the **Undeploy** dialog box. Reassign and redeploy it to a WinPlatform that is configured for redundancy on another computer.

AppEngine redundancy states

Redundant pairs of AppEngines can have one of the following states at a time:

- **Active**: The state of an AppEngine when it has communication with its partner object, its partner is in Standby-Not Ready, Standby-Sync'ing with Active, or Standby-Ready state. A Standby AppEngine transitions into this state when a failover condition is detected. In this state, an AppEngine schedules and runs deployed objects, sends checkpoint data and sends subscriber list updates to the Standby AppEngine.
- **Active - Standby not Available**: The state of an Active AppEngine when it determines it cannot achieve communications with its partner object. This could mean that checkpoint, subscription and alarm state changes are not successfully transmitted to the Standby object because the partner AppEngine is not deployed, number of heartbeats missed from standby objects exceeds the configured max consecutive number of heartbeats missed, or notification is received that the Standby AppEngine shutdown or is not running. If an AppEngine is in this state, it 1) continues normal execution of hosted objects, 2) cannot be manually switched to Standby state, and 3) while continuing to attempt communicate with the Standby, does not attempt to send data to the Standby object.
- **Determining Failover Status**: The initial state of a redundancy-enabled AppEngine when it is first started. It has not determined yet whether it is the Active or Standby AppEngine. Communication between the two AppEngines is attempted first over the RMC and then over the primary network to make this determination. If communication cannot be made after a certain time-out period, an AppEngine assumes the Active role if it has all of the code modules and checkpoint file data to do so. Continued attempts are made at communicating with its partner.
- **Standby - Missed Heartbeats**: The state of an AppEngine when 1) the heartbeat pings are not received from its Active partner through the RMC, but the number of missed beats haven't reached the maximum

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

consecutive number of heartbeats missed, or 2) the heartbeats from active engine have been missed through the primary channel. When in this state, the Standby object attempts to determine whether or not the Active object failed. If a manual failover is started by using the ForceFailoverCmd attribute, it is processed only if the heartbeats were missed over the primary network and not missed over the RMC.

- **Standby - Not Ready**: The state of an AppEngine when one of several conditions occurs: 1) its lost communications with its partner object or it maintains communications with its partner but missed checkpoint updates or alarm state changes from the Active AppEngine, 2) new objects are deployed to the Active AppEngine and necessary files are not installed on the Standby AppEngine yet, or 3) the Standby AppEngine lost communications over the RMC before it completed synchronizing data. Typically, the AppEngine's partner is in one of the following states: Active-Standby not Available, or Active.
- **Standby - Ready**: The state of an AppEngine when it completed synchronizing code modules and checkpoint data with the Active AppEngine. In this state, the AppEngine monitors for Active AppEngine failure by verifying heartbeat pings received from the Active engine and checks that all files required for execution are in sync with the Active engine. It receives the following from the Active AppEngine: checkpoint change data, subscription-related notifications, alarm state changes, and history blocks.
- **Standby - Sync'ng with Active**: The state of an AppEngine when it is synchronizing code modules with the Active object. If code modules exist on the Standby computer that do not exist on the Active node, they are uninstalled, and likewise, any code modules that exist on the Active node but not on the Standby node are installed. After all code modules are synchronized, the AppEngine transitions to Standby-Sync'd Code state.
- **Standby - Syncing Code**: The state of a Standby AppEngine that successfully synchronized all code modules with the Active object.
- **Standby - Syncing Data**: The state of a Standby AppEngine when all object-related data, including checkpoint and subscriber information, are synchronized with the Active object. An object in this state typically transitions to Standby-Ready state.
- **Switching to Active**: A transitional state when a Standby AppEngine is commanded to become Active.
- **Switching to Standby**: A transitional state when an Active AppEngine is commanded to become Standby. When the active engine is switched to standby, the engine will be restarted. This transition state can be switched off by the user changing the attribute of the engine.
- **Failed**: The state of a redundant partner when its process crashes or is terminated by the user. The AppEngine process can be restarted using Operations Control Management Console/PlatformManager.
- **Unknown**: The state of a redundant partner when a communication loss occurs between AppEngines or when the partner AppEngine is stopped, shutdown, or undeployed.

For examples on redundant configuration, see the AVEVA Software Global Customer Support website.

Troubleshoot redundant AppEngines

Most troubleshooting happens in the Operations Control Management Console. For more information about using the Operations Control Management Console, see the *Operations Control Management Console User's Guide*.

Certain requirements are validated by the system infrastructure. For example, the order in which you configure an object pair for redundancy is validated.

The following problems can occur when you are using redundancy:

- You can configure an AppEngine for redundancy before configuring its associated WinPlatform. If you do this,

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

you see an error message that the Platform (specifically, the RMC) is not configured yet.

- If the **RMC IP Address** parameter is not configured in both hosting WinPlatforms, then the configuration state of both Primary and Backup AppEngines changes to Error. You also see a message indicating that the host WinPlatform is not configured with the network adapter required for redundant communications. When the RMC IP Address is configured and the WinPlatforms are checked in, the hosted AppEngines are automatically revalidated and the Error state is resolved. If hosted AppEngines are checked out, they are not revalidated.
- If both Primary and Backup AppEngines are assigned to the same WinPlatform and you try to deploy both engines, both the Primary and Backup fail to deploy. You see a message that the Primary and Backup objects must be hosted by different WinPlatforms. Reassign the Backup object to another WinPlatform and deploy it separately.
- If both the **Network Address** and **RMC IP Address** parameters in the WinPlatform's editor refer to the same network card, you get a warning message when you save the configuration. These parameters must refer to different network cards.
- Before restarting a computer that hosts one of a redundant pair of AppEngines (either the Active or Backup), ensure that the Primary Network is connected. A restart while the Primary Network is disconnected makes the Primary Network bind to the RMC's IP address. An incorrect redundancy state occurs, indicating that redundancy functionality is good. Any time you restart a redundancy-enabled computer, check for proper network connections afterwards. For more information, see Multiple network interface cards.
- When scripting is used to activate attributes, attributes displayed in the watch window may appear to remain in an initializing state after deployment or engine failover. This happens because the script is capturing the attributes while they are still initializing on one of the redundant engines, before the engines are able to synchronize. The attributes will move to active state, but unless there is retry logic in the script, that information will not be returned. This situation can be avoided by ensuring that your scripts contain retry logic to capture the updated state of the attributes.

Generate alarms

When failover conditions occur, Application Server reports alarms to subscribed alarm clients like InTouch.

These alarms contain the following information:

- The name of the AppEngine reporting the alarm.
- The node name of the AppEngine reporting the alarm.
- The state of the AppEngine.
- The node name of the AppEngine's partner object.

Depending on what caused the failover, the Standby AppEngine can become the Active AppEngine in an Off scan state and alarms might not be generated.

If the Active AppEngine is shutdown off scan, the checkpointer can transfer that state to the Standby. When the Standby becomes the Active, it starts off scan. When the AppEngine is put on scan, alarms then are generated.

Reported alarms include the following:

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Alarm	Previous state	Current state	Alarm raised when	Alarm Cleared when	Alarm reported by
Standby Not Ready *	Active	Standby Not Ready	Standby Not Ready	Entering Stacdby Ready	Active Engine
Standby Not Available	Active	Active Standby Not Available	Active Standby Not Available	Entering Active	Active Engine
Failover Occurred			Standby becomes Active	During the next scan of the Active engine	Active Engine

* The Active AppEngine monitors the status of the Standby through the RMC to determine when to raise this alarm. Also, if the Active AppEngine is in Active-Standby not Available state, this alarm is not generated.

When a failover occurs, the Standby AppEngine that becomes active carries alarms outstanding from the old Active AppEngine. The state of those old alarms, though, will change to reflect the new partner's status.

Timestamps are preserved for alarms, including when:

- The alarm was acknowledged
- The alarm condition went true
- The alarm condition went false

Finally, the following information is preserved for alarms:

- An alarm was acknowledged
- The message input by the operator when the alarm was acknowledged

All alarm state information is collected and sent to the Standby AppEngine at the end of a scan cycle and before being sent to alarm clients.

The sequence of reporting alarms ensures that alarm clients do not report alarms in states that are different from those reported by the Standby AppEngine if the Active AppEngine fails.

Generate history

All active objects (AppEngine and hosted objects) report history data as they normally do in the run-time environment.

Historical data is reported to the historian only from the Active AppEngine.

Losing connectivity with the historian does not cause a failover. The Active AppEngine goes into store-forward mode and caches data every 30 seconds. Store-forward data is synchronized with the Standby AppEngine.

When failover conditions occur, no more than 30 seconds of history data is lost in the transition from Standby to Active status. For more information, see the AVEVA Software Global Customer Support website.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Data acquisition redundancy

The RedundantDIObject monitors and controls the redundant DIObject data sources at the object level. Unlike redundant AppEngines, individual DIObject data sources do not have redundancy-related states. For all practical purposes, they function as standalone objects.

Only one DIObject data source provides field device data through the RedundantDIObject at a time. Both data sources must have commonly-configured DAGroups. These must be reflected in and channeled through the RedundantDIObject, which monitors the two DIObject data sources. It also determines which one is Active at any given time. Both data sources must also have the same item address space.

After deployment or engine failover, attributes may appear to remain in an initializing state when scripting is used to activate the attributes. The script may capture the attributes while still initializing, before the engines are able to synchronize with each other. When synchronization occurs, the attributes will move to active state, but unless there is retry logic in the script, that information may not be returned. This situation can be avoided by ensuring that your scripts contain retry logic to capture the updated state of the attributes.

Configure data acquisition redundancy

Data acquisition redundancy objects involve two DIObjects and the RedundantDIObject. In data acquisition redundancy, you must configure all three components:

- Primary DIObject data source
- Backup DIObject data source
- Redundant DIObject data source

Because data acquisition redundant components are essentially standalone objects, all valid operations that apply to any other ApplicationObjects apply to the three objects.

All IDE commands, Galaxy Dump and Load functions, and import and export operations are valid on the two DIObject data sources and the RedundantDIObject.

See the online help associated with each DIObject for help in configuring its Object Editor. Also see the help associated with the RedundantDIObject.

Before you can deploy the RedundantDIObject, you must configure at least one scan group. Also, configure only scan, block read, and block write groups shared by the Primary and Backup DIObjects in the RedundantDIObject.

To configure the Redundant DIObject

1. On the **General** tab of the Object Editor, set the **Primary DI Source** and **Backup DI Source**.
2. Set up the common scan groups.
3. Set up the common block read and block write groups on the tabs of the Object Editor.

Deploy redundant DIObjects

Deployment for data acquisition redundancy is the same as individually deploying unrelated objects. No special conditions apply to each DIObject data source and the RedundantDIObject.

See Deploy for more information about deploying objects.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Data acquisition redundancy during run time

The three objects in the data acquisition redundancy scheme (RedundantDIObject and its two DIObject data sources) work like any other Application Server object when deploying, alarming, and historizing. They have no special redundancy-related states or restrictions.

During run time, the RedundantDIObject monitors the status of the DIObject data sources, and handles the switching from Active to Standby object if failure conditions occur.

RedundantDIObject and PLC connectivity

For the RedundantDIObject, you can use the scan group PingItem attribute to monitor the connection status of the PLC at run time. If you are using the redundancy feature of the RedundantDIObject to communicate with DIObjects, you should configure the PingItem attribute for each scan group.

Change network account utility

The Change Network Account Utility is a tool to manage credentials for node-to-node communications between System Platform components.

During the initial installation of a System Platform component, you must create a new user account or use an existing account. You must provide a user name and a password for the account you select.

You must use the same account on each computer that requires communication with other computers in a System Platform environment. If necessary, you can use the Change Network Account Utility to change your user account after System Platform components have been installed.

Important: When you change user account data with this utility, you can potentially affect communications between the computers with System Platform components installed. All System Platform nodes that require communication with other nodes must use the same account, user name, and password. Use this utility only to standardize user account data on all computers that communicate with each other in a System Platform environment.

Node-to-node communications

All computers that have System Platform software installed must be able to communicate with each other. This communication is enabled through a user account set up during the initial installation of a System Platform component on each computer.

Subsequent installations of System Platform components do not prompt for the user account information. They automatically use the account specified during the installation of the initial System Platform component.

The user account described in this document only enables node-to-node communications between System Platform components. This user account is not associated with Application Server security, which provides user authentication for individual access points to the infrastructure.

Use the change network account utility

Using the Change Network Account Utility, you can change your System Platform network user account,

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

including the following:

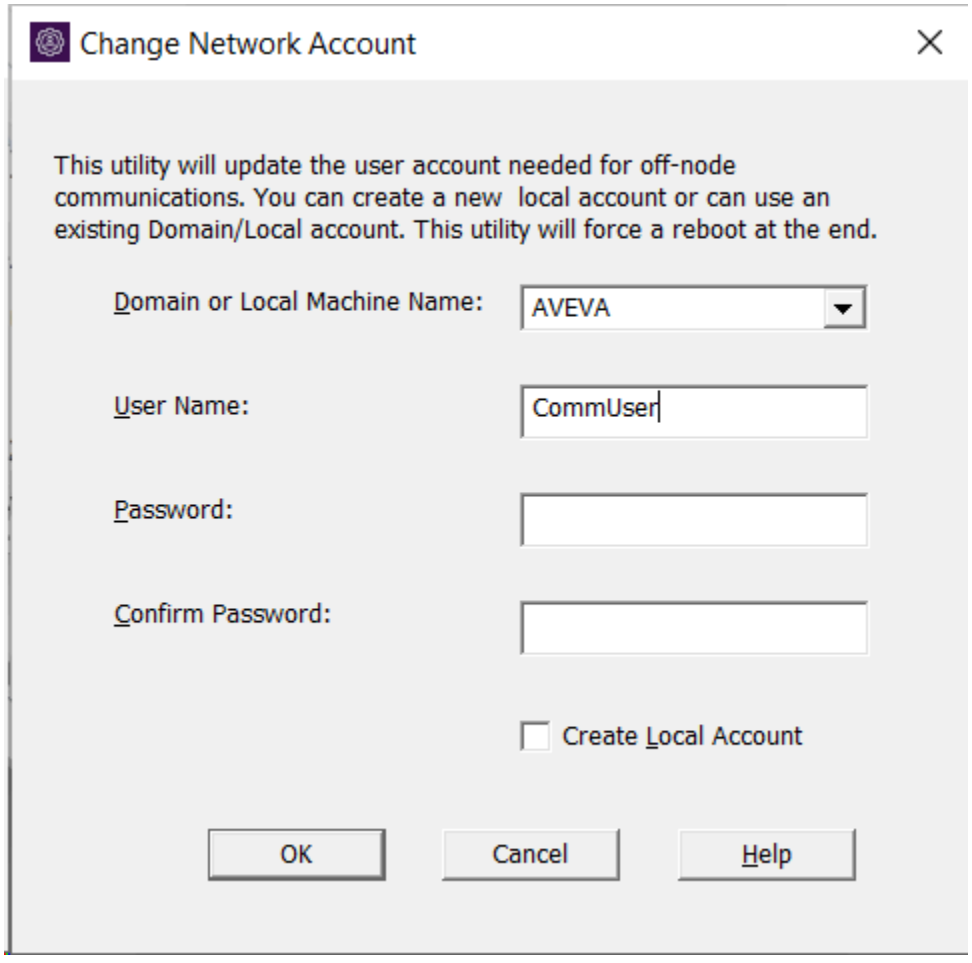
- Changing the password of the current account.
- Creating a new local user account.
- Using another existing local machine or network domain account.

Note: You must log on to the computer with Windows Administrator privileges to make changes with the Change Network Account Utility.

To start the Change Network Account Utility

1. From the Windows **Start** menu, select **Aveva**.
2. Select **Change Network Account**.

   The **Change Network Account** dialog box appears.



The **Change Network Account** dialog box shows the details of the current user account used for System Platform communication. The two password boxes are blank for security reasons.

After you change user account data in the **Change Network Account** dialog box, the Microsoft Windows security component on your computer may take several minutes to update this information on the Application Server Galaxy node. Until that update occurs, communication between nodes in your System Platform environment may be limited. Rebooting the Galaxy node causes the update to occur immediately.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

When you use the Change Network Account Utility to create or use an account that is different from the one originally set up during installation, the old user account is maintained and not deleted.

Note: When you change any data in the Change Network Account dialog box, ensure that all other computers that have System Platform software installed use the same type of user account with the same user name and password.

Change the password of the current account

You can use the Change Network Account utility to change the password of the user account for System Platform communication. If you click **Cancel** before saving your password changes, the user account reverts to the existing password.

The Change Network Account utility reboots the computer after changing the password. You should close all open applications on the computer before changing the password.

To change the password of the current account

1. In the **Password** and **Confirm Password** boxes, type the same new password. When you start the Change Network Account Utility, the **Password** and **Confirm Password** boxes are empty.
2. Click **OK** to save the new password.

   A confirmation dialog box appears and you are prompted to restart your computer. This is a requirement. Close all open applications, if necessary.
3. Click **OK** to reboot the computer.

Create a new local user account

You can use the Change Network Account utility to create a new user account for System Platform communication. If you click **Cancel** before saving your new user account, your current user account remains in effect.

The Change Network Account utility reboots the computer after creating a new user account. You should close all open applications on the computer before creating a new user account.

To create a local user account on your computer

1. Select the **Create Local Account** check box. The **Domain/Local Machine Name** box becomes unavailable.
2. In the **User Name** and **Password** boxes, type the new user name and password.
3. In the **Confirm Password** box, retype the same password.
4. Click **OK** to create the new local user account.

   A confirmation dialog box appears and you are prompted to restart your computer. This is a requirement. Close all open applications, if necessary.
5. Click **OK** to reboot the computer.

Use an existing local machine or network domain account

You can use the Change Network Account utility to change the System Platform communication user account. If you click **Cancel** before saving your user account changes, the current user account remains in effect.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

The Change Network Account utility reboots the computer after changing user accounts. You should close all open applications on the computer before changing user accounts.

Important: We recommend that you do not use a domain account whose password expires periodically or can be changed. Using an account with an expiring password is permissable. But, System Platform node-to-node communication stops if the password expires. If that occurs, then you must use the Change Network Account Utility to update the account data on each computer to re-establish communications.

To use a local machine or network domain account on your computer

1. Clear the **Create Local Account** check box. The **Domain or Local Machine Name** box becomes available.
2. In the **Domain or Local Machine Name** box, type a valid network domain name or select the local machine name from the list. If you do not know the valid network domain name, ask your network administrator.
3. In the **User Name** and **Password** boxes, type valid data for the selected user account.
4. In the **Confirm Password** box, retype the password.
5. Click **OK** to use the selected user account.

   The utility verifies the existence of the user account you typed and prompts you if any adverse conditions require your attention. Respond appropriately.

   A confirmation dialog box appears and you are prompted to restart your computer. This is a requirement. Close all open applications, if necessary.
6. Click **OK** to reboot the computer.

Error messages

When using the Change Network Account Utility, you may encounter the following error messages.

Error message	Action required
Administrative privileges are required to run this utility.	You must log on to the computer with Administrator privileges on the computer to run the Change Network Account Utility. Click **OK**, log on as a user with Administrator rights, and start the Change Network Account Utility again.
The Password was not correctly confirmed. Please ensure that the Password and Confirmation match exactly.	You mistyped either the password or confirmation password in the dialog box. These two passwords must match to ensure the accuracy of user account data. Click **OK**, retype both password and confirmation password text, and then click **OK** again.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Error message	Action required
Password can not be empty. Please enter a valid password.	You must type a password in the Change Network Account Utility. Click **OK**, type a password in the dialog box, type the same password in the Confirm Password box, and then click **OK** again.
User account 'a' already exists. Please enter another user name.	You chose to create a local account by selecting **Create Local Account** in the Change Network Account Utility. This account already exists. Click **OK**, type a new user name in the Change Network Account Utility, and click **OK**.
Domain/Local Machine name can not be empty.	You must enter an existing local machine or network domain account. Click **OK,** either type a valid network domain name or select the local machine name from the list, and click **OK**. If you do not know the valid network domain name, ask your network administrator.
User account 'a' does not exist. Please enter another user name.	You chose a local machine or network domain account that does not exist. Click **OK**. Ensure the user account you entered in the **Domain/Local Machine** Name box is valid and the user name and password you typed are valid for that user account, and click **OK**. If this message is displayed again, check with your network administrator about the existence of the account.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Error message	Action required
The password policy for this account lets the password expire and be changed. However, the password must not expire or should not be changed for the System Platform products to function properly. Do you want to use this account?	You chose a local machine or network domain user account whose password can be changed or expire. Click **OK** if you want to use this account or **Cancel** to return to the Change Network Account Utility. Then, you can choose to use or create another user account. We recommend that you do not use a domain account whose password expires periodically or can be changed. Using such an account is allowed, but after the expiration date or when the password is changed, System Platform node-to-node communications stops. You then must use the Change Network Account Utility on each computer to update the account data to re-establish communications.
Do you want to update user account 'a' to use this new password?	You chose to use a local machine user account, but the password you typed does not match the account's password. Click **OK** if you want to change the password for this user account or **Cancel** to revert to the original settings. If you click **OK**, the password for the account is changed. If you click **Cancel**, the Change Network Account Utility is displayed, allowing you to correct the password or type another user name and password.
Invalid Domain name 'dom'. Please enter a valid domain name.	You typed an invalid network domain name in the **Domain/Local Machine Name** box. Click **OK**, retype the domain name, and click **OK**.
Invalid Password for user account 'a'. Please enter the correct password.	You chose to use a network domain user account, but the password you typed does not match the account's password. Click **OK**, correct the password, confirm the password, and click **OK**.
The system will now reboot. Please close all your open applications and click **OK** when you are done.	You changed account information in the Change Network Account Utility. The computer must be restarted to implement those changes. Close the open applications, and then click **OK**.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Error message	Action required
User name can not contain these characters: " / \\ [ ] : ; | , + * ? < > Please enter a valid user name.	You typed one or more invalid characters in the **User Name** box. Click **OK**, type a valid user name, and then click **OK**.
A user name can not consist solely of periods [.] or spaces.	You must type a valid user name in the Change Network Account Utility. Click **OK**, type a user name in the dialog box, and click **OK** again.

Select a language

You can change the language of the **Change Network Account** dialog box.

To change the language of the Change Network Account dialog box

1. Click the icon on the title bar of the **Change Network Account** dialog box.
2. Click **Select Language**.
3. On the submenu, choose from the list of languages available.

   A message appears and informs you that the Change Network Account utility must be restarted for the new language to be displayed.
4. Click **OK** to close the message.
5. Click **Cancel** to close the Change Network Account utility.

   The language you selected appears in the **Change Network Account** dialog box the next time you start it.

Determine the change network account utility version

To see the version of the Change Network Account Utility, click the icon on the title bar and select **About Change Network Account**.

A message box appears, showing the version of the Change Network Account utility.

Galaxy Database Manager

The Galaxy Database Manager is a System Platform utility you can use to manage your Galaxies. Use it to back up your Galaxies so that if a Galaxy becomes corrupted, you can restore the Galaxy. You can also use a backup to reproduce a Galaxy on another computer.

The Galaxy Database Manager is automatically installed on every Galaxy Repository node.

Operations control management console

The Galaxy Database Manager is part of the System Platform Operations Control Management Console (OCMC) suite of utilities you can use to manage the System Platform environment. The OCMC is a snap-in to the

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Microsoft Management Console (MMC).

The MMC and each OCMC utility provides its own documentation. For more information, click **Help** on the **Action** menu. The first Help book shown (Microsoft Management Console) provides information about the MMC utility in general. Each OCMC utility is then represented by a Help book below the MMC book.

This document describes the functionality and user interface of the Galaxy Database Manager. It describes common features of the MMC only if required to explain Galaxy Database Manager functionality.

Note: Aspects of the Galaxy Database Manager user interface may look different when installed on different versions of Microsoft Windows. Also, the version of MMC installed on your computer affects the content and behavior of the user interface.

Start the galaxy database manager

To start the Galaxy Database Manager utility

1. From the **Start** menu:
   - Either type **Operations Control Management Console** in the search bar, or
   - Navigate to the AVEVA Start folder and locate **Operations Control Management Console** app.
2. Open the **Operations Control Management Console.**
3. Locate the Galaxy Database Manager in the console tree hierarchy.

   Note: You must have Administrator privileges on the Galaxy Repository computer to use the Galaxy Database Manager. You may need additional privileges if security for the Galaxy database is set up through Microsoft SQL Server. See SQL Server documentation for further instructions.

This Galaxy Database Manager utility provides a view of all Galaxies in your Galaxy Repository. Select the Galaxy Database Manager in the console hierarchy to see a list of Galaxies in the details pane of the MMC.

Navigate in the galaxy manager

From the OCMC console, you can access drop-down menus, context menus, and dialog boxes for backing up and restoring your Galaxies.

The contents of the title bar, menu bar, toolbar, and status bar are dynamic depending on which item is selected in the console tree or details pane. The contents of the details pane also change as you select different items in the console tree hierarchy.

The title bar shows the item selected in the console tree hierarchy. The status bar either shows the condition, if any, of the item selected in the console tree or provides a brief description of a highlighted command in a menu.

Console tree

The console tree is a hierarchical structure beginning at the Operations Control System Management Console (OCMC) root. All OCMC utilities are shown next in the hierarchy, including the Galaxy Database Manager. All Galaxies in your Galaxy Repository are shown on the next level of the hierarchy.

The contents of the details pane change as you select items in the console tree.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Details pane

The details pane is composed of two main elements:

- Column headings: when Galaxy Database Manager is selected in the console tree, two columns are shown, **Node** and **Galaxy**. The **Node** column identifies which node a Galaxy resides on. The **Galaxy** column displays the Galaxy's name. Use this information when backing up and restoring Galaxies.
- Data window: shows Galaxy data.

The contents of the details pane change as you select items in the console tree. See MMC Help for information about managing column headings.

Use the galaxy database manager

You use the Galaxy Database Manager to back up and restore a Galaxy.

Backing up a Galaxy creates a single backup file (.cab extension) containing all the files, configuration data, and object deployment states required to recreate the Galaxy in an empty Galaxy Repository.

During the backup, no write operations are allowed to the Galaxy Repository. If write activity is occurring, you should back up at a later time.

Restoring a Galaxy uses the backup file to overwrite an existing Galaxy or to create the backed up Galaxy in a different Galaxy Repository. The restore process prompts you for confirmation before a Galaxy is overwritten.

All objects should be undeployed before beginning to restore a Galaxy. During the restore, no clients can use the Galaxy Repository. If these conditions are not acceptable, you should restore at a later time.

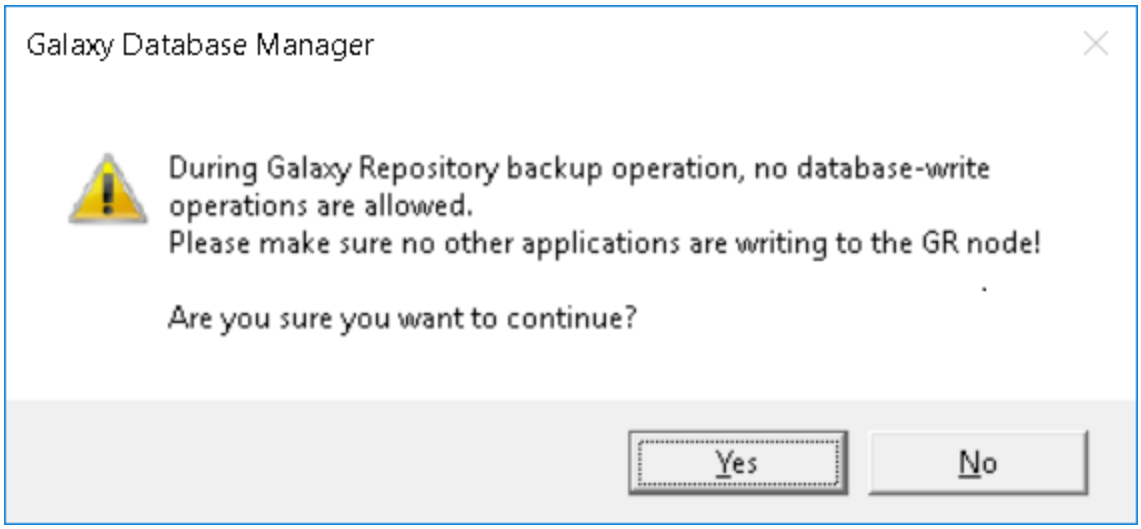
Back up a galaxy

The backup function of the Galaxy Database Manager archives all files and configuration data required to recreate the selected Galaxy in an empty Galaxy Repository.

To back up a Galaxy

1. In the console tree of the Operations Control Management Console, select **Galaxy Database Manager**. The details pane shows all Galaxies and the nodes they reside on.
2. In the details pane, select the Galaxy you want to back up.
3. On the **Action** menu, click **Backup**.
4. Read the warning message.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.



5. Click **Yes**.
6. In the **Save In** dialog box, enter the path and file name (.cab extension) of the backup file you want to create. Use the browse button to find a file or a location.
7. Click **Save**. To terminate the backup, click **Cancel**.
8. When the backup function is finished, click **Close**.

Restore a galaxy

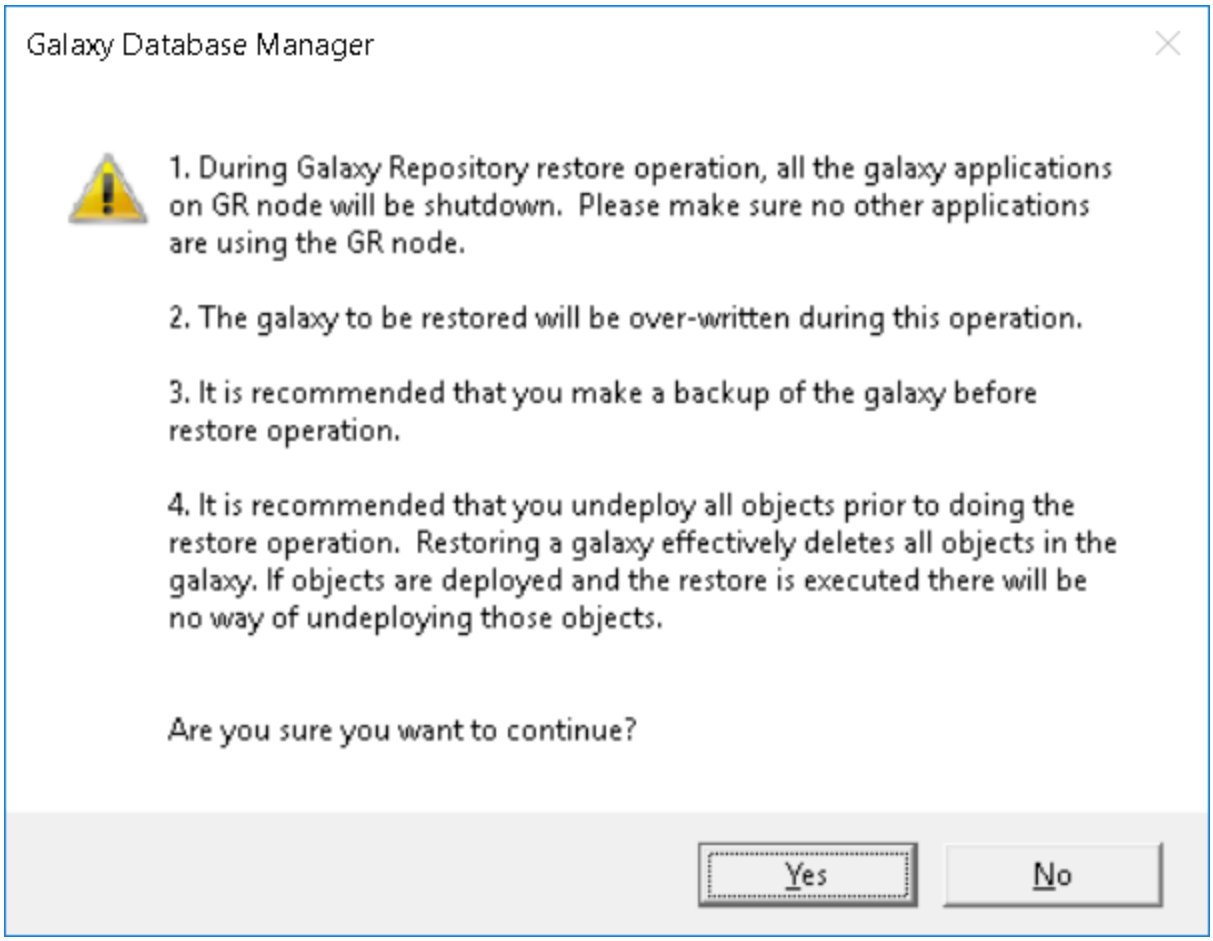
You can choose a backup file to overwrite an existing, corrupted Galaxy or to reproduce a Galaxy in another Galaxy Repository.

If you are reproducing a new Galaxy on another Galaxy Repository with the restore function, first use the System Platform IDE to create the new Galaxy in the target Galaxy Repository. The names of the new Galaxy you create in the IDE and the backed up Galaxy you use in the restore process must be the same. See the IDE documentation for more information about creating Galaxies.

To restore a corrupted Galaxy or to reproduce a Galaxy in another Galaxy Repository

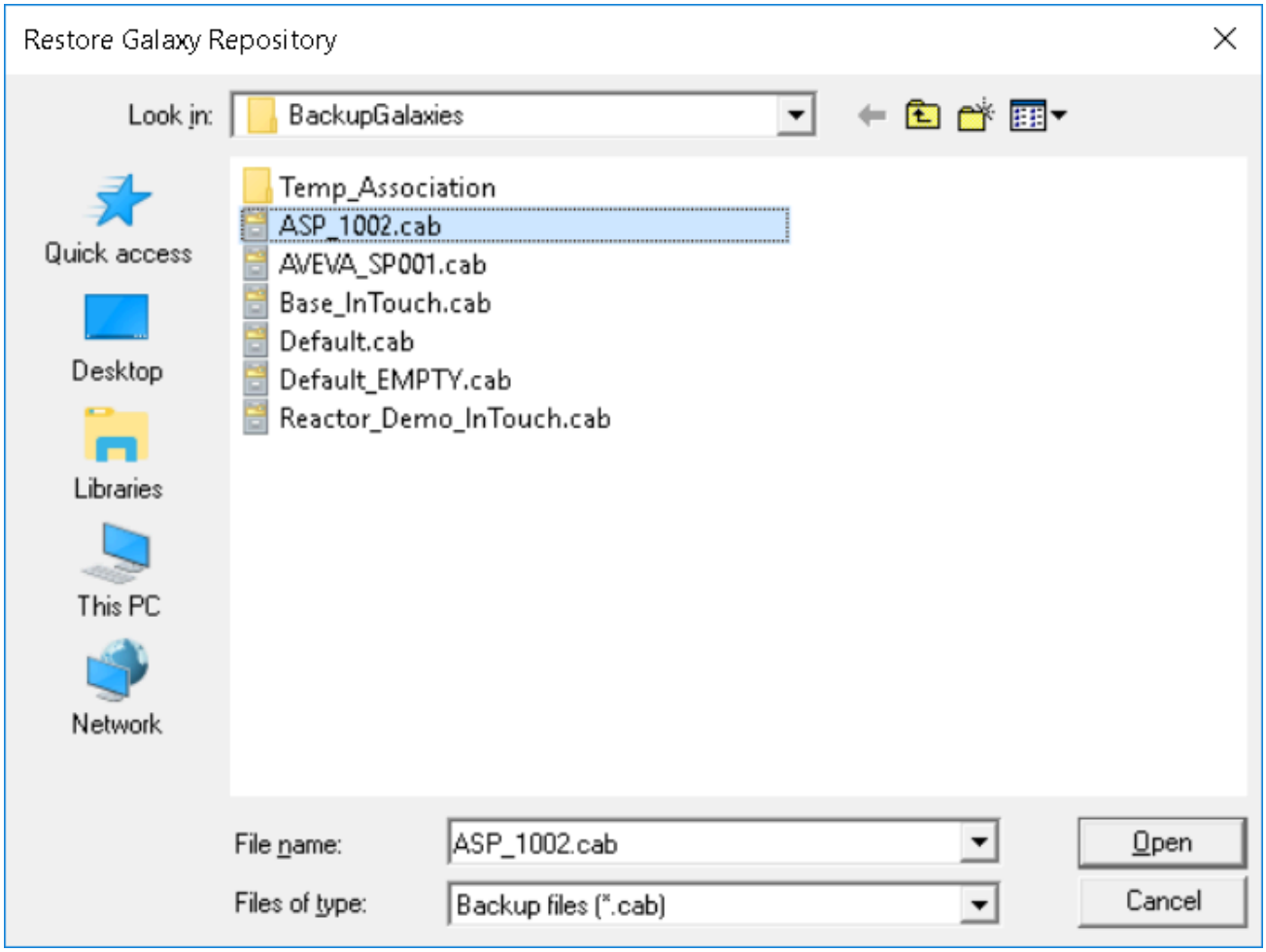
1. In the console tree of the Operations Control Management Console, click **Galaxy Database Manager**. The details pane shows all Galaxies and the nodes they reside on.
2. In the details pane, select the corrupted Galaxy you want to restore or the name of the Galaxy you want to reproduce.
3. On the **Action** menu, click **Restore**.
4. Read the warning messages. Click **Yes** to continue restoring and **No** to terminate the restore function.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.



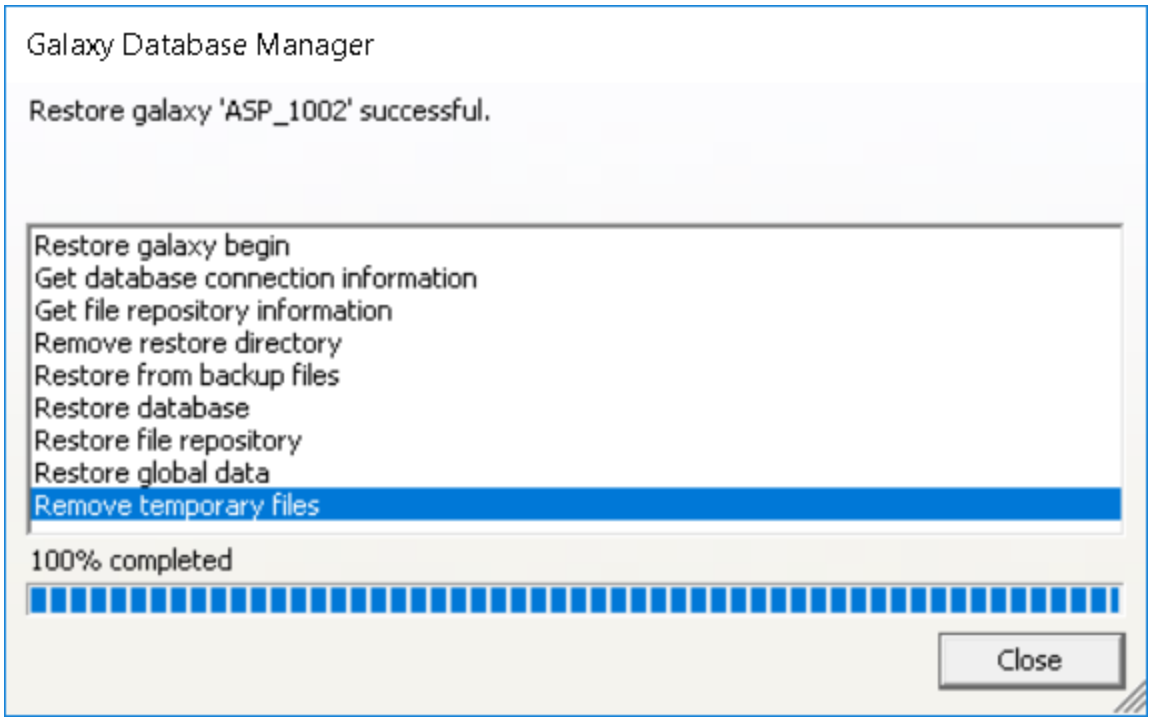
5. Navigate to the path and file name (.cab extension) of the backup file you want to use. Either type the path and file name or use the browse button to explore for a file or location. The default path is the last folder used for a backup or restore.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.



6. To continue, click **Open**. If you have an active client connected to the Galaxy Repository, a message appears. You are required to quit those client programs before restore can continue. To terminate the restore function, click **Cancel**.
7. When the restore function is finished, click **Close**.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.



Determine the galaxy database manager version

You can view the version, copyright and other information about the utility.

To view galaxy database manager information

1. Select **Galaxy Database Manager** in the Operations Control Management Console hierarchy.
2. On the **Action** menu, click **About Galaxy Database Manager**.
3. Click **OK** to close the window.

Common Platform Services

Common Platform services are used to set various parameters that control license behavior, user management, and authentication. There are three components to be configured:

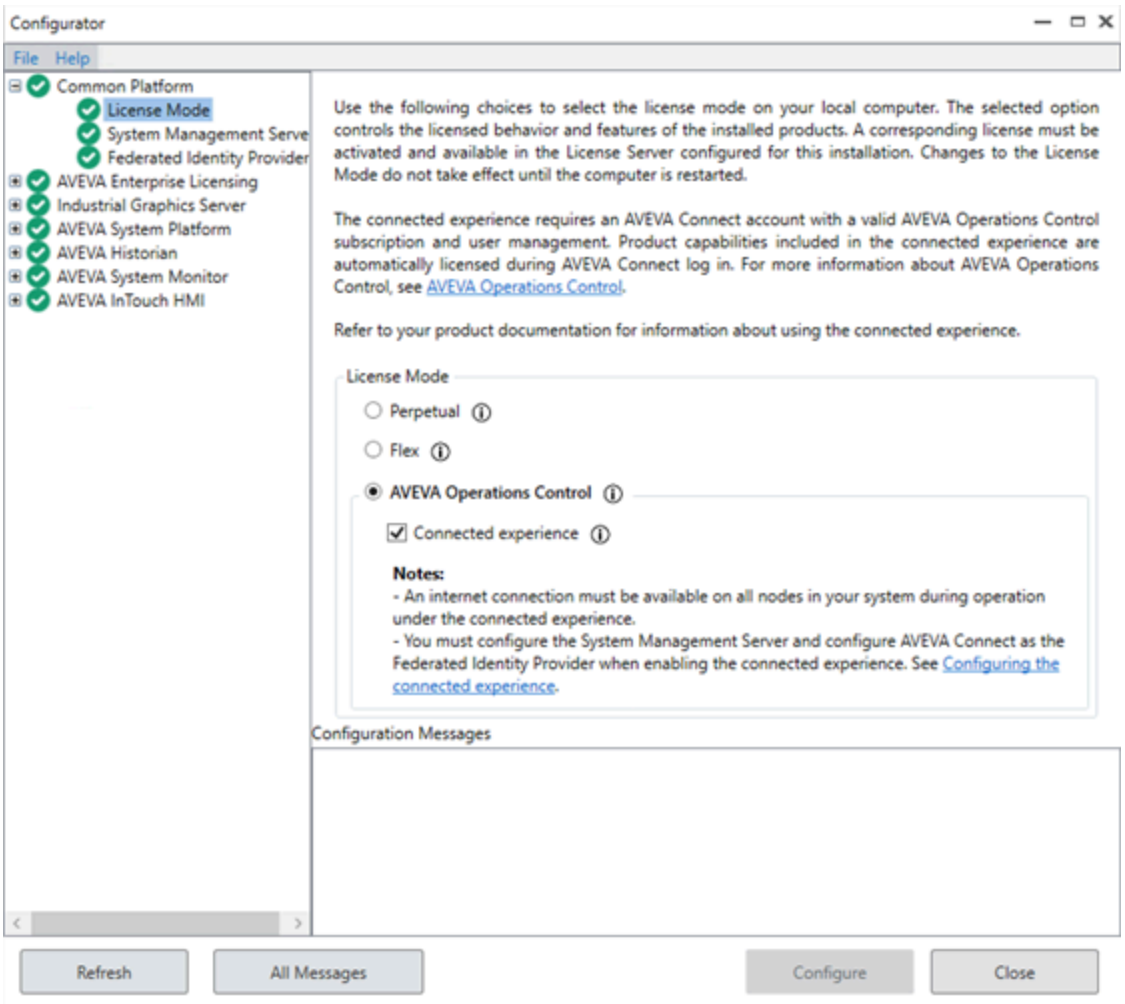
Common Platform services include the following components:

- License Mode
- System Management Server
- Federated Identity Provider

License Mode

You can configure the **License Mode** in the **Configurator** under **Common Platform** tab.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.



The following **License Mode** options are available:

- **Perpetual:** A specific AVEVA product license purchased for use in perpetuity.
- **Flex:** Choose subscription license separately for a range of AVEVA products. Purchase Flex credits to license use of any AVEVA cloud, hybrid or on-premises products for a recurring period.
- **AVEVA Operations Control:** A subscription license for at least one of two AVEVA Operations Control packages (Edge, Supervisory); includes unlimited use of all products in the product package for your defined set of users.
  - **connected experience:** Select to enable a Single Sign-on (SSO) experience across all Operations Control products for that node with AVEVA Connect cloud capabilities.
    - The **connected experience** requires an AVEVA Connect account with a valid Operations Control subscription and user management.
    - Selecting the **connected experience** option enables the behavior of all Operations Control products on that node to require log in authentication with AVEVA Connect when starting the first product on the node. Products on the node subsequently launched will authenticate using SSO. AVEVA Connect-based authorization is the only security mode available under the **connected experience**.
    - The **connected experience** must be enabled on all nodes in your system. Applications previously build on nodes not enabled for the connected experience must be reconfigured to function in the **connected experience** environment.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

- You can deselect the **connected experience** at any time, but the **v** must be disabled on all nodes in your system. Applications built under the **connected experience** must be reconfigured to function under a non-Connected Experience environment including both authentication methods and product license.

System Management Server

AVEVA software products need post-installation configuration in order to use encrypted communications. You need to configure your products using the Configurator after you have installed them. The Configurator lists all product components that require post-installation configuration.

Note: Configuring a System Management Server (SMS) is highly recommended to ensure the security of your System Platform. It is required when redundancy is enabled for Application Server nodes.

System Management Server overview

The System Management Server (SMS) allows for encrypted communication between machines. Encrypted communications can be used when a trust relationship between one or more machines running AVEVA products is established. This is achieved through the System Management Server by utilizing certificates.

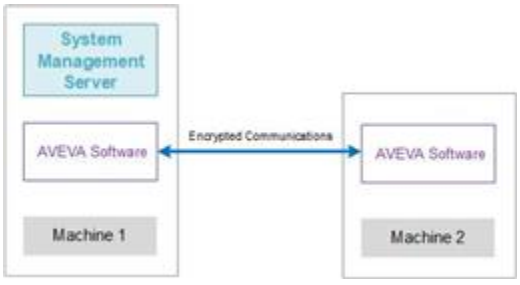
Only one of the machines in the network is identified and configured as a System Management Server. Machines running AVEVA products can then connect to that single System Management Server to establish trust and configure encrypted communications.

Note: To connect to the System Management Server, you need to be a member of either the "aaAdministrators" or the "Administrators" group on the machine where the System Management Server is installed.

Install System Management Server

Regardless of the size of your system setup, only one System Management Server is required for encrypted communication. System Management Server can be installed on one of the machines running an AVEVA application or on a separate machine on the network.

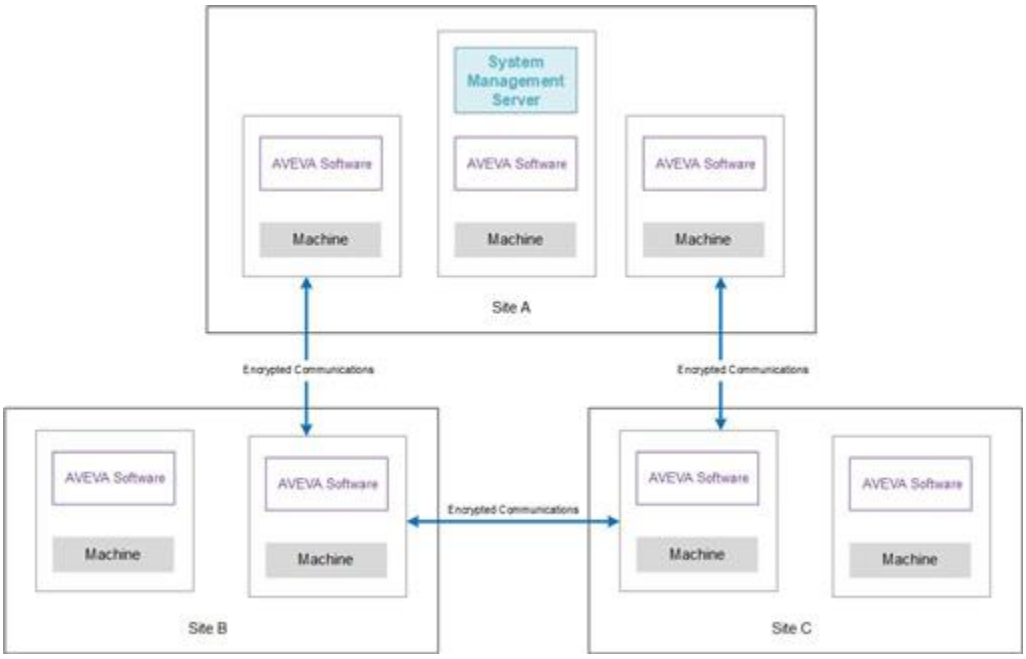
Note: You need to include only one System Management Server in your entire system. If other AVEVA products are installed, confirm that System Management Server has not been configured elsewhere before proceeding, as communication disruptions may occur.



System Management Server can also be installed in a large, multi-site environment running multiple AVEVA products. In such systems, the location of the System Management Server may be governed by one or more products. However, all AVEVA products should be able to connect to the System Management Server so that certificates can be renewed when required.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

An example configuration of a multi-site system is shown below:



Redundant SSO server

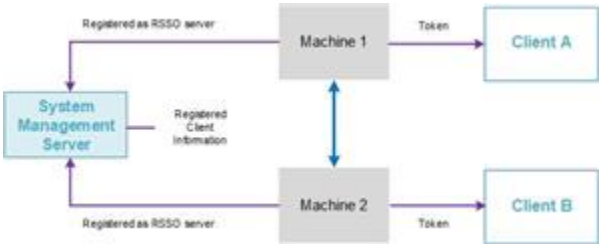
You can configure a machine that connects to an existing System Management Server as a Redundant Single Sign-On (RSSO) server. When you select the System Management Server and configure the current machine, the SSO capability from the System Management Server is shared with the RSSO server.

The purpose of setting up RSSO servers is to:

- distribute the workload between RSSO servers
- eliminate single point of failure

Note: Not all AVEVA products make use of the Redundant SSO Server functionality. Refer to your product documentation to see if this feature is supported.

The following diagram illustrates the working of RSSO servers:



A brief description of the steps in the above workflow is given below:

1. Machine 1 and Machine 2 connect to an existing System Management Server and are configured as RSSO servers using the Configurator.
2. Client A and Client B are registered with the System Management Server.
3. When a workflow is initiated on Client A, it requests a token from Machine 1.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

4. Machine 1 sends a token to Client A as if it were sent from the System Management Server.
5. When a workflow is initiated on Client B, it requests a token from Machine 2.
6. Machine 2 sends a token to Client B as if it were sent from the System Management Server.

Note: The client needs to be configured manually to select the RSSO server with which it will communicate for obtaining a token.

The main difference between the System Management Server and the Redundant Single Sign-On (RSSO) server is that a client can register only with the System Management Server, not with the RSSO server. If you configure RSSO servers, it is recommended that the clients communicate with an RSSO server to obtain a token.

A workflow is initiated and completed on a single RSSO server; it cannot be split between RSSO servers. Subsequent client requests for a token should be made to the RSSO server that issued the original token. In addition, token renewal is also possible only with the same RSSO server.

If the original RSSO becomes unavailable, a new token needs to be requested from another, available RSSO server.

An RSSO server can run independently without the System Management Server, provided that the latest client / resource configurations have already been synchronized.

Note: Configuration such as client registration can only be made with the SMS. RSSO does not accept configuration requests.

Work with Configurator

You can complete the following tasks in the Configurator:

- Configure the System Management Server
- Connect to a System Management Server

In addition to the above tasks, you can configure **Advanced** settings.

Note: If you have other products installed that use the Configurator, the Configurator may be located in a different program group.
If System Management Server is not configured, capabilities like connected experience, Web OMI, Azure AD/ AVEVA Connect federation will not be available.

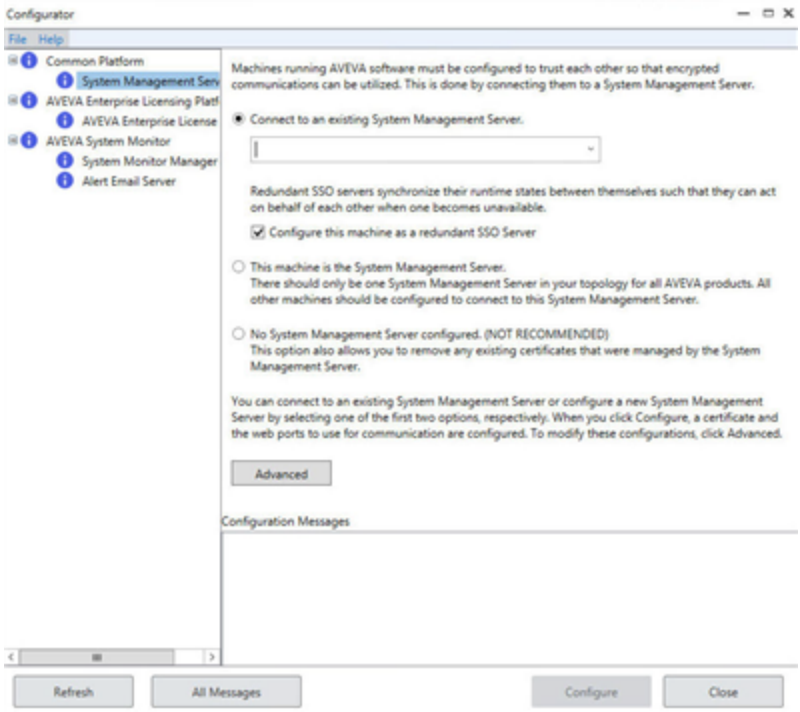
Connect a machine to a System Management Server

Machines running AVEVA products need to connect to a System Management Server via a configured certificate in order to use encrypted communication.

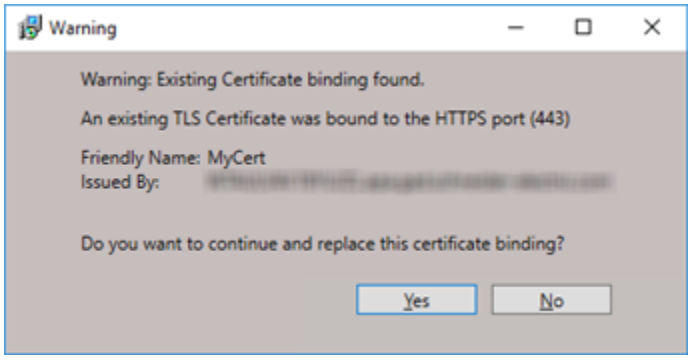
To connect a machine to a System Management Server

1. Start the Configurator on the machine you wish to connect. The Configurator screen is displayed.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.



2. In the left pane, select **Common Platform > System Management Server**.
3. If you wish to connect to an already existing System Management Server, select the **Connect to an existing System Management Server**.
4. From the list of System Management Server available, select the required System Management Server. Note that the list displays all machines on the network that have been configured to function as System Management Server. In most cases, there is only one System Management Server.
5. If you wish to configure a machine as a Redundant SSO (RSSO) server, under **Connect to an existing System Management Server,** select **Configure this machine as a Redundant SSO Server.**
6. Select **Configure**. The System Management Server configurator verifies the configuration, and if any conflicting certificate configuration or communications port binding is detected, the following **Warning** message is displayed.

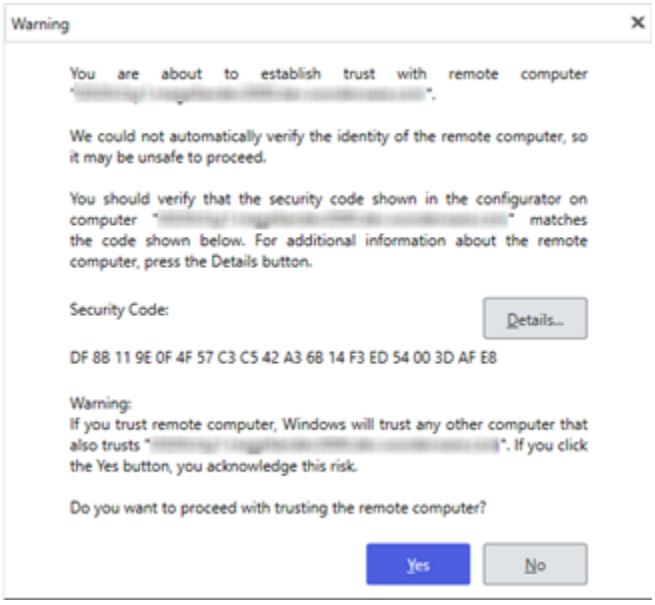


7. Selecting **No** results in the following message being displayed in the **Configuration Messages** area, and the machine will not be connected to the System Management Server.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

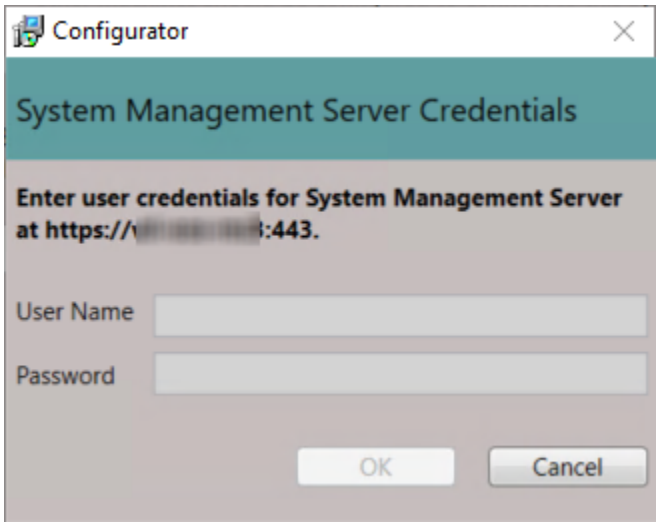


8. Selecting **Yes** replaces the binding. By default, the root certificate is downloaded from the System Management Server. This is possible only when the **Automatically Generated** certificate option is selected on the **Advanced** page. The following message is displayed.



9. Review the message carefully before you select **Yes**. Selecting **No** cancels the configuration process.
10. Select **Details**, to view more information about the certificate.

    If you are not a member of the "aaAdministrators" or "Administrators" group on the System Management Server, a dialog box prompting you to log on to the System Management Server with administrative credentials is displayed. Enter the credentials of a user that is a member of the "aaAdministrators" or "Administrators" group on the System Management Server and select **OK**.



Note: If you have configured a communications proxy on the server where the System Management Server is installed, contact the AVEVA Global Customer Support team for information about installing and connecting

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

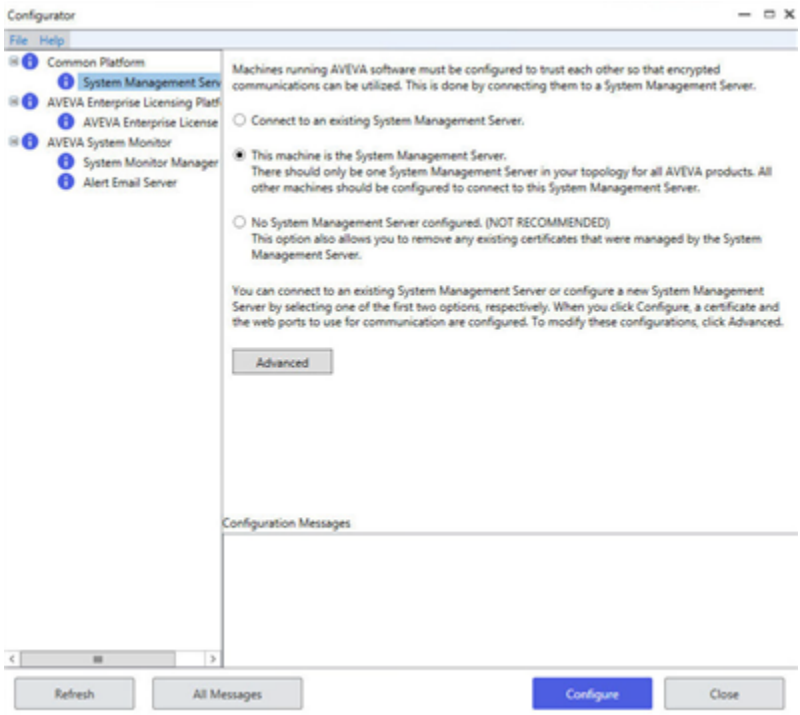
to the System Management Server.

11. The **Configuration Messages** area displays the steps in the configuration process and the progress. If the configuration is unsuccessful, you can view details of the errors in System Management Console.
12. Select **Close** to exit the Configurator.

Configure the System Management Server

To configure the System Management Server

1. Start the Configurator. The Configurator screen is displayed.

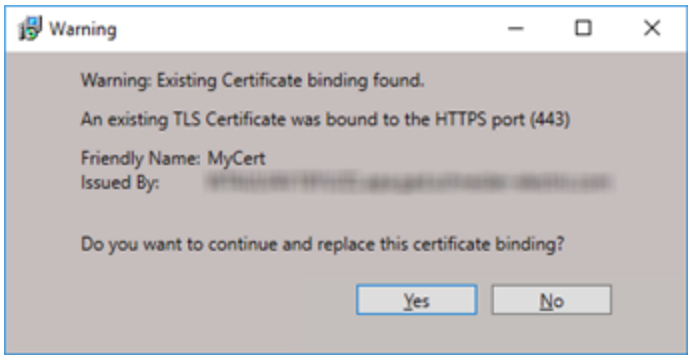


2. In the left pane, select **Common Platform > System Management Server**.
3. Select **This machine is the System Management Server**. Review the notes on the screen before you start the configuration.

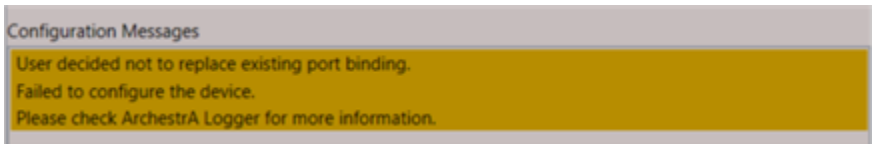
   Note: You should include only one in your entire system. If other AVEVA products are installed, make sure that a has not been configured elsewhere before proceeding as communication disruptions may occur.
4. Select **Configure**.
5. Select **Yes** for the **Warning** message after you confirm there is only one System Management Server in your entire system. If other AVEVA products are installed, make sure that a has not been configured elsewhere before proceeding as communication disruptions may occur.

   The System Management Server configurator verifies the configuration, and if any conflicting certificate or communications port binding is detected, the following **Warning** message is displayed.

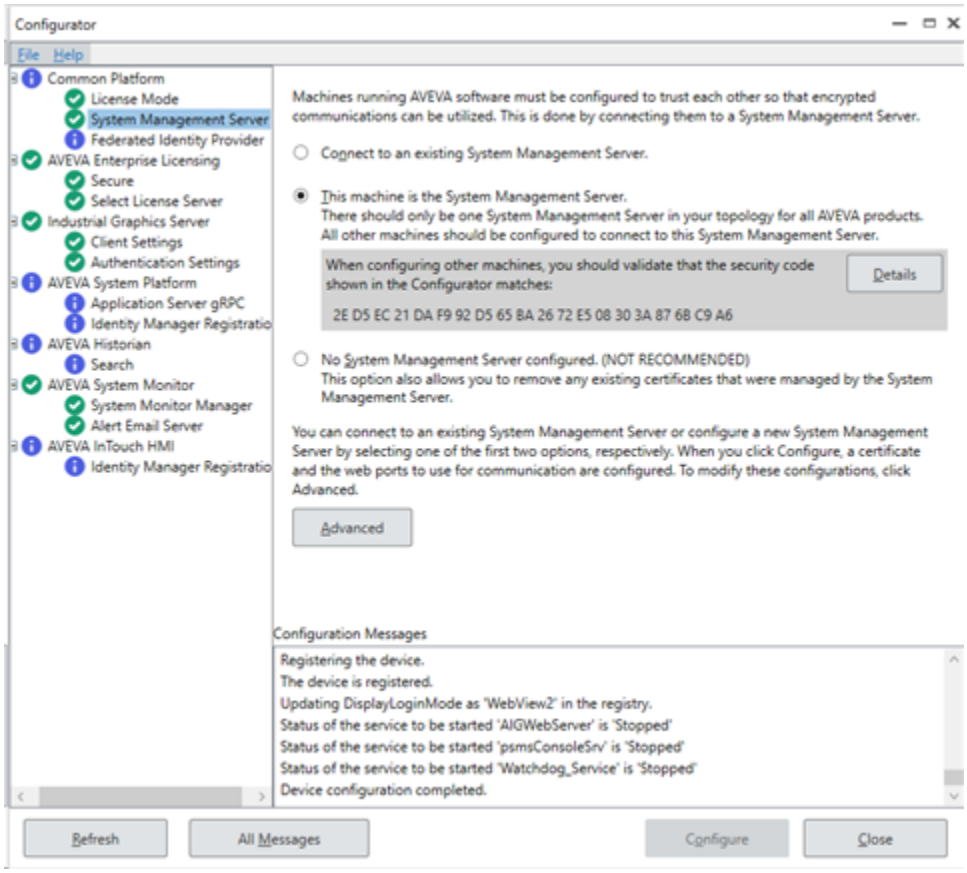
© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.



6. Selecting **No** results in the following message being displayed in the **Configuration Messages** area, and the machine will not be connected to the System Management Server.



7. Selecting **Yes** replaces the binding. The Configurator will start configuring the System Management Server.
8. On successful configuration, the message **Device configuration completed** is displayed. The security code is displayed in the Configurator as shown below. It is recommended that you make a note of the security code because you will need to verify the security code when you add other machines to this System Management Console.
9. Select **Details,** to view more information about the certificate.



If the configuration is unsuccessful, you can view details of the errors in the System Management Server.

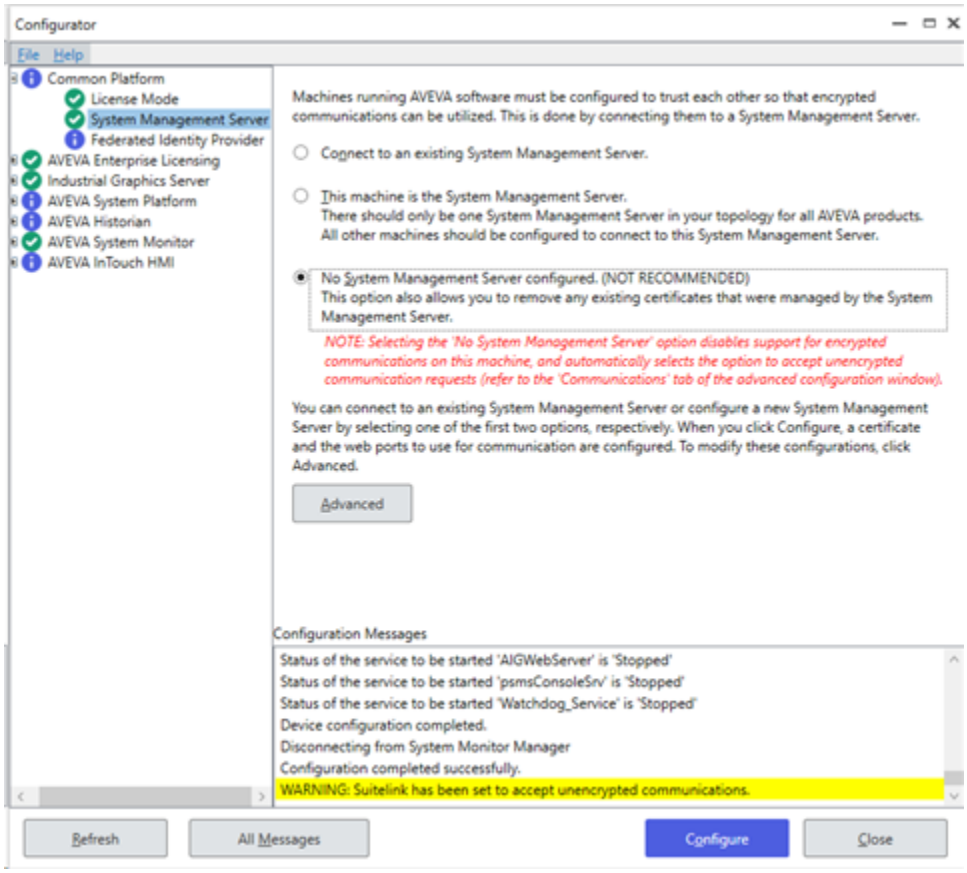
© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

10. Select **Close** to exit the Configurator.

Run products without a System Management Server

To run AVEVA products without a System Management Server configured

1. Start the Configurator.
2. In the left pane, Select **Common Platform > System Management Server**.
3. Select **No System Management Server configured**. Selecting this option results in each individual AVEVA product managing their secure communications, which may or may not be available without the System Management Server.



Note: This procedure is not recommended.

Advanced Configuration

If you have already configured a System Management Server or have selected a System Management Server to connect to, the configuration may be modified if it is required.

To modify an existing configuration

- Select **Advanced**. The **Advanced Configuration** dialog is displayed.

The **Advanced Configuration** window consists of the following tabs:

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

- Certificate: To configure the certificate for secure communications.
- Ports: To configuring the http and https communication ports.
- Communications: To enable or disable the ability to use a non-encrypted channel for SuiteLink communications, and limiting which users have access to NMX communications.
- Authentication: To enable browser to authenticate user.

Certificates tab

System Management Server uses a security certificate to ensure that communication between nodes is encrypted. The certificates tab includes the following configurable fields:

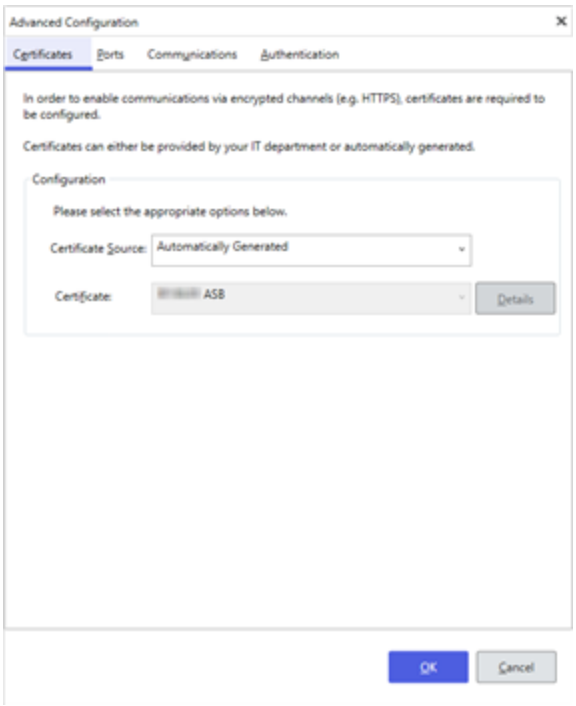
- Certificate Source: Select either **Automatically Generated** (default), or **Provided by IT**. If your IT department is providing the certificate, select **Import** and navigate to the certificate file.
- Certificate: The certificate name is displayed. If you imported a certificate, select Details to know more. The certificate is periodically renewed through an automatic update process, both on the server node and on remote nodes.
- System Management Server: If you are connecting to an existing System Management Server, the name and port number of the server you selected is shown.

To configure the certificate for secure communications

1. If you want the Configurator to generate a certificate, select **Automatically Generated** from the **Certificate Source** list. By default, automatically generated is selected. The **Certificate** field is disabled if automatically generated option is selected, or if certificates generated earlier have been deleted. To view more information about the certificate, select **Details**.

   Note: Automatically generated certificates are renewed automatically.
2. If you want to use a certificate generated by your IT Department, select **Provided by IT (import/select)** from the **Certificate Source** list.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.



3. From the **Certificate** list, select the certificate you want to use.
4. Select **Import** to use a certificate not listed in the **Certificate** list. The **Import Certificate** dialog is displayed.



5. In the **Certificate file** field, browse to select a certificate.
6. From the **Certificate Store** list, select the type of certificate – **Root**, **Intermediate**, or **Personal**. The certificate is configured to be used for encrypting communication channels.

   Depending upon the type selected here, the certificate is stored in the **Certificate Store** identified by the certificate type.
7. In the **Password** field, type the password for the selected certificate Store. The Certificate Store does not have a password. This is the optional password for the certificate being imported.
8. Select **OK** to save the details and close the **Import Certificate** dialog.

Note: The IT Department needs to renew certificates they generate as and when required.

9. Select **Details**, to view the details of the certificate.

   The **System Management Server** field is displayed only if you have selected the **Connect to an existing System Management Server** option in the previous screen. You can use this field to change to a different System Management Server. From the **System Management Server** list, select the machine on which you want the certificate to be generated.

Note: You can only specify a computer name or a fully-qualified domain name for the System Management

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Server. It is recommended that you always use a fully-qualified domain name to identify the SMS. Specifying the name in a different format, for example an IP address, may result in errors.

Ports tab

The System Management Server uses HTTP and HTTPS for communications with AVEVA software. Remote nodes must be configured with the same port numbers as configured here. By default, the System Management Server uses HTTP port 80 and HTTPS port 443. Generally, you can use the default settings. Select the **Advanced** button, then select the Ports tab and edit the port numbers as needed.

To configure the HTTP and HTTPS communication ports

This is the port number on which the is configured and is automatically populated. It is recommended that you also check the port number manually.

1. Select the **HTTP Port** and **HTTPS Port**. The defaults are 80 and 443, respectively.



These ports are local ports on the current machine, which are used by web services and clients connecting to this machine.

Note: If you change the default ports you need to
(i) restart the machine for the change to take effect, and
(ii) update the port number(s) on all client products that point to servers running on this machine.
This excludes the System Management Server because it discovers port numbers automatically.

HTTP and HTTPS ports range from 0 to 65535. Within this range, you may choose any port that has not been blocked or is currently in use. Otherwise, you will receive "Port number conflict" error.

2. Select **OK** to save your settings. The Configurator's main screen is displayed.
3. Select **Configure**.

   The **Configuration Messages** area displays the steps in the configuration process and the progress. On successful configuration, the Certificate is generated on the local machine and signed by the selected System

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

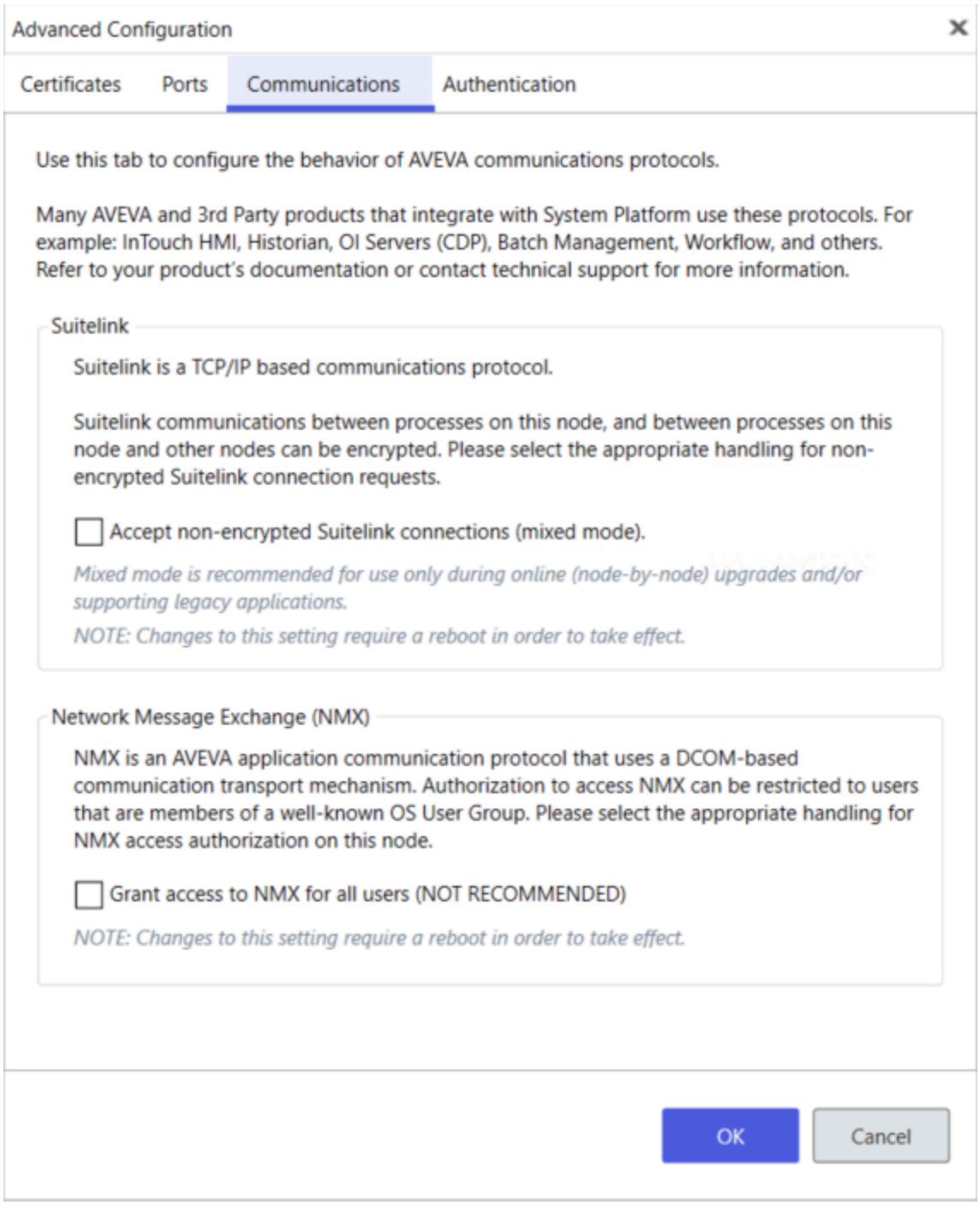
Management Server. The server name is displayed in the Certificate field on the Advanced Configuration dialog. If the configuration is unsuccessful, view details of the errors in the System Management Console.

Communications tab

The Communication tab allows you to configure the behaviour of AVEVA Communications Protocol. The communications tab includes the following configurable fields:

- Suitelink: TCP/IP based communication protocol
- Network Message Exchange (NMX): AVEVA application communication protocol

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.



Note: Communications tab will be automatically hidden if the Configurator do not detect a product that uses Suitelink or NMX communications.

SuiteLink mixed mode setting

Prior to System Platform 2023, enabling the System Management Server, either by connecting to an existing server or by setting this machine as the System Management Server resulted in the following behavior for SuiteLink connections:

- The system first attempted to make a secure connection between a SuiteLink client and the SuiteLink server.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

- If a secure connection could not be established, an unsecured SuiteLink connection was made. Users were not notified if the SuiteLink connection was not secure.

As of System Platform 2023, the System Platform Configurator includes an option to force all communications to be encrypted for SuiteLink connections.

- Mixed mode enabled: This is the default setting if you are upgrading a node from a prior release. With the checkbox set to true (checked), the behavior described above is used, in which unsecured connections are allowed. This mimics legacy System Platform behavior, prior to the System Platform 2023 release. This setting is NOT RECOMMENDED except for the use cases listed below.
- Mixed mode disabled: This is the default setting for new installations. With the checkbox set to false (unchecked), client connections to the SuiteLink server are only successful if the connection is secured, that is both nodes must configured to use the System Management Server. This option ensures that the connection between the SuiteLink Server and SuiteLink clients is always secure (encrypted). If a secure connection is not available, the connection will not be allowed. A secure connection between client and server is only possible if both are configured to use the System Management Server.

Mixed mode use cases

Mixed mode is recommended for use under the following conditions:

- While upgrading an existing System Platform installation (performing a node-by-node upgrade). Reset the mode to disable mixed mode after the upgrade is complete.
- To support legacy applications that do not use encrypted SuiteLink communications.

Note: Whenever the SuiteLink communication mode is changed, a system restart is required before the new mode will take effect.

NMX user access setting

The AVEVA Network Message Exchange (NMX) is an application communication protocol that leverages a DCOMbased transport mechanism for communication between nodes. For new installations, the default setting is to disable access for all users to NMX communications. If you are upgrading an existing System Platform installation, access for all users is enabled by default. Reset the mode to restrict access after you complete the node-by-node upgrade.

- Enable access to NMX for all users: This is the default setting if you are upgrading a node from a prior release. Allowing access for all users is NOT RECOMMENDED except for the use cases listed below.
- Disable access to NMX for all users: This is the default setting for new installations. With the checkbox to false (unchecked), NMX communication is allowed only for the users and accounts that require it. NMX access is allowed for:
  - Members of the OS User Group aaRuntimeUsers
  - Members of the OS Administrators group
  - The System Platform Network Account
  - The local system account (NT System)

Access to NMX for all users use cases

Access for all users is recommended for use under the following conditions:

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

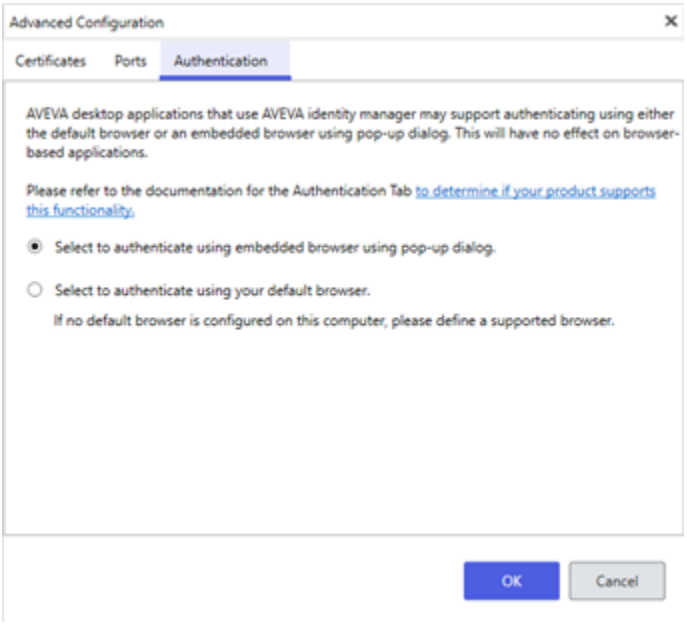
- While upgrading an existing System Platform installation (performing a node-by-node upgrade). Reset the mode to disable access for all users after the upgrade is complete.
- To support legacy applications that require access for all users.

Note: Whenever the NMX mode is changed, a system restart is required before the new mode will take effect.

Authentication tab

The Authentication tab allows you to authenticate using two options:

- Select to authenticate using your default browser: This option displays the AVEVA Identity Manager login page in an embedded browser using a pop-up window when you are prompted to authenticate..
- Select to authenticate using embedded browser using pop-up dialog: This option displays the AVEVA Identity Manager login page in your default browser when you are prompted to authenticate.



Note: The default behavior - an embedded pop-up dialog - is best for a machine that has no browser installed. Selecting to use your computer's default browser will support all other uses.

Federated Identity Provider

Federated identity is a method of connecting a user’s identity across multiple separate identity management systems. Users can move between systems while maintaining security. It allows authorized users to access multiple applications and domains using a single set of credentials.

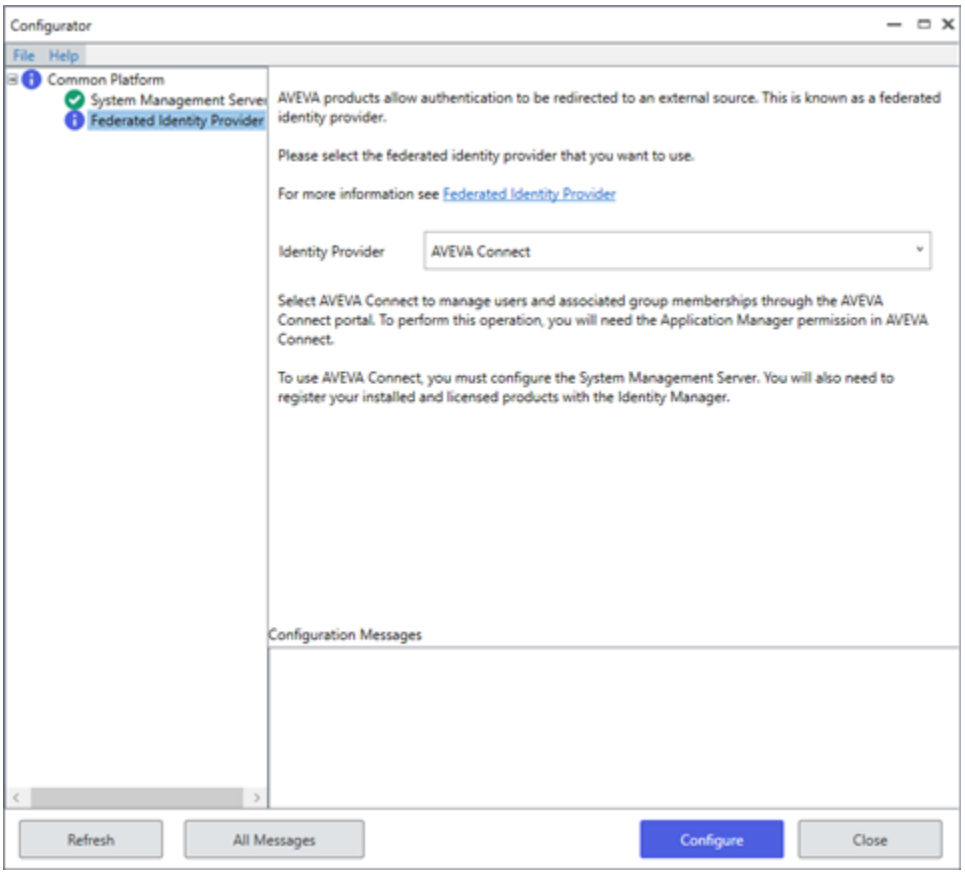
The Federated Identity Provider plugin registers on prem AIM server with the external identity provider (Azure AD or AVEVA Connect), establishing a trust-based relationship between them. The user authentication is delegated to the external identity provider.

When you launch an AVEVA product on a node that’s configured to be a connected experience node, you are prompted to authenticate via one of the two authentication user experiences (as configured) using their federated ID with AVEVA Connect. This requires your Active Directory to be federated or synced with your AVEVA

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Connect account. AIM acts as a middle layer for all the session and authentication redirects and capabilities.

All on-prem Operations Control products are required to use AIM as a local identity provider to run in Operations Control mode. AIM is configured to federate with AVEVA Connect and AVEVA Connect is federated with your identity provider. All cloud services use AVEVA Connect as an identity provider, and it can be configured to federate to your Azure AD or other identity provider.



Before you register your product with federated identity provider, ensure the following:

- Enable AVEVA Operations Control connected experience as your license mode
- Configure System Management Server (SMS)

For more information refer to the following links:

- Key Concepts of Identity Manager
- Certificates
- Identity Provider Options- Azure AD authentication, AVEVA Connect authentication
- Federated Identity Provider Workflow
- Complete Federated Identity Provider configuration
- AVEVA Identity Manager guide

Troubleshooting connection problems

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Configure a multi-user development environment

The ability to have multiple users working concurrently on a single galaxy is often required, particularly for large and/or complex projects that often require multiple developers to work together. If multiple developers will be working with the IDE to build objects and applications, you can structure your environment to enhance team coordination and avoid disconnects and conflicts. Coordinating this effort can be difficult. Even before starting, there are IT infrastructure requirements and limitations that must be considered.

A multi-user environment can be set up with multiple concurrent IDE connections from either standalone computers within the galaxy or RDS client sessions. Advantages of a multi-user development system include:

- Multiple engineers can work simultaneously.
- Changes can always be rolled back.
- The Production Galaxy is isolated from the development system, and development work done on remote workstations does not interfere with or disrupt work being done on the Production Galaxy.
- Individual updates from remote workstations can be released to the Production Galaxy, without requiring check-in/check-out or long wait times.
- The IDE remains responsive, with no interruptions from security updates.

This section describes different ways to configure your development system, and also describes the recommended architecture that will support the most concurrent users. If you select one of the alternative architectures, keep in mind some basic considerations:

- SQL Server Express vs SQL Server: a full version of SQL Server is tolerant of more concurrent users than SQL Server Express (up to three for SQL Express vs. up to five for a full SQL Server version).
- Multi-galaxy pairing must be enabled whenever the development galaxy spans more than one GR.
- In a shared-GR development environment, the IDE may disconnect periodically from the galaxy as the number of users increases. You can avoid this if you use the recommended multi-user development architecture.
- Running InTouch OMI Preview may occasionally crash the the ASB services in shared GR development environments. Again, you can avoid this if you use the recommended multi-user development architecture.

Development system architecture

This section describes three basic system architectures for multi-user development, each with its own set of advantages and disadvantages. The recommended multi-user environment leverages AVEVA™ Integration Studio, a cloud-based development environment that provides high allows a virtually unlimited number of concurrent users, without negatively impacting performance.

In addition to the recommended architecture, two alternative basic system architectures for multi-user development are described in the following sections. Each has its own set of advantages and disadvantages, including the recommended architecture. The basic system architectures are:

- Shared GR with local IDE nodes (alternative architecture)
- Shared GR and IDE node (alternative architecture)
- Local GR nodes plus shared GR (recommended)

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Object wizards: best practices

Regardless of the architecture that you implement, it is recommended that you have a set of hierarchical object wizards to use as the basis for your galaxy, and ensure that best practices for object wizard usage are observed. The most important of these are to develop a three-tier hierarchy of object wizards:

- Top-level object wizards define corporate standards, and serve as templates for derived object wizards.
- Second-level derived object wizards contain refinements that take into account, for example, regional differences or the requirements for different business units.
- Third-level object wizards contain additional refinements, as needed for a particular site or facility.

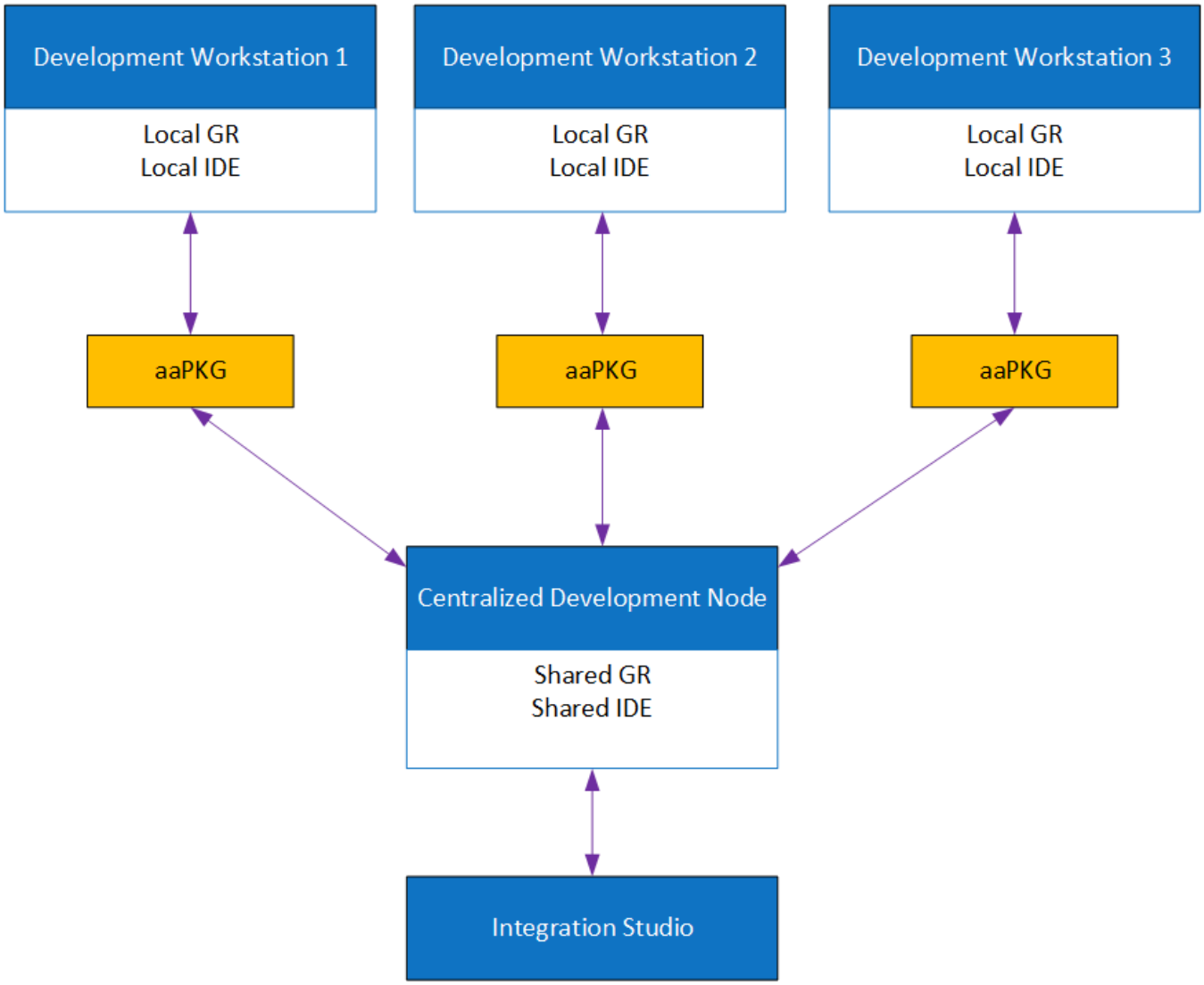
Local GR nodes plus shared GR (recommended)

Local GR nodes for each developer, combined with a shared, centralized GR node offers high performance and eliminates many of the challenges that can affect other development architectures. Most development work can be done on the local nodes, and then uploaded, via an aaPKG file, to the shared GR node for integration and/or testing. The shared GR node can be an on-premises system, or you can use Integration Studio, a cloud service hosted by AVEVA. With a local GR for each developer, most of the challenges that apply to other multi-user development environments are avoided.

Click here for more information about AVEVA Integration Studio.

Many concurrent users are supported (no upper limit has been identified).

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.



This approach has the following advantages and disadvantages:

Advantages:

- High performance: with most development work done on local nodes, there are no slow downs associated with shared access and network traffic.
- Remote access: developers can work from different sites and geographic locations.
- IT infrastructure: with a shared GR and IDE, there are fewer potential bottlenecks between the two Application Server components.
- Multiple projects and versions: the centralized IDE allows better project/version management.

Disadvantages:

- Transferring work between the remote GR nodes and the shared GR node requires manually exporting and importing aaPKG files.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Create a multi-user development system

Distributed multi-user development solution

The following practices are highly recommended:

1. On the GR node that contains your production (functional) Galaxy, add a second Galaxy. We will call this second Galaxy the **Development Galaxy**. The Development Galaxy will act as a central storage location, and will be used to:
   - Maintain the current master templates, including top-level Object Wizards for all objects.
   - Distribute master templates and Object Wizards to each Local GR for development (via .aaPkg export).
   - Control master template updates from local developers (via .aaPkg import).
   - Maintain ownership relations between developers and master templates via OS User Based or OS Group Based security.

     The figure below shows an example of template hierarchy levels, shown in Derivation view. Set access and ownership within the team/organization to each template level to maintain control and security of the project.

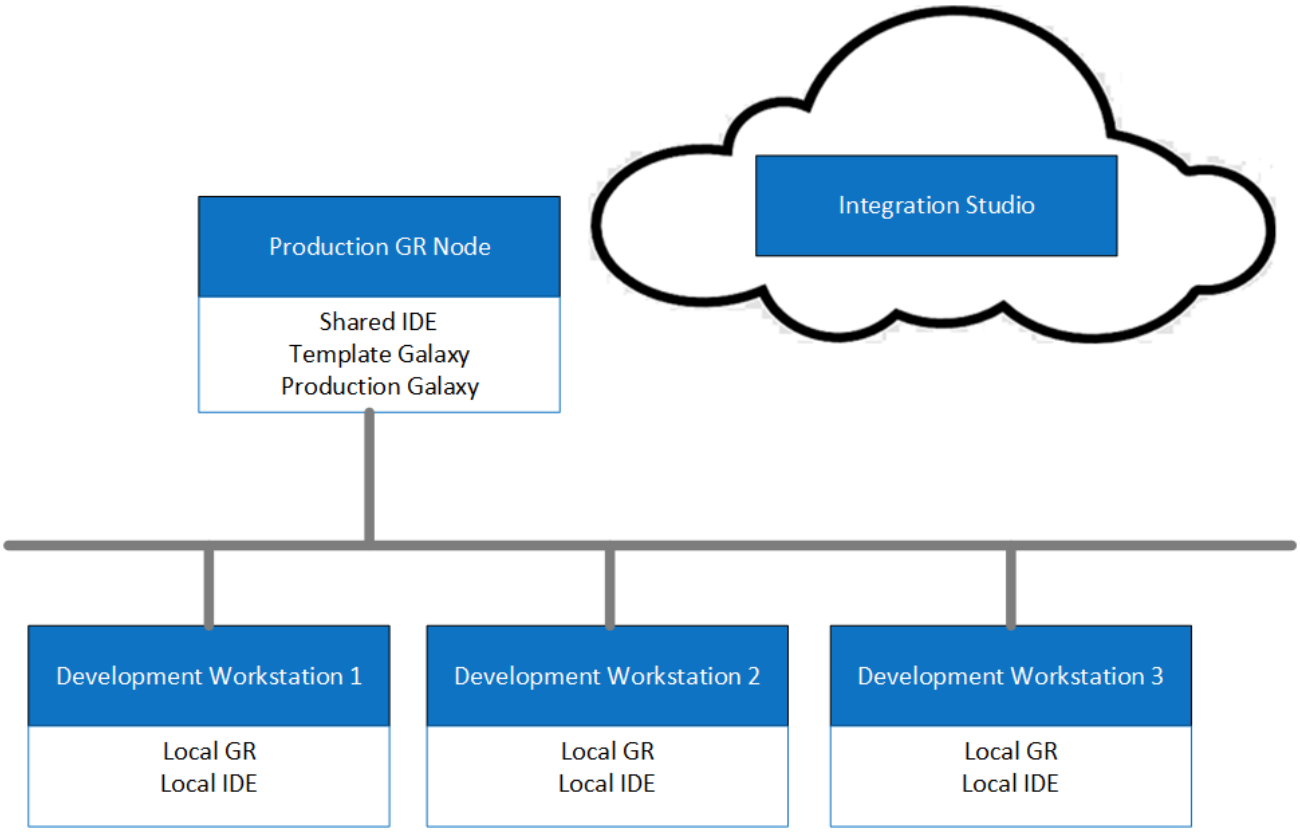


2. Install a temporary, local GR at each IDE development node. The local GR allows each developer to work independently, without interfering with or interference from other developers that may be checking in/out objects, security updates, and deployment actions.

   The local GR can be used for both development and testing.

   Note: The local GR should not require an additional platform count license because it is temporary, and it is not used for run time.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.



Development process

1. Log in to the Template Galaxy.
2. Select and check out template(s) you need for development.
3. Export the selected templates as an aaPKG file from the Template Galaxy.
4. Log in your local GR.
5. Import the aaPKG file to the local GR.
6. Edit, deploy, and test the new galaxy you develop from the imported aaPKG objects.
7. Export the new galaxy as aaPKG file.
8. Log in to the Template Galaxy and undo the checkout on the templates you exported (see step 2).
9. Import the aaPKG file from the local GR (see step 7) to the Template Galaxy.
10. To document your development efforts, check out and then check in (with comments) the changed objects .

Production galaxy updates

Before updating the Production Galaxy:

- Back up the existing Galaxy before making any changes.
- Transfer templates to the Production Galaxy when the project is ready to be updated.
- Create new instances/assets in the Production Galaxy as needed.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Benefits of a distributed multi-user development system

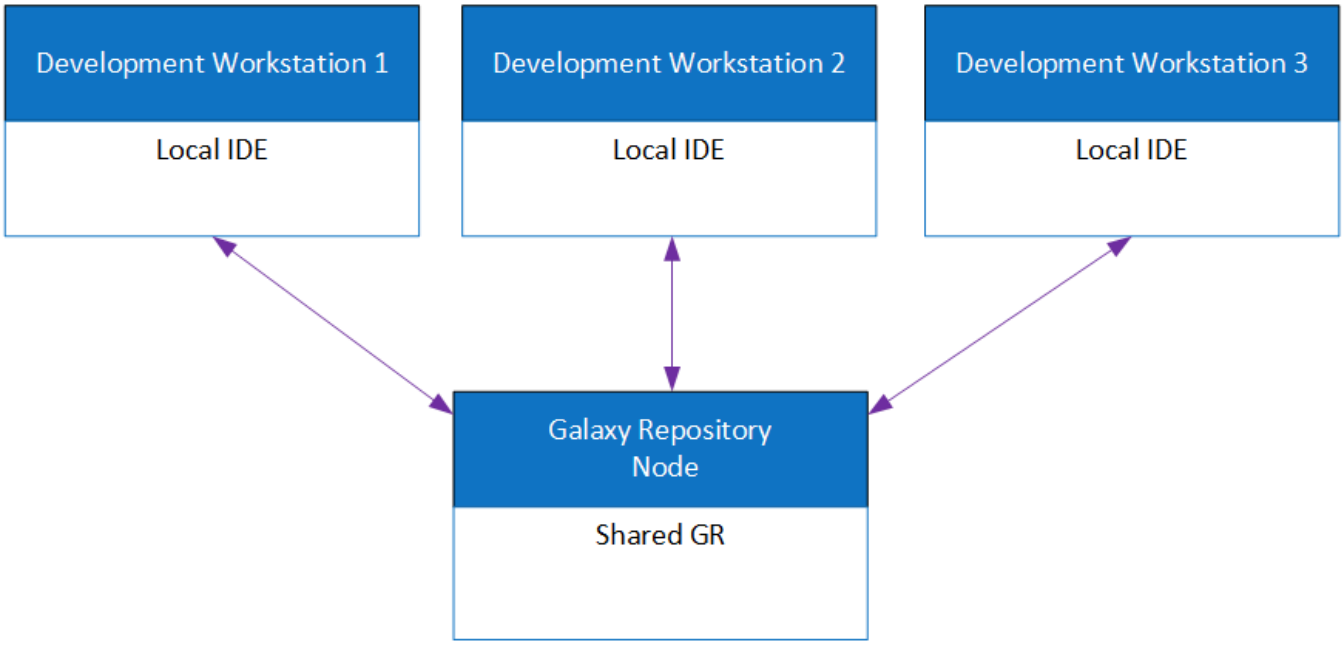
- Several engineers can work simultaneously.
- Unwanted changes can be easily rolled back.
- The Production Galaxy is isolated. This eliminates disturbances and disruptions due to development/ updates.
- Updates and new releases to the Production Galaxy can be managed and controlled.
- Eliminates check-in/check-out, long waits, and time-outs.
- Eliminates non-responsive IDE and aaGR service hanging.
- Eliminates IDE connection interruptions due to security updates.

Shared GR with local IDE nodes (alternative architecture)

As an alternative to using the recommended multi-user development architecture, you can utilize a centralized GR in combination with local IDE nodes for each developer. Note that this approach has a number of disadvantages, which are listed below.

The GR is shared among users, and each user works on objects within the repository, thus ensuring that all developers can see the work that others have done. An object that a user is currently working on is checked out, and unavailable to other users, thus preventing one user overwriting another user's concurrent work.

Using a centralized GR supports up to five concurrent users.



This approach has the following advantages and disadvantages.

Advantages:

- Single repository: All work is stored in, and accessible from, a centralized repository.

Disadvantages:

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

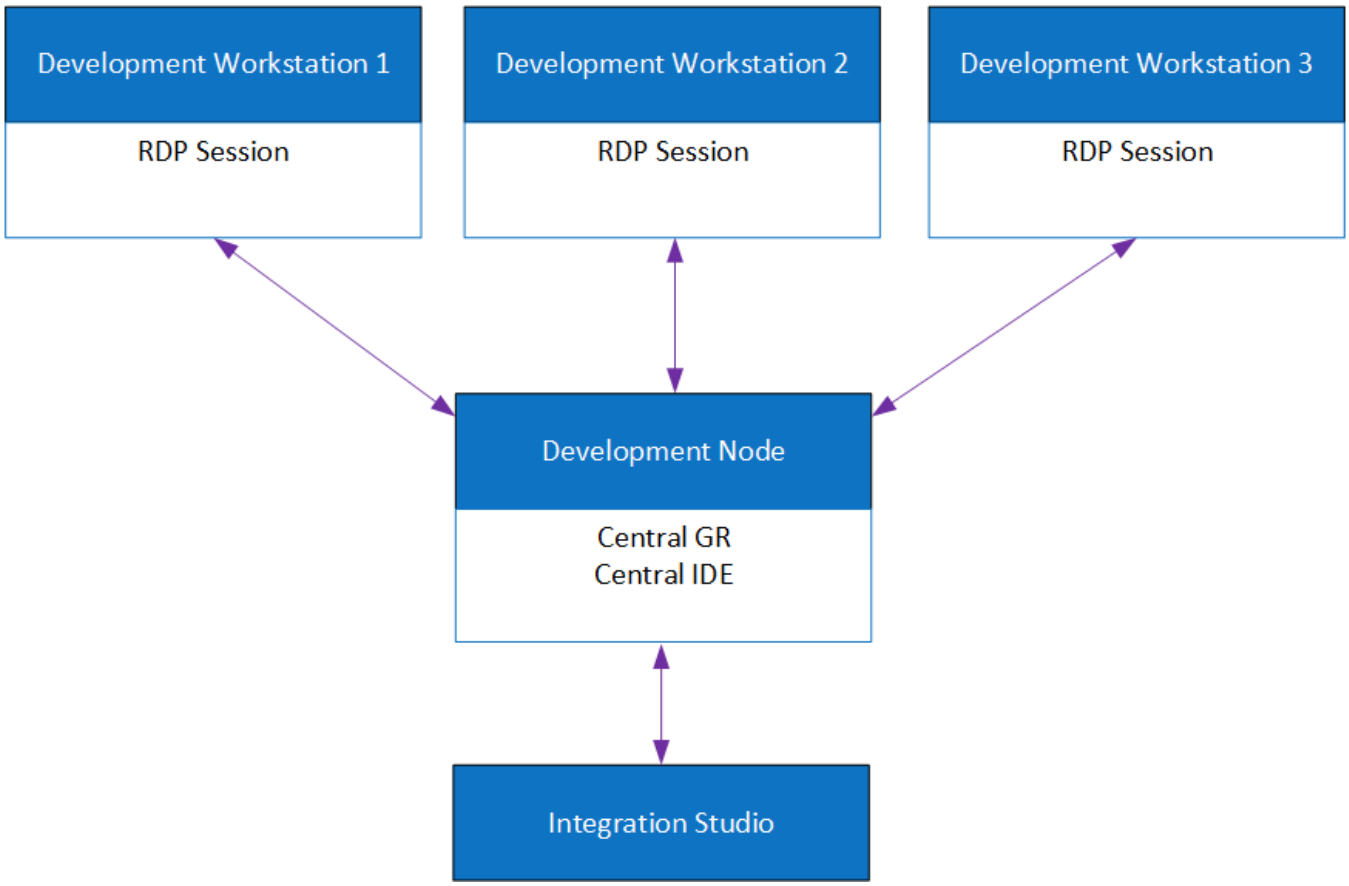
- Congestion: since all users are accessing the same GR, system response can slow down. A limit of five concurrent users is recommended.
- Team coordination: with multiple users working within the same GR, planning and communication between users is essential.
- Project coordination: managing multiple projects, as is the ase with multiple users, requires planning and coordination.
- Versioning: an effective way to maintain versioning of a project is difficult to implement.
- IT infrastructure: adequate bandwidth and resource provisioning must be in place to enable adequate system response time.

Shared GR and IDE node (alternative architecture)

As a second alternative to using the recommended multi-user development architecture, you can utilize a centralized GR in combination with centralized IDE. Developers then use RDS to access these resources.

This arrangement, with both the GR and IDE shared by developers, allows better coordination between team members that having a local IDE for each developer, although like the other alternative architecture described above, it also has its share of disadvantages including cost and congestion, as listed below.

Sharing both the GR and IDE requires more resources than simply sharing the GR however. The maximum number of concurrent users will vary between two and five, depending on computer hardware resources, whether or not you are using SQL Server Express (which will support fewer users) or a full version of SQL Server, and the size of the galaxy.



© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

This approach has he following advantages and disadvantages:

Advantages:

- Single repository: All work is stored in, and accessible from, a centralized repository.
- IT infrastructure: with a shared GR and IDE, there are fewer potential bottlenecks between the two Application Server components.
- Multiple projects and versions: the centralized IDE allows better project/version management.

Disadvantages:

- Congestion: since all users are accessing the same GR and IDE, system response can slow down. A limit of five concurrent users is recommended.
- Cost: RDP CALs (client access licenses)

Managing Security for Application Server

This section provides a general overview on how to securely deploy your AVEVA software product as an Industrial Control Systems (ICS) application.

This section is not meant to be comprehensive, and it does not provide any detailed instructions. It is only a collection of basic concepts and recommendations that you can use as a checklist to secure your own systems. If you need help with a specific item in this guide, see the official documentation for that item -- for example, if you need help with your anti-virus software, see the documentation for that software.

AVEVA's approach to securing site networks and ICS software is driven by the following principles:

- View security from both Management and Technical perspectives
- Ensure that security is addressed from both IT and ICS perspectives.
- Design and develop multiple network, system and software security layers.
- Ensure industry, regulatory and international standards are taken into account.
- Aim to prevent security breaches, supported by detection and mitigation.

These principles are realized by implementing the following security recommendations:

- Prevent security breaches using the following components:
  - Firewalls
  - Network-based intrusion prevention/detection
  - Host-based intrusion prevention/detection
- Segregate IT and Plant networks
- Include a clearly defined and clearly communicated change management policy. For example, firewall configuration changes.

  Note: AVEVA strongly recommends following the guidelines prescribed by the U.S. Department of Commerce for securing ICS software. The document "Guide to Industrial Control Systems (ICS) Security" [NIST Special Publication 800-82 Revision 2] (https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-82r2.pdf) provides detailed information about ICS, typical system topologies, security threats and vulnerabilities, and recommendations for implementing security measures.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

General considerations for security

Before you review the information in this section, it is recommended that you go through the following checklist to ensure you plan to cover the security areas that apply to your ICS and organization.

Security Area	Reference Section
Physical and virtual access to the host	General guidelines for securing the host
Latest Windows patches applied	Windows updates
Protecting the host from viruses and malware	Scanning the Host
Access to content on the host	Protecting the applications and content on the host
Securing your network	Secure the network
Configuring services and ports	Manage network services and ports
Securing client/server communication	Secure communication between the client and server
User and group management	Secure systems through authentication and authorization
Planning for emergencies	Contingency planning

For a list of security feature help topics refer to the table at the end of this section.

Secure the host

Given the sensitive nature of industrial control, it is important to secure not only the ICS software, but also:

- the host on which it runs
- the network to which it is connected
- the hardware used for the ICS software.

Note: The "host" is the Windows computer or Windows Embedded device on which your ICS software is installed and running.

There are several factors to consider for securing the host including:

- Access to the host
- Keeping track of and applying the latest Windows updates
- Keeping the host computer free of viruses and malware
- Protecting the applications and content on the host

Each of these factors is covered in the sections below.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

General guidelines for securing the host

Here are a few guidelines to secure the host:

- Use an account with administrative privileges to install the ICS software, and one without administrative privileges to run the ICS software.
- Restrict configuration of ICS to a limited set of users.
- Consider running the ICS software as a Windows service, if that option is available. If the ICS software is run as a service, run it as a low privileged virtual service account.
- Once the host is fully configured and placed in its permanent location, restrict physical access and remote access to it so that only authorized personnel (for example, system administrators, application engineers, run-time operators) can use it.
- Consider disabling or removing physical ports (for example, USB, memory card) that might be used to connect external storage devices and then transfer data.

Windows updates

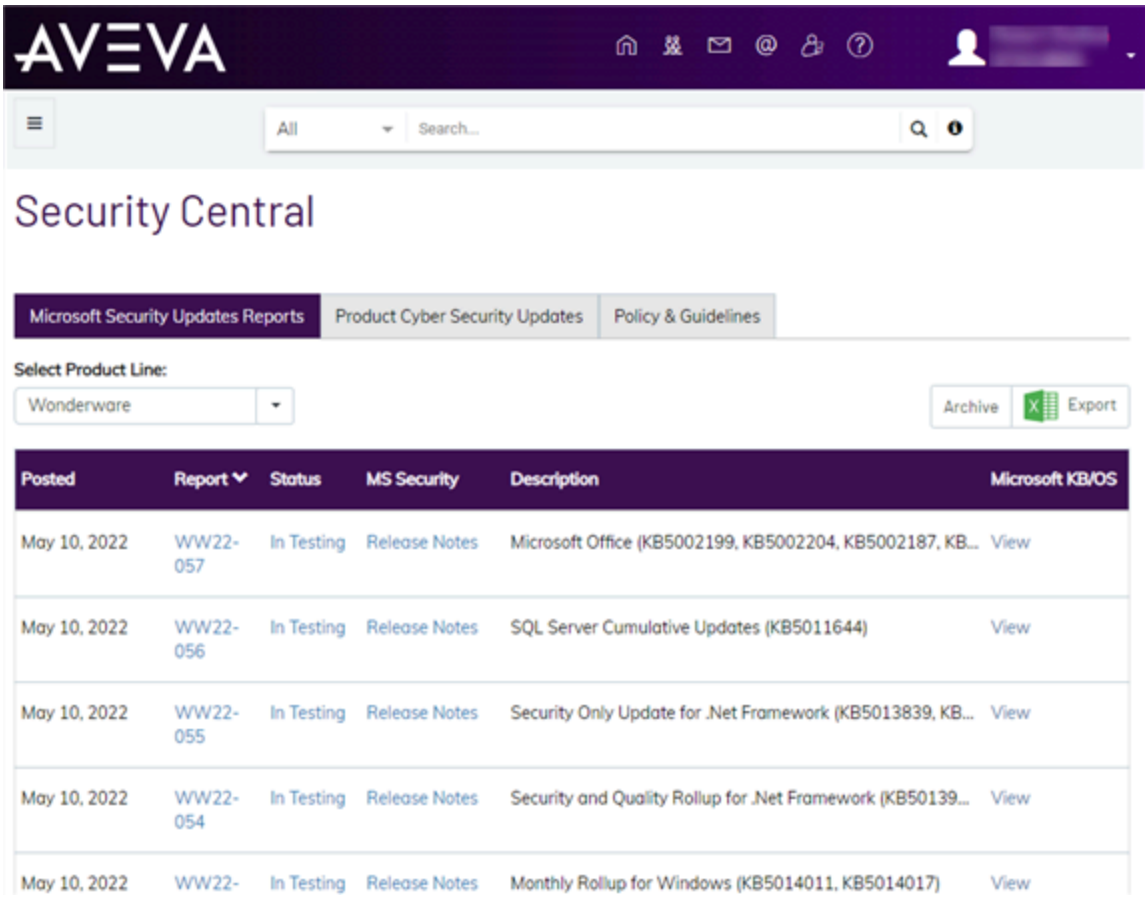
Check that the Windows operating system on the host is a version that is under what Microsoft calls "mainstream support", which means Microsoft actively maintains and releases updates for it. Older versions of Windows are under Microsoft "extended support", which means they are not actively maintained and therefore might become vulnerable without notice. For more information about the different versions of Windows and the different levels of support, see the Windows lifecycle fact sheet on the Microsoft Web site..

Automate Microsoft product updates using Microsoft Windows Server Update Services (WSUS), which enables you to manage and distribute updates to computers on your network. For more information about WSUS, see Windows Server Update Services on the Microsoft Web site. If the host does not or will not have a reliable connection to the WSUS server, perhaps because it is located on a private network, you can either develop a procedure to manually apply updates or consider changing the operating system to a Long-Term Servicing Channel (LTSC) version of Windows, which is updated less frequently.

In addition, AVEVA ICS software is tested for compatibility with Microsoft updates the results of which are published on the Security Central site. Security advisories and bulletins are also published on this site.

Note that when Windows updates run in the background, there is the possibility that different software processes can be adversely affected. Therefore, it is important to schedule the updates to run only during planned shutdown periods.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.



Configuring Automatic Updates

If Windows is configured to update automatically, these automatic updates, when running in the background, can disrupt System Platform components, including Application Server and OMI during installation/upgrade, configuration and run-time operations. These updates may cause the IDE, GR, visualization core platform, and other services to shutdown unexpectedly. Therefore, we recommend that you DISABLE automatic Windows updates, or else make sure that the updates will run only when production systems are shut down and that no development work is occurring.

Scanning the Host

Use both anti-virus and anti-malware software and file integrity checking software to regularly scan the host.

Windows includes Windows Defender by default, but you may choose to install and use additional software that scans for more types of malware or performs other functions. If you do that, make sure the software is provided by a reputable company. And, as with the operating system, if the host does not or will not have reliable access to the software's update service, develop a procedure to manually apply updates. If you develop a manual update procedure, it should account for all devices on a network or at a site, because a single outdated device can leave the entire network or site vulnerable.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Protecting the applications and content on the host

To protect the applications and content on the host:

- Enable Windows Firewall, and configure it to close all ports that are not used by the ICS software. For more information about port usage, see Manage network services and ports.
- Disable Windows features like remote desktop and file sharing, and remove unnecessary programs like games and social media.
- Restrict access to the files, databases, registry and other resources on the host.
- Use Windows BitLocker to encrypt the hard drive of computers that are either mobile or not located in a secure facility. However, BitLocker may impact the performance of computers.
- Consider using server-class storage (SANs) infrastructure to avoid storing sensitive data on mobile devices.
- If your application stores data in SQL Server, Windows authentication can provide better application security than SQL Authentication. If you switch from Windows Authentication to SQL Authentication, a pop up dialog will appear recommending that you use Windows Authentication for this reason. If you choose to ignore this warning and proceed with SQL Authentication, click **OK**. A similar message will be logged in the OCMC (SMC) Log Viewer.

AVEVA leverages the security built into the Windows operating system to store and manage encryption keys. The encryption keys are stored in a local storage location called the encryption store. For more information about the Windows encryption store, refer to the Microsoft documentation, located at Certificate Stores (https://docs.microsoft.com/en-us/windows-hardware/drivers/install/certificate-stores).

Phases of Data Protection

Data exists in three different phases, and protection must be provided for each phase:

- At rest
- In transit
- In use

Data at rest

Data at rest is data that is not currently being used or accessed, such as data stored on a hard drive, laptop, flash drive, RAID array, network attached storage (NAS), storage area network (SAN), or archived/stored in some other way. Data protection at rest aims to secure inactive data stored on any device or network. For protecting data at rest, you can simply encrypt sensitive files prior to storing them and/or choose to encrypt the storage drive itself. BitLocker Drive Encryption, which you can invoke via the Windows Control Panel, can be used to invoke whole-drive encryption.

In the context of SCADA and ICS systems, data at rest includes stored configuration data, historical data, backups, and other static data. The duration of storage, that is, long term or short term, does not impact this classification of data at rest. Protection for data at rest is applicable for as long as the data exists in this condition; it is not a fixed condition.

Proper authorization rights need to be set in place to ensure the data is not being viewed by unauthorised users. Other steps can also help, such as storing individual data elements in separate locations, such as a corporate-

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

approved offline backup to decrease the likelihood of attackers gaining enough information to commit fraud or other crimes. Offline backups are the best mitigation against the threat of ransomware.

Data in transit

Data in transit, or data in motion, is data that is actively moving from one location to another.

In the context of SCADA and ICS systems, this encompasses deploying a project to a run-time node, transmitting process variables, VTQ data, and other data that is sent between nodes in a running, production system. This includes alerts and alarms.

Data protection in transit is the protection of this data while the data traveling, including the following examples:

- From node to node within a network
- From network to network
- Accessed via internet
- Transferred from a local storage device to a cloud storage device

Wherever data is moving, effective data protection measures for in-transit data are critical as data is often considered less secure while in motion. Best security practice is to ensure TLS 1.2 encryption is used for all communications using the HTTPS protocol.

Data in use

Data in use refers to data that is being processed or accessed either locally or remotely. This generally involves placing data into memory (RAM) for access and processing by applications and users, potentially multiple users across different computers, mobile devices, remote terminals or other device. Data in use is particularly vulnerable to attack. To protect data in use, encryption, user authentication, and identity management is highly recommended.

In the context of SCADA and ICS systems, data in use can apply to databases, such as those used actively by a historian or deployed to a run-time node. This needs to be safeguarded by a secure transfer channel.

Secure the network

Usually, the host computer will have some sort of network access; it is increasingly rare for an ICS device to run as an entirely standalone device. The host may use the network to communicate with other ICS components such as controllers, sensors, databases, remote clients, and even other hosts in peer-to-peer relationships. You may also use the network to manage several ICS devices from a development or supervisory computer.

Once you determine that the host will have network access, decide how it will connect to the network. In recent years there has been a shift from wired networks (that is, "Ethernet") to wireless networks ("Wi-Fi"), even for business and industrial uses. We recommend against using Wi-Fi for your ICS network because you do not have physical control over who or what might access the network. Any computer or device within range of the Wireless Access Point (WAP) can try to access the network, and even if the network is ostensibly secure, an intruder can intercept and analyze network traffic and potentially discover a vulnerability.

Nevertheless, if you decide to use Wi-Fi for your ICS network, enable all access control features on the WAP including encryption (for example, WPA/WPA2), a strong password and a list of authorized MAC addresses. Do not try to "hide" the Wi-Fi network by disabling broadcast of the Service Set Identifier (SSID), because doing so

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

actually generates more network traffic that can be intercepted and analysed.

Segment the ICS network

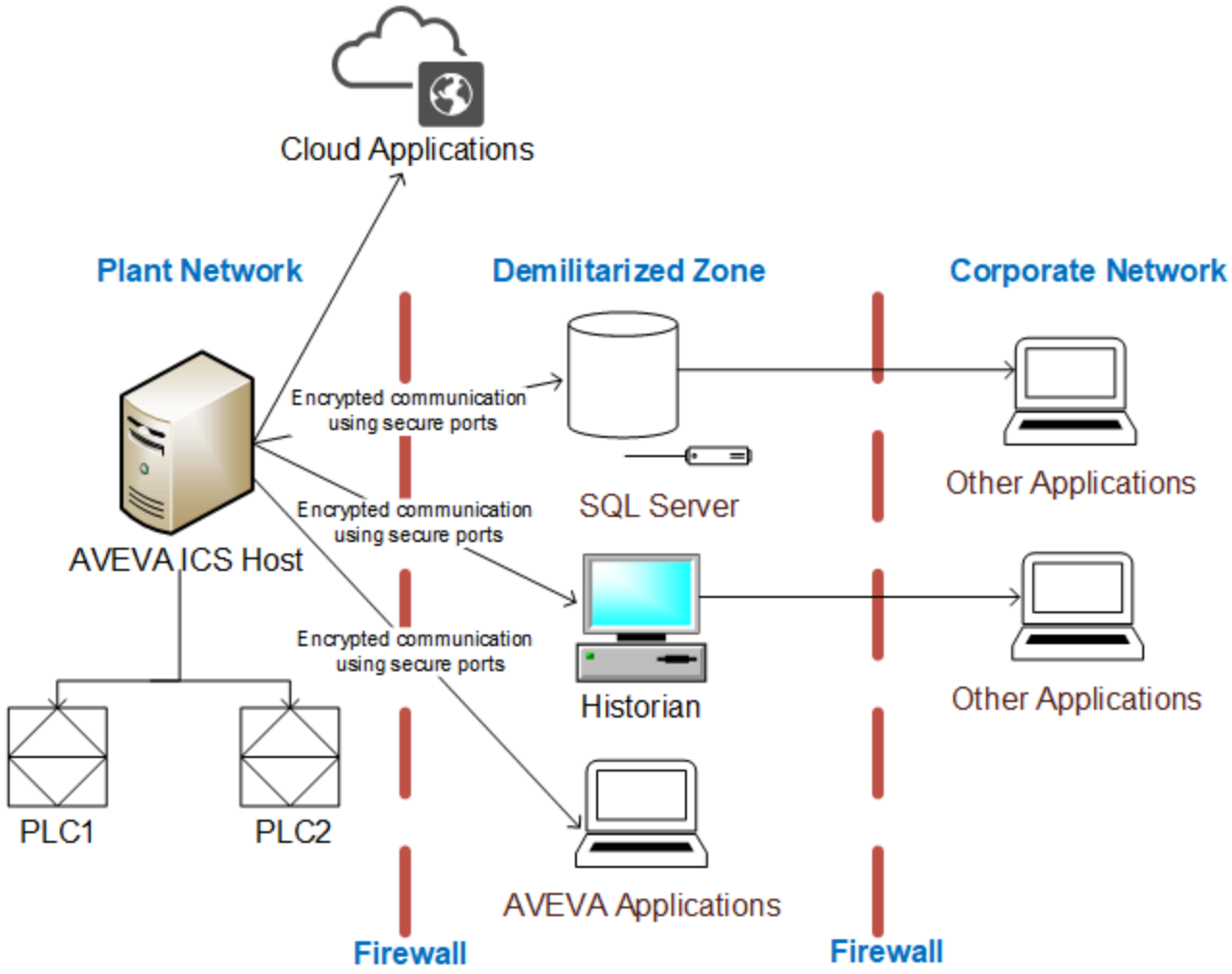
The ICS network itself can be either physically or logically segmented from your other corporate networks. A physically segmented network is by definition the most secure. The network hardware and all computers and devices connected to it form a single closed network with no physical connection to any other network, so an intruder cannot access the network unless they also have access to the physical location.

In contrast, a logically segmented network is physically connected to your other corporate networks and/or the public internet, but it uses various methods to segregate ICS network traffic from other network traffic. This may include:

- Using a unidirectional gateway
- Implementing a Demilitarized Zone (DMZ) network architecture with firewalls to prevent network traffic from passing directly between the corporate and ICS networks
- Having different authentication mechanisms and credentials for users of the corporate and ICS networks.
- The ICS should also use a network topology that has multiple layers, with the most critical communications occurring in the most secure and reliable layer.

Given below is a sample deployment topology.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.



In no case should your ICS network and devices be directly accessible from the public internet. If there is some part of your ICS that you want to be accessible, (for example, if you want be able to view web-enabled HMI screens from a browser or smart phone), your ICS software should include features that securely pass the necessary traffic between your ICS network and a public-facing server.

Manage network services and ports

A network port is an endpoint of communication in an operating system. While the term is also used for hardware devices, in software it is a logical construct that identifies a specific process or a type of service. In other words, a network port is conceptually different from hardware ports like USB, memory card, and even the wired network connection.

Computers and devices can access many different network services at the same time by communicating on different network ports. Each network service or communication protocol has an associated port number. Some port numbers are specified by international standards, and therefore they are universally recognized. Other port numbers are claimed by proprietary software, and in most cases they can be changed in the software settings if there is a conflict with other software or services.

Firewalls control network traffic by either accepting or refusing communication on these network ports. If a port is open, it accepts communication, and if a port is closed, it refuses communication. Almost every layer of a network -- from the operating system on an individual computer or device, to the router that manages traffic

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

within a network, to the gateway that manages traffic between networks -- has its own firewall.

The documentation for your ICS software should include a list of network ports that are commonly used by the software. Given the nature of ICS, the list typically includes services like web, email, file transfer, external databases, device drivers, and the ICS software itself for server-client communications. Configure the firewalls to open only those network ports that are actually used by your ICS. Disable all unused services and close all unused ports.

Secure communication between the client and server

Like most server-client applications, your ICS software should support secure communication between the server and client in order to prevent the messages sent between those two stations from being read by any other stations on the same network. Note that this is different from securing the network itself in order to prevent unauthorized access to the network.

This sort of communication is also sometimes known as "Encrypted Channel" because it uses the Transport Layer Security (TLS) standard to encrypt the server-client messages. The latest version of the standard is TLS 1.3 (released August 2018), but it is not yet in common use. The latest version of the standard in common use is TLS 1.2 (released August 2008). TLS supersedes the earlier Secure Sockets Layer (SSL) standard, although SSL is still used in older applications.

Certificates

TLS and SSL use a system of certificates and keys to digitally "sign" the messages sent between the server and client. When the server establishes communication with the client (and vice versa), it presents its certificate which identifies its name, network address, organization, physical location, and so on. The client can then choose to either accept or refuse the certificate as presented. If it accepts the certificate, it agrees to accept messages encrypted with the same certificate, and it uses the associated key to decrypt those messages.

When you configure this sort of communication, you need to choose one of the following:

- Using self-signed certificate
- Using certificates signed by a Public Certificate Authority (CA)
- Using Domain-issued certificates or certificates signed by a Private Certificate Authority using systems like Microsoft Active Directory Certificate Service (AD CS)

A self-signed certificate is issued and signed by the same application that presents it. Self-signed certificates are easy to create and manage, but they are secure only if you control both the server and the client and therefore control which certificates are installed on each.

In contrast, CA-signed certificates are slightly difficult and expensive to acquire, but they are more flexible than self-signed certificates because you do not need to control both the server and the client. If you configure the server to present a CA-signed certificate, the client will accept the certificate because it recognizes the Certificate Authority.

Domain-issued certificates are internal certificates typically managed by your IT department. They are issued and validated by an Active Directory Certificate Authority. Domain-issued certificates are free and can be issued instantly.

You need to renew CA-signed and Domain-issued certificates at regular intervals.

For more information about how to enable Encrypted Channel features and manage self-signed certificates in your ICS software, see the documentation for that software. However, acquiring a CA-signed certificate and then using it to sign other certificates is typically beyond the scope of ICS software documentation.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Note: Encrypted and unencrypted communications typically use different network ports.

Secure systems through authentication and authorization

Typically, ICS software is comprised of a large number of systems, each accessed by a variety of users including engineers, operators and managers. The level of access that each type of user requires is different. So, it is necessary to manage user authentication and authorization to secure the system.

Authentication

Authentication is the process of verifying a user's/system's identity. Authentication can be managed in the following ways:

- Within the ICS software through application accounts
- Through Windows accounts, which can be local to a single computer
- Through Authentication systems (see the next section for details)

While ICS software allows for user and role management, it can become cumbersome and complicated to manage a large number of user accounts as employees and roles change. Because of this, use of Windows accounts is generally preferred.

Authentication systems

Authentication systems such as Active Directory and Lightweight Directory Access Protocol (LDAP), referred to as authentication servers, are a repository of and provide centralized management for all system accounts and individual user accounts. An authentication protocol is used for all communication between authentication servers and the user or server requesting authentication.

Even though use of authentication systems provides improved scalability, the following factors must be considered depending upon the size and complexity of your operations:

- It is important that the authentication servers are highly secured.
- The authentication server system creates a single system for managing all system accounts. Therefore, it requires to be available at all times. To ensure minimal disruption during an emergency, redundancy must be considered.
- Permit caching of user credentials only for users who have authenticated their identity recently.
- Networks that support the authentication protocol must be reliable and secure to assist in trouble-free authentication.

It may also be worthwhile implementing two-factor authentication using additional applications such as PingID.

Authorization

Authorization is the process of providing the correct level of privileges to users by applying access rules to authenticated users, systems (HMIs, field devices and SCADA servers) and networks (remote sites' LANs).

Manage users and groups through ICS software

Your ICS software should have a built-in security system that controls who may use the software and what privileges they have.

Users should be assigned permissions that determine what each user is authorized to do within the ICS system.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Permissions can be managed either on a per-account basis or on a group basis by making use of roles. Group or role-based access control is preferred as it greatly simplifies management. Users can be moved from one role to another as the organization's needs change, and can also be members of multiple roles if required.

Each user should have their own user account with a unique user name and a strong password. The user account can then be assigned to one or more groups.

Accounts should always be assigned the least privileges necessary to perform their functions. Accounts with Windows Administrator permissions should be reduced to the minimum, and typically only used to install and configure the software. Likewise, accounts with SQL Server SysAdmin privileges should be reduced to the minimum, and typically only used to install and configure the software.

In most cases, the ICS software will allow associating Windows Groups with roles within the product. While defining and assigning roles, consider the following:

- Roles should be defined to have the least privileges necessary for their functionality.
- Roles should be limited to a single purpose in order to simplify the permissions assigned to them.
- Users can be members of multiple roles if necessary.

Manage users and groups through windows

When you configure security, you may choose to do one of the following:

- Keep the configuration local to a single application.
- Share the configuration between multiple applications.
- Manage the configuration as part of the network domain (for example, using Active Directory). This option typically allows users to have the same user account for the network, the host, and the ICS software. Using Active Directory gives you the following advantages:
  - A centralized repository for user and group data, enabling effective implementation of security policies and procedures.
  - Provides a single point of access to all network resources after the user is identified and authenticated.

To manage users and groups:

- First define a specific role for each group, and then configure the group privileges to fit that role.
- Groups may overlap, but it is often better to have clearly separate groups and then assign individual users to multiple groups, if necessary.
- Set or change the password for the ICS software's default user (e.g., "guest").
- Define stringent password policies to force users to create strong passwords. Enforce mandatory password updates on a regular basis.

Contingency planning

Incidents are inevitable. It is, therefore, important to develop a strategy to detect an incident quickly and respond to it in a timely manner in order to minimize loss and protect your system. An organization must consider contingencies arising from incidents such as fire, flood and so on, and those arising from failure of hardware or software components. Cyber attacks such as ransomware are becoming more common and must

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

also be considered.

An organization should have contingency plans in place to cover the entire range of failures and eventualities. Employees should be trained and be familiar with the contents of the contingency plans.

As part of planning for contingencies, it is important to establish a site, physically separated from the central one, that has replication capability. Doing so will ensure the integrity of an operational system where the central site is at risk from fire, floods or other disasters. The replication capability includes having duplicated hardware, and requires software configuration and key state information to be periodically propagated from the central site to the recovery site. Each recovery scenario is unique, so it is important to consult with system integration experts regarding the design of communications equipment, hardware and the configuration of the software.

Protecting the data stored in your system is also of paramount importance. Full and incremental backups must be scheduled on a regular basis. Backups should be verified by running tests to restore from backed up data. Backups should be stored offline so that they are safe from cyber attacks such as ransomware.

Organizations should also have business continuity and disaster recovery plans that are similar to contingency plans. These plans are covered briefly in the sections to follow.

Business continuity planning

Business continuity planning addresses strategies to maintain or re-establish production in the event of any disruption. These disruptions may be caused by a natural disaster (flood, earthquake, etc), by an intentional or unintentional human-made event (arson, operator error, power outage, etc), or by system failure.

Depending upon the duration of the potential ICS application outage caused by a disruption, operational recovery plans for short-term outages and disaster recovery plans for long-term outages must be formulated. It is also important to employ physical security for areas of a production site that house data acquisition and control systems that might have higher-level risks. Your business continuity plan should specify system and data recovery procedures for your systems. Once the recovery procedures are documented, a schedule should be developed to test the recovery procedures. Particular attention must be paid to the verification of backups of system configuration data and product or production data. The procedures should be reviewed periodically.

If you are accessing cloud-based solutions, ensure that systems are available at all times. In case of a disaster, services should switch to a new physical location to provide continued service.

Disaster recovery planning

A disaster recovery plan (DRP) is a set of procedures to protect and recover an IT infrastructure in case of a disaster. It contains the procedures to follow before, during and after a disaster. Disasters can be natural, environmental or human-made (intentional or unintentional).

A DRP is essential for continued availability of the ICS, and should cover the following:

- When the DRP should be activated depending upon an event, its duration and its severity.
- Detailed course of action for operating the ICS manually until external connections are secured.
- Personnel responsible for each procedure.
- Processes for securely backing up data and restoring it. This should cover:
  - Requirements for building redundancy.
  - File backup procedures.
  - Frequency of backups.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

  - Storage mechanism for full and incremental backups.
  - Safe storage of installation media, license keys and configuration information.
  - List of individuals responsible for performing, testing, maintaining and restoring backups.
- List of personnel with physical and virtual access to the ICS.
- Detailed configuration information about the components of the ICS.
- Schedule for testing the DRP.

Security configuration for InTouch HMI

The table below lists the security areas that you need to configure for InTouch HMI and the details of the section(s) in this guide that provide the corresponding instructions.

Security Area(s)	**Topic(s) in this guide**	**Summary**
Protecting the Applications and Content on the Host	Configuring an Inactivity Time-Out	Configure WindowViewer to automatically log off an inactive operator from an InTouch application. Inactivity time-out is not supported when connected experience is enabled.
Securing Systems through Authentication and Authorization	Authentication and Authorization Based Security	Users needs to authenticate themselves before using an InTouch application. InTouch verifies whether the authenticated user is authorized to use specific functionality.
	Using Virtual Accounts	Using virtual accounts provides an additional layer of security when accessing alarm functions. See Using Virtual Accounts in the InTouch HMI Alarms and Events Guide.
Protecting the Applications and Content on the Host	Locking System Keys	Restrict operator access to standard Windows functions by disabling system keys on the computer running an InTouch application.
Securing Systems through Authentication and Authorization		

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Security Area(s)	Topic(s) in this guide	Summary
Managing Users and Groups through ICS Software	Using InTouch-Based Security	Restrict which functions an operator is allowed to perform by linking those functions to internal tags.
Managing Users and Groups through Windows	Using Operating System-Based Security	Inherit some user/group account policies from the Windows operating system.
Securing the application data folder	Applying Role-based Security to the Application Folder	Users can define user roles and user groups to manage access to the application data folder.

Security configuration for AVEVA Application Server

The following table below lists the security areas that you need to configure for Application Server and the details of the section(s) in this guide that provide the corresponding instructions.

Security Area(s)	Topic(s) in this guide	Summary
Basic information about using and configuring the Application Server IDE.	System Platform IDE	Description of the Application Server Galaxy and the objects that comprise it. Also describes how to create, connect to, and log into a Galaxy.
Configuring user permissions, credentials, and galaxy security.	Configure security	How to configure security to manage access to configuration and run-time aspects of your Application Server application.
Protecting objects and attributes at configuration time and setting object security.	Object editor layout	How to use the Object Editor restrict the write access that application builders have while configuring objects through locking/unlocking template attributes, setting object security, and group locking.
Configuring alarm and event notifications, and setting user access and security at run time.	Alarms and events	How to configure alarm and event notifications and enable security for system operators, such as which individuals are allowed to acknowledge alarms.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Security Area(s)	Topic(s) in this guide	Summary
Deploying objects and Galaxies to run time.	Deploy	How to deploy an entire Galaxy or individual application objects to run time, and how to view operational and security status at run-time.
Galaxy management and creating a new Network Account.	Manage galaxies	Backing and restoring Galaxies, configuring multiple network cards, managing the Network Account, and synchronizing Galaxy objects.
Configuring large Galaxies that span more than a single GR node.	The multi-galaxy environment	How to pair Galaxy Repositories to configure multi-galaxy communication. Includes troubleshooting information.
Scripting methods for security.	InTouch functions	The following scripting methods can be used in run-time applications to enhance security: • AddPermission() Function • AttemptInvisibleLogon() Function • ChangePassword() Function • GetAccountStatus() Function • InvisibleVerifyCredentials() Function • IsAssignedRole() Function • LogonCurrentUser() Function • PostLogonDialog() function • QueryGroupMembership() Function • SignedAlarmAck() • SignedWrite()

Data protection in Application Server and System Platform

Protection for System Platform data can be enabled as it is stored (data at rest), transferred (data in transit), and used (data in use).

- **Data at rest** refers to the data that is stored on a hard disk drive (HDD) or solid state drive (SSD). In the context of System Platform, this data includes SQL databases, code modules, and configuration data that exist on a GR node or Historian node, as well as any other stored data, such as Galaxy backups.
- **Data in transit** refers to data being transmitted across nodes, such as deploying a Galaxy to a run-time node,

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

and changing or propagating configuration changes to GR nodes.

- **Data in use** refers to run-time data, such as changing a set point, acknowledging an alarm, or the data being viewed on a visualization client.

The following is a summary of how Application Server allows data to be protected through encryption utilizing industry standards.

Data protection summary

- **Data in transit** is encrypted between System Platform nodes, provided that AVEVA recommendations are followed and the System Management Server (SMS) is enabled. SMS configuration is normally done when System Platform is installed, and applies to all System Platform nodes for which SMS has been enabled. Communications are encrypted using TLS 1.2.
- **Data at rest** can be protected, if necessary, by enabling BitLocker. BitLocker is a Windows feature that encrypts an entire storage volume. Note, however, BitLocker can slow down performance.
- **Data in use** can be protected by enabling strong user authentication policies and procedures. Enable Galaxy security to limit user access to the functions that each user needs to perform their job. The use of Named Credentials within AVEVA OMI ViewApps can enhance user-based or group-based security at run time.
- **Encryption key management and storage:** TLS certificates are stored locally by the operating system in the Windows Certificate Store. Keys can be managed with the Windows Certificate Manager.
- **Federal Information Processing Standards (FIPS):** Application Server does not support the FIPS security policy option in Microsoft Windows.The Federal Information Processing Standards are United States Government standards that provide a benchmark for implementing cryptographic software. If FIPS is enabled in the Local Security Policy settings, disable it.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Diagnose

System Platform includes several programs and utilities that can be used monitor, diagnose, and troubleshoot Application Server and other System Platform products.

Most of the diagnostics are accessed through the Operations Control Management Console. To open the Operations Control Management Console:

1. From the Start menu, open the **AVEVA** folder.
2. Select **Operations Control Management Console**.

Operations Control Management Console diagnostics

- Platform Manager provides diagnostics for System Platform Galaxy applications.
- Operations Control Logger displays the messages logged to the system logger.
  - Log Flag Editor can be used to assign a flag value to each message category issued by a System Platform component.
  - Log Monitor is a utility for monitoring the status of local and remote loggers

Other diagnostic programs

- Galaxy Repository diagnostic generation, and an associated diagnostic utility (aadr.EXE), can be utilized to collect and analyze data emitted by the aaGR process.
- AVEVA™ System Monitor tracks and manages the performance and availability of the AVEVA System Platform system and the related hardware and network infrastructure. The AVEVA System Monitor is accessible from the Start menu and includes integrated help documentation.

  A PDF version of the *System Monitor User Guide* can be accessed from the Start menu, under **AVEVA Documentation**.

Platform Manager

The Platform Manager is an extension snap-in to the Operations Control System Management Console (OCMC).

The Platform Manager provides diagnostics for System Platform Galaxy applications.

Platform Manager

Use the Platform Manager to perform run-time administrative tasks and diagnostics on System Platform Application Server platforms and engines.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

About Platform Manager

The Platform Manager provides diagnostics for System Platform Galaxy applications. You can view run-time status of some system objects and perform actions upon those objects. These actions include:

- Setting platforms and engines in an executable or idle mode.
- Starting and stopping platforms and engines.
- Removing platforms from the local computer.

The Platform Manager is an extension snap-in to the Operations Control System Management Console (OCMC). The OCMC is a Microsoft Management Console (MMC) container snap-in for all of the diagnostic and management utilities for your Galaxy application.

Other extension snap-ins include the Galaxy Database Manager, the Log Viewer, and the Operations Integration Server Manager.

Start Platform Manager

Platform Manager is a common component of a Galaxy application and it is available from any computer that has a deployed platform; therefore, you do not need to install it onto each node. This ensures that all nodes used within a Galaxy have access to Platform Manager.

With Platform Manager, you can access the platforms and engines deployed to the local computer and to any other computer in the Galaxy. Platform Manager does not require the Galaxy Repository to be installed on the local computer.

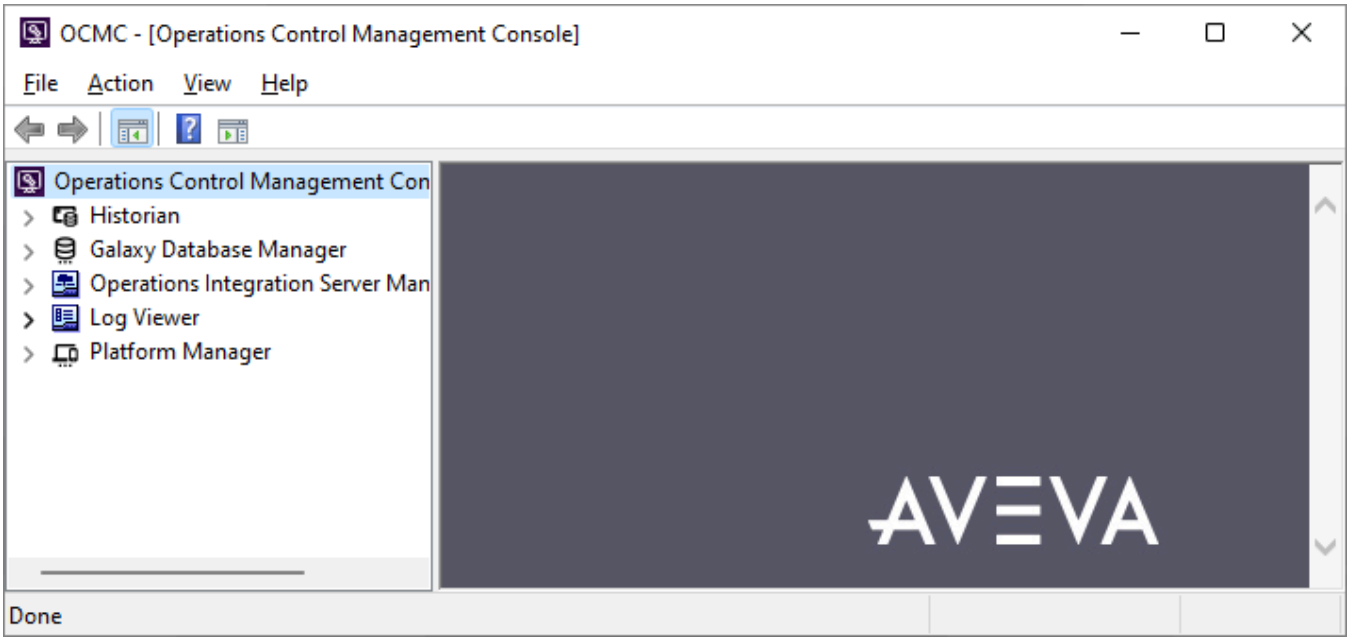
To start Platform Manager

1. Ensure your Galaxy application is running.
2. On the Windows **Start** menu, point to **Programs**, navigate to **AVEVA**, and then click **Operations Control Management Console**. If Platform Manager has security enabled, the Platform Manager **Login** dialog box appears.
3. Enter your user name and password.
4. If the configured security is OS User Based, select the domain from the **Domain** list.
5. Click **OK**.

Navigate in the Platform Manager

After successfully logging on, or if no security is enabled for Platform Manager, the Platform Manager snap-in appears under the Operations Control Management Console (OCMC) root node.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.



The left pane, or console tree has a Windows Explorer-type layout, with the Operations Control Management Console appearing as the root node and the Platform Manager appearing below this node. You can expand and collapse the console tree by clicking on the "+" or the "-" symbols that appear next to the snap-in.

The console tree shows the items that are available in the console. Other snap-ins that may appear below the OCMC node include the Galaxy Database Manager, the Log Viewer, and the Operations Integration Server Manager.

Security

For all System Platform administrative utilities, including Platform Manager, security is configured through the IDE. By default, there is no security enabled for Platform Manager or any of the other utilities.

There are four authentication modes for security that can be enabled for Platform Manager:

- No authentication
- Galaxy authentication mode
- OS User Based authentication mode
- OS Group authentication mode
- Authentication Providers mode

When no security is enabled from the IDE, the user is automatically logged into Platform Manager as **DefaultUser**. Without security, the logon dialog box does not appear when Platform Manager is launched and the user is granted all permissions.

If you enable security, make sure that users are given the level of permissions that will allow them to perform all necessary operations associated with their roles.

Galaxy authentication

Galaxy authentication requires the user to log on to Platform Manager every time the utility is started.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

OS user-based authentication

OS User Based authentication allows users who have matching OS accounts to log on, while all others are rejected.

OS group-based authentication

In OS group authentication, the user defines roles that match OS groups. At log on, the OS groups are matched with the roles.

For more information about security and authentication, see the *Application Server User Guide*.

Authentication providers mode

The **authentication providers** security mode uses the AVEVA Identity Manager (AIM) to create a unified security management infrastructure across your local System Platform nodes and VMs by leveraging operating system security and either CONNECT or Azure Active Directory (AD).

Users with matching AIM accounts are permitted to log on.

Change users

If security is enabled from the IDE, the user is logged in (either manually or automatically) when you open the Platform Manager.

If security is disabled in the IDE, clicking **Change User** results in no action. The command is ignored.

To log on as a new user

1. Select the Galaxy node.
2. On the **Action** menu, click **Change User**. A log-on dialog box appears.
3. Provide the requested log-on data and click **OK**. The new user is now logged on with only those permissions specific to that user.

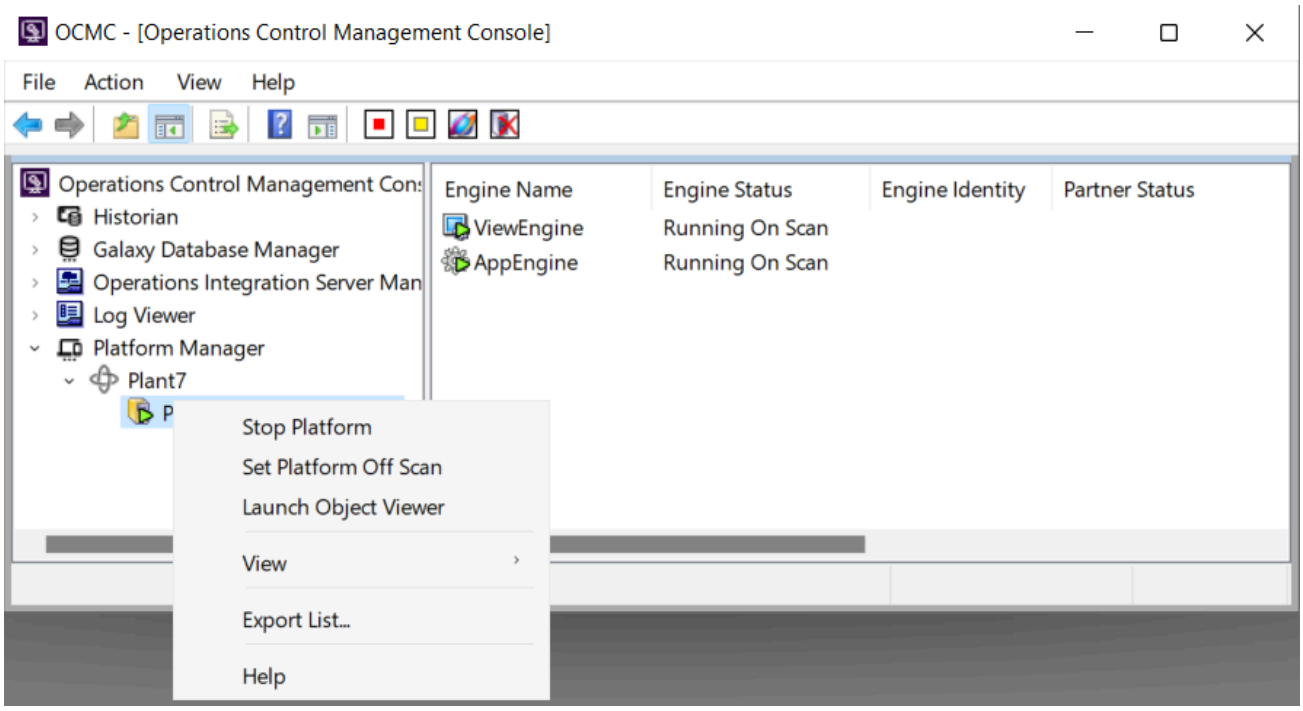
About Object Viewer

Object Viewer is an administrative utility that lets you view and modify an object's attribute values.

Object Viewer can be accessed by one of the following methods.

- Through the IDE or Platform Manager, while a platform or an engine is selected in either the console tree or the details pane.
- From a shortcut menu available by right-clicking a platform or engine.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.



- When you start Object Viewer from Platform Manager with a platform or engine selected, the utility starts and shows the attributes of the selected platform or engine. After Object Viewer is launched, it becomes independent from Platform Manager.

If Object Viewer is running, subsequent launches of Object Viewer from Platform Manager brings Object Viewer to the foreground with the selected platform or engine in focus.

For more information on Object Viewer, see the Object Viewer Online Help.

Use Platform Manager

Using Platform Manager, you can monitor and change the state of any platform or engine in your Galaxy application.

Platform and engine status

For diagnostic purposes, it is helpful to know if the platforms and engines are deployed, running, or are executing within a Galaxy application.

Object status description

All the states of platforms and engines are visible from Platform Manager, except for the **Not Deployed** state. However, some transitional states may occur so quickly that you may not be able to observe them. For example, the **Shutting down** state changes so quickly that you may not see this temporary transition.

Platforms and engines can have the following status.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Object status	Description
Not Available	Platform is powered off.
Not Deployed	Object is not deployed.
Deploying	Object is in the process of being deployed.
Undeploying	Object is in the process of being undeployed.
Shutdown	Object has been shut down normally.
Shutdown - Failed	Object has been shut down due to failure.
Shutting Down	Object is in the process of being shut down.
Starting On Scan	Object is in the process of starting in an execute mode.
Starting Off Scan	Object is in the process of starting in an idle mode.
Running On Scan	Object is running and executing.
Running Off Scan	Object is idle.

Note: The Platform Manager and Object Viewer utilities use different time-out functions for determining the status of an AppEngine on a remote node when a network connection is broken. This problem results in a Not Available status for the object. When viewing this changing status in the Platform Manager, the response time may be different from the Object Viewer. This is normal functionality.

On scan/off scan states

An object whose status is **On Scan** is performing its normal processing, while an object whose status is **Off Scan** is considered idle and not available for execution.

Some conventions apply for objects starting and running on scan and off scan. For example, engines can only run on scan when the status of the platform that hosts the engine is also running on scan. Or, platforms can only be shut down after the engines that are hosted by the platform are placed off scan.

Checkpointed state

When managing platforms and engines, the platform saves the last scanned state of an object to a local hard drive. This act is called checkpointing and the last scanned state is known as the checkpointed state. Checkpointing allows for the quick recovery of an object's state in the event of a failure without the dependence on the application's database or GalaxyRepository.

Platforms and engines that are started by Platform Manager are started in their checkpointed scan state. When stopping platforms and engines, their last scan state is checkpointed.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

View platform status

If you monitor the platform during its transition from a running state to a failed state, its status changes from **Running On Scan** or **Running Off Scan** to **Shutdown - Failed**. This indicates that the platform is in a shutdown mode and in a failed state.

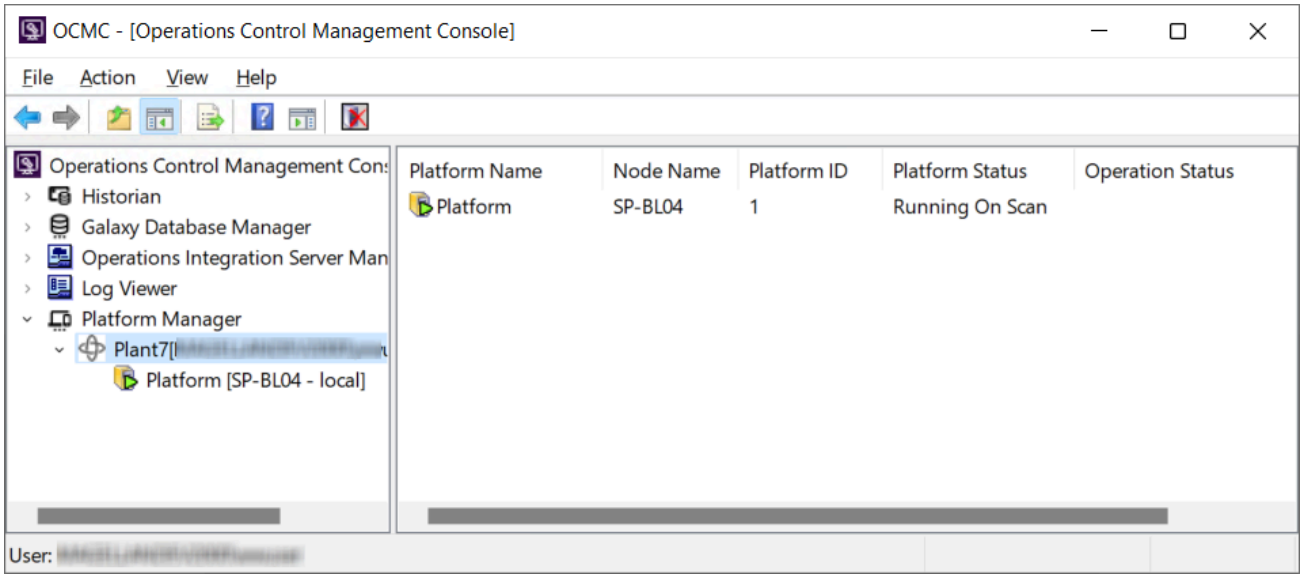
If an event occurs that causes a platform to fail, then the status of that platform is **Shutdown - Failed**.

Note: A network failure between platforms is not a platform failure, but instead, the platform is not available.

To view platform status

1. Expand the console tree and click the Galaxy name. In the console tree, a list of all platforms deployed within the application appears under the Galaxy name.

   The platform node name is listed in brackets after the platform name. A platform that is on the same node that Platform Manager is running on will have "local" appended to the node name. The platform name and node name also appear in the details pane.
2. In the details pane, locate the platform and view its status in the **Platform Status** column.
3. To set the order in which the platforms are displayed, click on the **Platform Name** or **Node Name** column header.



The following table describes each column:

Column	Description
Platform Name	The platform tag name.
Node Name	The computer name.
Platform ID	The platform's identifier when it was instantiated from a template.
Platform Status	The current status of the platform.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Column	Description
Operation Status	The progress of the transition after a command is placed on the platform. This changes throughout the transition until the stable state is achieved. An empty column indicates a successful transition.

View engine status

If you monitor the engine during its transition from a running status to a failed status, its status changes from **Running On Scan** or **Running Off Scan** to **Shutdown - Failed**. This indicates that the engine has failed and is not running.

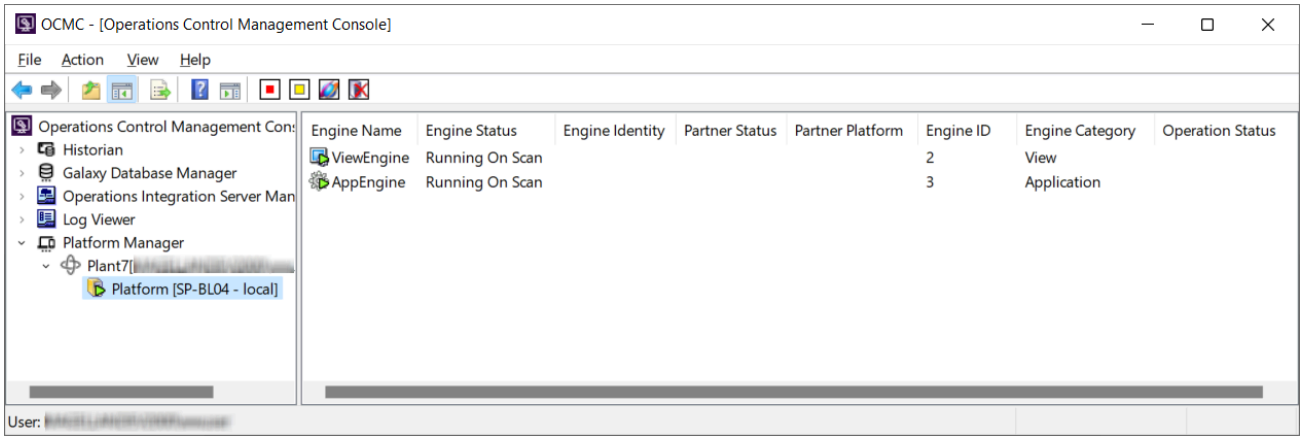
If an event occurs that causes an engine to fail, then the status of the engine is **Shutdown - Failed**.

If you have implemented redundancy, see Redundancy state descriptions for information about redundant engine status.

Details about each engine are listed in columns, and engines can be sorted by the information contained in each column. The identifying information for each engine includes Engine Name, Engine Category, Engine Status, Engine Identity, Partner Platform, and Partner Status. The latter three columns apply only to redundant engine pairs. The columns are described below.

To view engine status

1. Expand the console tree and locate the platform that hosts the engine you want to view.
2. From the console tree, select the platform. In the details pane, a list of all engines hosted by the platform appears.



3. In the details pane, locate the engine and view its status in the **Engine Status** column.

Each column is described in the following table. You can change the order in which the columns are displayed, and you can add or delete columns.

To change the columns, click on the platform name in the console tree, select the **View** menu, and then click on **Add/Remove Columns.** You can also bring up this option by right-clicking on the platform name.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Column	Description
Engine Name	The engine tagname.
Engine Status	The current status of the engine.
Engine Identity	Applies only to redundant pairs. A redundant engine can be either Active or Standby.
Partner Status	Applies only to redundant pairs. The current status of the partner engine.
Partner Platform	Applies only to redundant pairs. The name of the platform on which the partner engine is deployed. If the partner engine is undeployed or unassiged, there will be no value in the column (column will contain an empty string).
Engine Category	The type of engine (i.e., application, view, or DIEngine).
Engine ID	The engine's identifier when it was instantiated from a template.
Operation Status	The transitional state of an engine after a command is issued to it. The status will change until the stable state is achieved. An empty column indicates a successful transition.

Export a list

It may be helpful for you to export the resulting object status to a list. You can do this by using the MMC export function located from the console menu bar or on the console toolbar.

To export a list

- On the **Action** menu bar, select **Export List**. A dialog box appears, allowing you to save the items in the resulting pane as either plain text or Unicode text.

Manage platforms

After commanding a platform to a new state, you can view the transition status in the **Operation Status** column in the details pane.

In the console tree, the name of the platform will be preceded by an asterisk (*) if the status of the platform is anything other than Running On Scan. For example, if a platform is running off scan or if it is shut down, its name will be preceded by an asterisk.

The action of placing a platform on scan occurs when you place a platform in a run-time state in which it is performing its normal processing as part of its execute method. However, you can place a platform off scan,

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

which is the state that indicates that it is idle and not ready for execution. You can also control the startup and shutdown of platforms.

Sort platforms

You can change the order in which the platforms are listed in the detail pane. To sort the platforms, either click on the column that you want to use for sorting, or right click on the galaxy name in the console tree to bring up a context menu. Then, select the column name you want to use for sorting the platforms from the menu. Platforms can be sorted by:

- Platform Name
- Node Name
- Platform ID

See View platform status for additional information about these columns.

Set a platform on scan

To set a platform on scan, its status must be **Running Off Scan**. Since the platform is running off scan, the platform name will be preceded by an asterisk (*) in the console tree.

To set a platform on scan

- In the details pane, right-click the platform, and then click **Set Platform On Scan.**

  The status for the platform changes from **Running Off Scan** to **Running On Scan**, which indicates the platform is performing normally. The asterisk will no longer appear next to the platform name in the console tree.

Set a platform off scan

To set a platform off scan, its status must be **Running On Scan** and the status of any engines hosted by the platform must be either **Running Off Scan** or **Shutdown**.

To set a platform off scan

1. In the details pane, right-click the platform, then click **Set Platform Off Scan**.
2. If an engine hosted by the platform has an on scan status, a dialog box appears informing you that continuing with the operation that places the platform off scan also puts the engines off scan.
3. Click **No** to cancel the operation or click **Yes** to put the engines off scan before setting the platform off scan.
4. The status for the platform changes from **Running On Scan** to **Running Off Scan** and the platform becomes idle. The platform name is now preceded by an asterisk.

Start a platform

After a platform is stopped, it can be restarted by Platform Manager in either the on scan or off scan state if its Start Up mode is configured to **Manual Start**. You can verify the platform's configuration from the IDE.

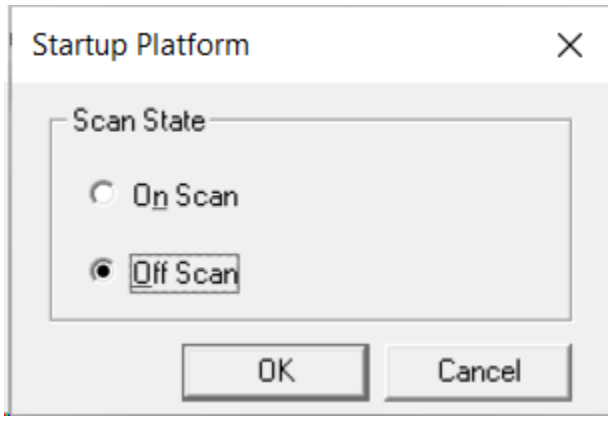
If a platform's Start Up mode is configured to **Auto Start**, then it starts automatically in its last checkpointed scan

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

state when the power is reapplied to the computer.

To start a platform

1. From the details pane, right-click the platform, and then click **Start Platform**. The **Startup Platform** dialog box appears.



2. Select **Off Scan** or **On Scan,** and then click **OK** to set the platform's startup scan state.

   The platform begins to start and its status changes from **Shutdown** to **Starting Off Scan** or **Starting On Scan**. After the platform starts successfully, its status changes from **Starting Off Scan** or **Starting On Scan** to **Running Off Scan** or **Running On Scan**. If the platform is running off scan, an asterisk will precede the platform name in the console tree. No asterisk will be displayed if the platform is running on scan.

   The status of all hosted engines changes from **Shutdown** to **Running Off Scan**.

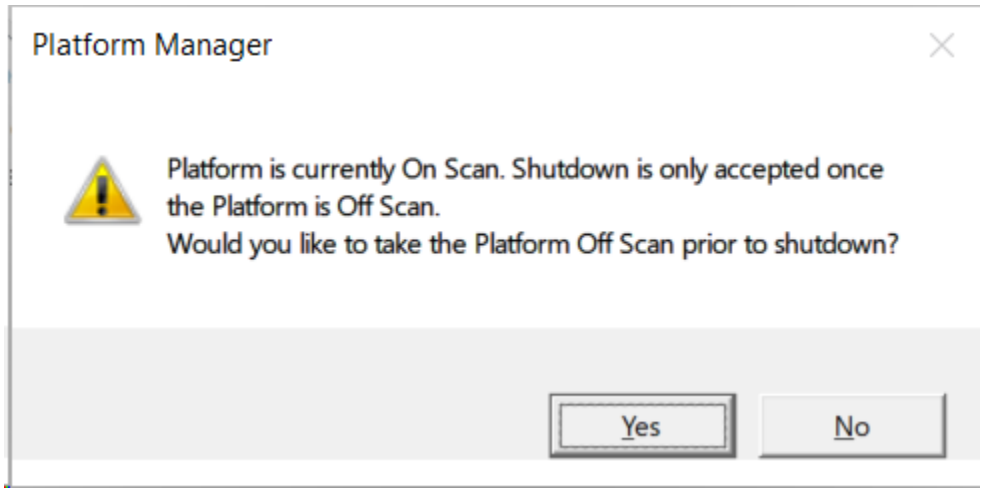
Shut down a platform

To shut down a platform, the status of the platform must be **Running Off Scan**. If the status of any of its hosted engines are **Running On Scan**, then the hosted engines are first placed off scan.

To shut down a platform

1. From the details pane, right-click the platform, and then click **Stop Platform**.

   If the status of the platform is **Running On Scan**, you are prompted to take the platform off scan before shutdown.



© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

2. Click **No** to cancel the operation or click **Yes** to continue shutting down the platform.

   If you select **Yes** and the status of an engine on the platform is on scan, then the engine is placed off scan and then shut down.

   As the platform shuts down, its status changes from **Running Off Scan** to **Shutting Down**. If the platform shuts down successfully, its status changes from **Shutting Down** to **Shutdown**. The target platform is shut down and the platform's last scan state is checkpointed to **Off Scan**.

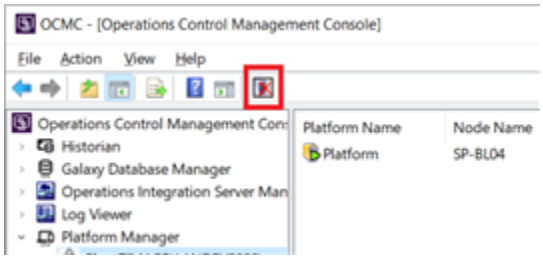
Remove a local platform

Use this procedure to remove an inactive local platform prior to upgrading System Platform, to remove an orphaned local platform, etc.

Note: You cannot undo this operation, and you cannot add a platform through the Platform Manager. You must use the IDE to add a platform.

To remove a local platform

1. In the console tree, select the local platform that you want to remove.
2. Click the **Remove Platform** icon on the toolbar. You will be prompted to confirm that you want to remove the platform.



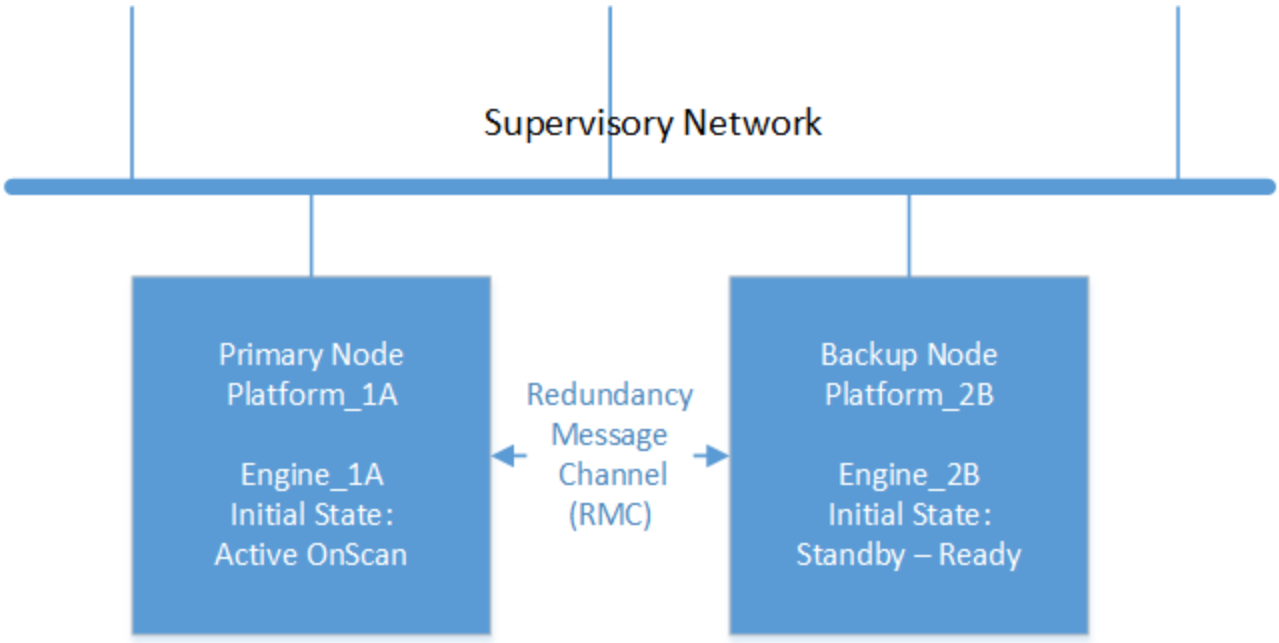
3. Click **No** to cancel the operation or click **Yes** to remove the platform.
4. If you click **Yes**, the local platform is removed and the database is cleaned up.
5. If the IDE is running on the node where you removed the platform, restart the IDE.

Restart redundant platforms

Redundant AppEngines are hosted on redundant platform objects. Some maintenance tasks, such as applying security updates, may require that you shut down and restart redundant platforms. Additional information about redundancy and redundancy states can be found under Manage engines.

Before proceeding, close any open System Platform applications on the redundant platforms, such as the IDE, InTouch WindowMaker, InTouch WindowViewer, and AVEVA OMI ViewApps. In this example, the primary node is Platform_1A which hosts Engine_1A, and the backup node is Platform_2B which hosts Engine_2B. Engine_1A is active and Engine_2B is standby.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.



Note: The redundancy message channel (RMC) allows the redundant platforms to communicate directly with each other. For more information, see the *Application Server User Guide*.

To restart redundant platforms

1. In the console tree, select the platform that hosts the active engine.
2. In the details pane, select the engine.
3. When the engine is highlighted, click the **Force Failover** button or right-click and then select the failover option from the context menu. See Redundancy state descriptions for details.

   The partner engine status will show "Switching to Active."

Engine Status: Running On Scan	**Partner Status: Switching to Active**

   After failover, the primary engine status changes to Standby - Ready and the partner engine status changes to "Active."

Engine Status: Standby - Ready	**Partner: Active**

4. Stop the selected platform.

   After stopping the platform, the status of the primary engine changes to shutdown, and no partner status is shown, even though the partner engine is now active on scan. Since the platform is not running, it cannot see the status of its partner. See Shut down a platform for details about stopping the platform.

Engine Status: Shutdown	**Partner Status: No status shown**

5. **Maintenance:** Peform any required maintenance tasks on the node that is shutdown.
6. Restart the platform by setting the platform On Scan. See Start a platform for details.

   After restarting the platform, the status of the primary engine changes to back to "Standby - Ready," and the partner status shows again as "Active."

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Engine Status: Standby - Ready	Partner Status: Active

Note: If there are additional engines hosted on the platform that are running Off Scan, set them to On Scan.

7. When all engines are running on scan, force failover back to the primary engine(s). The status of the primary engine changes to "Running On Scan" and the partner engine status changes to "Standby - Ready."

Engine Status: Running On Scan	Partner Status: Standby - Ready

8. In the console tree, select the platform that hosts the standby engine. In the example above, this Platform_2B. Repeat steps 4, 5, and 6 to update the Backup Node.

Manage engines

You can view the transition status of an engine from the **Operation Status** column in the details pane after changing an engine to a new state.

The action of placing an engine on scan occurs when you place an engine in a run-time state in which it is performing its normal processing as part of its execute method. However, you can place an engine off scan, which is the state that indicates that it is idle and not ready for execution. You can also control the startup and shutdown of engines.

If you are using redundancy, you can control the run-time state of the redundant pairs the same way you would for a standalone engine. You can also initiate a forced failover, where the active engine is placed in standby status and the standby engine becomes active.

Engine status icons

Platform Manager provides icons to represent the status and identity of engines at a glance. The engine status icon indicates whether it running on scan, off scan, on standby, or shutdown. The status icon overlays the engine identity icon; the identity icon indicates if the engine is primary or backup. Icons are not displayed for partner engines if the engine is part of a redundant pair.

Icon	Description
	Application engine (not redundant)
	Primary engine (redundant pair)
	Backup engine (redundant pair)
	Engine running on scan
	Engine running off scan

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Icon	Description
	Engine on standby
	Engine is shutdown

Redundancy state descriptions

If you have configured redundant engines, the status both partner engines (active and standby) will be shown. Additionally, the redundancy status of the partner engine is listed with the partner engine name. The name of the platform on which the partner engine is deployed is listed as well.

You will also be able to see if an engine is running on-scan or off-scan, its standby status, and upgrade status.

Note: When the local engine is shutdown, no partner status will be shown (field will be empty).

Engine status	Redundancy status	Description
Standby - Ready	Active	Engine is running and can communicate with its partner engine.
Running On Scan	Active - Standby Not Available	Engine is running but cannot communicate with its partner engine.
Running On Scan	Active - Upgrade	Partner engine software must be upgraded.
Determining Failover Status	Determining Failover Status	Engine is starting but is not yet able to resolve the state of its partner engine.
Standby - Ready	Standby - Ready	Standby engine is available and can switch to active if active engine fails or if commanded.
Standby - Missed Heartbeats	Standby - Missed Heartbeats	Standby engine is not receiving heartbeats from its active partner engine, but the number of missed heartbeats is not currently sufficient to establish that failure has occurred.
Standby - Not Ready	Standby - Not Ready	Standby engine cannot communicate with the active engine, or engines are out of sync.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Engine status	Redundancy status	Description
Standby - Syncing Code	Standby - Syncing Code	Standby engine is synchronizing code modules with active engine.
Standby - Syncing Data	Standby - Syncing Data	Standby engine is synchronizing all object-related data with active engine, and will now transition to Standby - Ready state.
Standby - [Ready] [Not Ready] [Not Available]	Switching to Active	Backup engine has been commanded to take over as active engine.
Switching to Standby	Switching to Standby	Active engine has been commanded to switch to standby engine.
Standby Not Available	Standby Not Available	Partner engine has crashed or has been shutdown.
Standby Not Available	Standby Not Available	Communication with the partner engine has been lost.

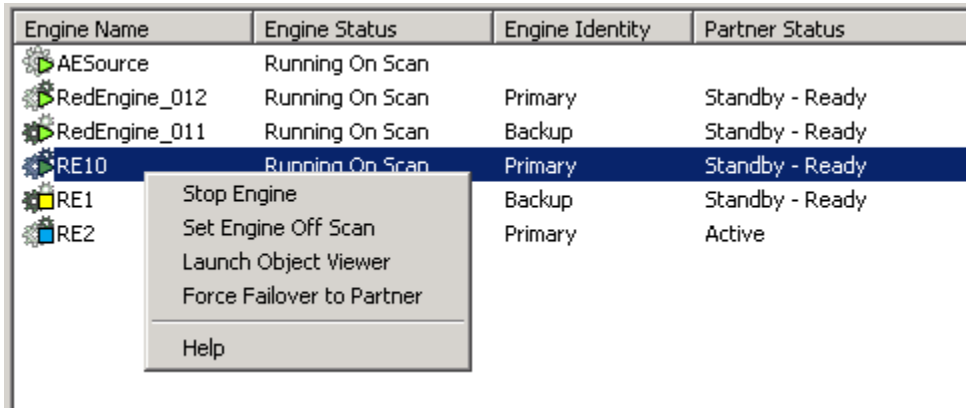
To force failover to partner engine

Note: Force failover applies only to redundant engines. It can only be invoked when the active engine is running (either on- or off-scan) and the status of the partner engine is Standby - Ready.

1. Select the engine to failover.



2. When the engine is highlighted, click the **Force Failover** button. Alternatively, you can right click to display a context menu, and then click **Force Failover to Partner**.



© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Set an engine on scan

To set an engine on scan, the status of the platform that hosts the engine must be **Running On Scan**.

To set an engine on scan

- From the details pane, right-click the engine, and then click **Set Engine On Scan**.

  The status for the engine changes from **Running Off Scan** to **Running On Scan**, which indicates the engine is performing normally. Any hosted ApplicationObject is also set on scan, you can verify this using Object Viewer.

Set an engine off scan

To set an engine off scan, its status must be **Running On Scan**.

To set an engine off scan

- From the details pane, right-click the engine, and then click **Set Engine Off Scan**.

  The status for the engine changes from **Running On Scan** to **Running Off Scan** and the engine becomes idle. All hosted ApplicationObjects are also set off scan. You can verify this using the Object Viewer.

Start an engine

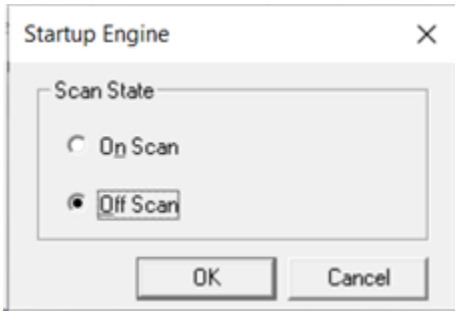
To start an engine, the status of the platform that hosts the engine must be **Running On Scan** and the engine's Start Up mode must also be configured to **Manual Start**. You can verify the engine's configuration from the IDE.

If the engine's Start Up mode is configured to **Auto Start**, then the platform starts the engine in the last checkpointed scan state. Engines configured in the **Semi Auto Start** mode start in the **Off Scan** state.

To start an engine

1. From the details pane, right-click the engine, and then click **Start Engine**.

   The **Startup Engine** dialog box appears.

2. Click **Off Scan** or **On Scan.**
3. Click **OK** to set the engine's startup scan state.

   The engine begins to start and its status changes from **Shutdown** to **Starting Off Scan** or **Starting On Scan**. After the engine starts successfully, its status changes from **Starting Off Scan** or **Starting On Scan** to **Running Off Scan** or **Running On Scan**.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Shut down an engine

If you try to shut down an engine whose status is **Running On Scan**, it is set to **Running Off Scan** before shutting down.

To shut down an engine

1. In the details pane, right-click the engine, and then click **Stop Engine**.

   If the status of the engine is **Running On Scan**, you are prompted to take the engine off scan before shutdown.
2. Click **No** to cancel the operation or click **Yes** to continue shutting down the engine.

   If you click **Yes**, the engine shuts down and its view changes from **Running Off Scan or Running On Scan** to **Shutting Down**. If the engine shuts down successfully, its status changes from **Shutting Down** to **Shutdown** and the engine's last scan state is checkpointed to **Off Scan**.

Galaxy Repository diagnostic generation

Diagnostic information is continuously sent by the Galaxy Repository service to a shared memory segment that functions as a circular buffer. This shared memory segment is backed by a file on disk, and can be used to troubleshoot error conditions. The shared memory segment is designed with a small footprint to minimize any impact to the aaGR process. By default, the shared memory segment uses only 8192 bytes, allowing it to hold 4096 double-byte diagnostic data. The shared memory segment is accessible through the global ID "Global\0xdeadbeef:PS".

The backing file to the shared memory segment is created in the ArchestrA temporary folder, although a different location can be specified as an override. Overrides to the operational constraints to diagnostic generation, including the backing file location, may be specified in the Windows Registry, under the location:

HKLM\SOFTWARE\Wow6432Node\ArchestrA\Framework\GR\Diagnostics

The diagnostic configuration registry values are as follows:

- File [REG_SZ]: Fully qualified path to the location of the backing file

  Default: The temp file is located in the ArchestrA temp folder.
- ID [REG_SZ]: Unique global identifier to associate with the shared memory segment.

  Default: Global\0xdeadbeef:PS

  Note: The string ":PS" is appended automatically to the value of the user-specified ID.
- Size [REG_DWORD]: Size of the shared memory segment in double-byte characters

  Default: 4096

  Note: The recommended size is 4096, but should never be set to exceed 8192.
- Append [REG_DWORD]: If true (1), diagnostic information is appended to the backing file when the aaGR process restarts. If false (0), the backing file is recreated during a restart.

  Default: true (1)
- FlushCadence [REG_DWORD]: Determines when the contents of the shared memory segment are flushed to disk. FlushCadence accepts an unsigned long (DWORD) that represents a [wide] character count. When the number of characters written to the memory-mapped file (as measured from the last time that it was flushed to disk) equals the stipulated FlushCadence, the contents of the shared memory segment are flushed

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

to disk. A FlushCadence value of 0 causes the flush to occur when end-of-segment is reached, and before the write wraps to the beginning of the segment.

Default: 0

Note: The default setting of 0 means that flush occurs when the end of the segment is reached, and before writing resumes at the beginning of the segment. If set to a non-default value, the dirty contents of the shared memory segment are flushed to disk when the contents of the segment crosses the FlushCadence value.

aaGR diagnostic utility (aadr.EXE)

A specialized utility, aadr.EXE, may be installed and launched by an AVEVA support engineer to consume the diagnostic information sent by the aaGR process, allowing for the capture of a virtually unlimited set of data. The aadr utility preserves consumed diagnostic data in a collection of files stored at a user-specified location on disk. The size, number, and location of these files is configurable through registry settings under:

HKLM\SOFTWARE\Wow6432Node\ArchestrA\Framework\GR\Diagnostics\Consumer

The following registry values must be specified for the aadr utility to operate:

- Directory [REG_SZ]: Fully qualified path to a directory on disk where the consumed diagnostic data is to be collected.
- FileSize [REG_DWORD]: Size, in double-bytes, of each file in the collection.
- NumFiles [REG_DWORD]: Total number of files in the collection.

In addition to the IDE and galaxy events sent by the aaGR process, information captured by the aadr utility includes information about the aaGR process. This information includes baselines for, and changes to, the allocated memory, and the open handle and thread counts.

Troubleshoot a remote galaxy connection

Informational and error messages remain the same in a multi-galaxy environment as they are for single-galaxies. The diagnostic tools already available, such as Object Viewer and the OCMC Logger, continue to function in a multi-galaxy environment as they do in a single-galaxy environment.

Object Viewer displays communication or configuration errors, and displays data quality. The OCMC Logger displays informational messages about system events.

Important: Watchdog and other ASB core services must be running whether or not you are operating in a multi-galaxy environment. Galaxy deletion, configuration, or run-time operations can fail if ASB services are not running, even when not operating in a multi-galaxy environment. See Local services in the following troubleshooting table for more information.

Indications of communication or configuration errors can include:

- Galaxies in a multi-galaxy environment do not appear in the **Galaxy:** list box in the **Galaxy Browser**.
- **Attribute References** in an Object Viewer watch list show Bad quality.
- **Attribute References** in an Object Viewer watch list show communication errors.
- Specifying an **Attribute Reference** in the Object Viewer text box returns a configuration error.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

- Specifying an attribute reference in an InTouch application returns a configuration error or fails to return run-time data.
- A galaxy in the multi-galaxy environment does not appear in the **Browse Node** dialog box when attempting to select a Galaxy Repository with which to pair.
- You cannot connect to a galaxy in the multi-galaxy environment.
- You cannot undeploy or deploy an instance of an Application Server Service.
- Writes from InTouch WindowViewer fail when security is enabled.

When communication or configuration issues arise in a multi-galaxy environment, you can perform the following series of quick checks and remedies to determine if the issues have been caused by a multi-galaxy configuration error.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Check ...	Description	Procedure
Local services	The AVEVA Watchdog Service is a utility to start all the System Platform Core Services. The Watchdog Service starts automatically when you start your operating system. The Watchdog Service and other services are tightly coupled with Application Server configuration and run-time operations. • The Watchdog Service must be running in order to create service instances and nodes, to deploy, undeploy, or delete user-defined services. • The Watchdog service must be running whether or not you are using a multi-galaxy environment.	Do not stop the Watchdog service even if you do not plan to use a multi-galaxy environment. These steps use general Windows operating system for illustration. 1. From the Windows Start menu open **Windows Administrative Tools**, then select **Component Services**. The **Component Services** utility opens. 2. Open **Services (Local)**. 3. Select the **AVEVA Watchdog Service** to view its state. 4. If the Watchdog service is not running, **Start the service**. 5. Close the **Component Services** utility.
Service discovery configuration	By default, the Local Galaxy Server primary node is not configured (blank), and must be designated before you can enable galaxy pairing. By default, the Cross Galaxy Server primary node is not configured (blank), and must be set to the designated Cross Galaxy Server before you can enable galaxy pairing. There can be only one Cross Galaxy Server in a multi-galaxy environment. All galaxies in the environment must designate the same Cross Galaxy Server primary node.	Do these steps for each galaxy in the multi-galaxy environment for which you are having connectivity issues. 1. From the IDE ribbon, select **Galaxy**, then **Configure**. 2. Select **Communication**, then **Cross-galaxy server**. 3. Verify or configure the **Local Galaxy Server Primary node** as the local galaxy GR node. 4. Verify or configure the **Cross Galaxy server Primary node** as the node you have defined as the Cross Galaxy Server for all paired galaxies.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Check ...	Description	Procedure
Multi-Galaxy configuration	Multi-Galaxy configuration establishes and maintains the trust relationship between paired galaxies for the duration of the pairing session. Remote pairing must be enabled on each GR to be paired, each of which requires the same unique passphrase.	1. From the IDE ribbon, select **Galaxy**, then **Configure**. 2. Select **Communication**, then **Cross-galaxy server** to open **Multi-Galaxy configuration** page. 3. Verify that the galaxy or galaxies to be paired are listed in the **Galaxy Repository** list box. a. If a particular galaxy is not listed, click the **Add** button to open the **Select Galaxy Repository to pair with** dialog box. b. Enter the **Target Galaxy Repository node** in the text box. c. Enter the **Passphrase** in the text box. 4. If pairing fails, ensure that remote pairing is enabled on the target GR, and that you have the correct passphrase entered.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Check ...	Description	Procedure
Services configuration	The user-configurable Application Server Services provide communication channels for attribute browsing and run-time data access. By default one instance of each service is configured and locally deployed. At least one instance of each service must be deployed on each galaxy in the multi-galaxy environment for multi-galaxy communication to function. **Note**: When you set or change the Local Galaxy Server or the Cross Galaxy Server, the core ASB services automatically restart. This can create a timing issue if you immediately deploy a platform. The ASBMxDataProvider and the ASBAuthentication services can fail to deploy. Run-time connectivity would then fail in the multi-galaxy environment.	For more information about configuring Application Server Services, see Configure and deploy PCS services. For browsing issues, verify the ASBGRBrowsing service instance. For run-time data subscription issues, verify the ASBMxDataProvider service instance on the publisher side, and ensure that the galaxy is deployed. For secured write issues, verify the ASBAuthentication service instance. 1. From the IDE ribbon, select **Galaxy**, then **Configure**. 2. Select **System**, then **Services** to open the **Configure Services** dialog box. 3. Expand the GR node in the hierarchy pane to view the installed services. Expand each service in the hierarchy to view the service instances. a. Select the instance to view its configuration in the configuration pane. b. Verify that the **Galaxy Name** is correct (ASBGRBrowsing service only). c. Verify that the service instance is assigned to a node, visible in the **Assignments** pane, and that the node is correct. d. Verify that the service instance is deployed. 4. Edit the service instance configurations as necessary.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Check ...	Description	Procedure
Client configuration	The Galaxy Browser, Object Viewer, and Managed InTouch applications can function as clients to browse attributes in the System Platform namespace, or to subscribe to run-time data. Using these clients in a multi-galaxy environment is described in Access multiple galaxies.	For Object Viewer communication errors, review the Attribute Reference syntax. In particular, ensure that the correct <GalaxyName>: prefix is present and points to the correct galaxy. For Galaxy Browser errors, ensure that you have selected the correct galaxy in the **Galaxy:** list box. For Managed InTouch errors: • Ensure that you have selected the correct galaxy when accessing tagnames and attributes from the Galaxy Browser. • Ensure that syntax is correct, including the "Galaxy:" access name prefix, and the full attribute reference in quotes, with the remote galaxy prefix as part of the expression. Correct any incorrect syntax. **Note**: If you selected InTouch HMI **Development and Run Time** during installation, you cannot create a multi-galaxy environment. This installation does not include a GR, required for multi-galaxy functionality.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Check ...	Description	Procedure
Security configuration	OS User with domain users only and OS Group (supports only domain users) are the only security models supported in a multi-galaxy environment. Security issues, evident with failed writes, can occur for any of the following possible causes: • Security mode of galaxy is set to Galaxy Security • Security mode of InTouch is not set to ArchestrA • User has not logged into the remote Galaxy at least once • Security modes of local and remote galaxies don't match • User doesn't have sufficient permissions to perform the write • Default User Authentication service is not deployed on the GR node	For information about security configuration, see Configure security. In the IDE, set the security mode to **Galaxy Security**. In InTouch HMI, set the security mode to **ArchestrA**. In the multi-galaxy environment, set local and remote galaxy security to the same mode for each galaxy. In the multi-galaxy environment, log into each remote galaxy at least one time (for each user who requires access to those galaxies and has sufficient write permissions). In the Configure Services dialog, verify the ASBAuthentication service instance. 1. From the IDE, select the **Galaxy** menu, then **Configure**. 2. Select **System**, then **Services** to open the **Configure Services** dialog box. 3. Expand the GR node in the hierarchy pane to view the installed services. Expand the each service in the hierarchy to view the service instances. a. Select the instance to view its configuration in the configuration pane. b. Verify that the service instance is assigned to a node, visible in the **Assignments** pane, and that the node is correct. c. Verify that the service instance is deployed. 4. Edit the service instance configurations as necessary.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Check ...	Description	Procedure
Unpairing galaxies	Unpairing galaxies requires that communication is established with the remote node to be unpaired. If communication with a remote galaxy is broken, unpairing with that node will fail. This condition can occur in different scenarios, including: • The remote node is down or not on the network. • The System Authentication service is not running. • The Watchdog service is not running • A mismatch or corruption has occurred in a security key.	When an unpair operation fails, you will see the **Un-pair operation with remote Galaxy Repository failed** error message and dialog box. This dialog box offers you a choice to force the unpair, or to wait until the remote GR is back online. • Click **Yes** to force the unpair. The forced unpairing occurs only on the local GR. When the remote GR is back online, any client running on the remote GR will be unaware that an unpairing has occurred, and will still be able to discover and connect to the service on the local GR from which you performed the failed unpairing operation. • Click **No** to end the unpairing operation and wait until the remote GR is back online. When you click **No**, the system takes no action, and pairing remains as configured. You can now troubleshoot why the remote GR is offline, then retry the unpairing operation.

Connect to a remote GR from the IDE

As of System Platform 2017, communication from a GR with a local IDE (single node with both IDE and GR) is blocked by default to a remote GR. This applies to using the IDE to configure a Galaxy on a remote GR, as well as runtime.

If the IDE you are running has a GR on the same node and you try to pair with a Galaxy on a remote GR, you will get an error message that communication with the GR node cannot be established.

Scenario 1

The nodes have the following configuration:

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

- Node 1: IDE + GR (Galaxy 1)
- Node 2: GR only (Galaxy 2)
- Node 3: IDE only

To establish communication with the blocked GR in Scenario 1

1. Open the IDE on Node 1 and connect to Galaxy 1.
2. Configure Service Discovery for Node 1, and set Node 1 as the Local Galaxy Server and Cross Galaxy Server (**Galaxy > Configure > Communication > Cross-galaxy server**).
3. Open the IDE on Node 3 and connect to Galaxy 2.
4. Configure Service Discovery for Node 2, and set Node 2 as the Local Galaxy Server. Set Node 1 as the Cross Galaxy Server (**Galaxy > Configure > Communication > Cross-galaxy server**).
5. Configure Multi-Galaxy for each Galaxy. Use the Node 1 IDE to configure Multi-Galaxy for Node 1, and the Node 3 IDE to configure Multi-Galaxy for Node 3.

   Make sure that you use the same Passphrase for both nodes.

Scenario 2

The nodes have the following configuration:

- Node 1: IDE + GR (Galaxy 1)
- Node 2: IDE + GR (Galaxy 2)

To establish communication with the blocked GR in Scenario 2

1. Open the IDE on Node 1 and connect to Galaxy 1.
2. Configure Service Discovery for Node 1, and set Node 1 as the Local Galaxy Server and Cross Galaxy Server (**Galaxy > Configure > Communication > Cross-galaxy server**).
3. Open the IDE on Node 2 and connect to Galaxy 2.
4. Configure Service Discovery for Node 2, and set Node 2 as the Local Galaxy Server. Set Node 1 as the Cross Galaxy Server (**Galaxy > Configure > Communication > Cross-galaxy server**).
5. Configure Multi-Galaxy for each Galaxy. Use the Node 1 IDE to configure Multi-Galaxy for Node 1, and the Node 2 IDE to configure Multi-Galaxy for Node 2.

   Make sure that you use the same Passphrase for both nodes.

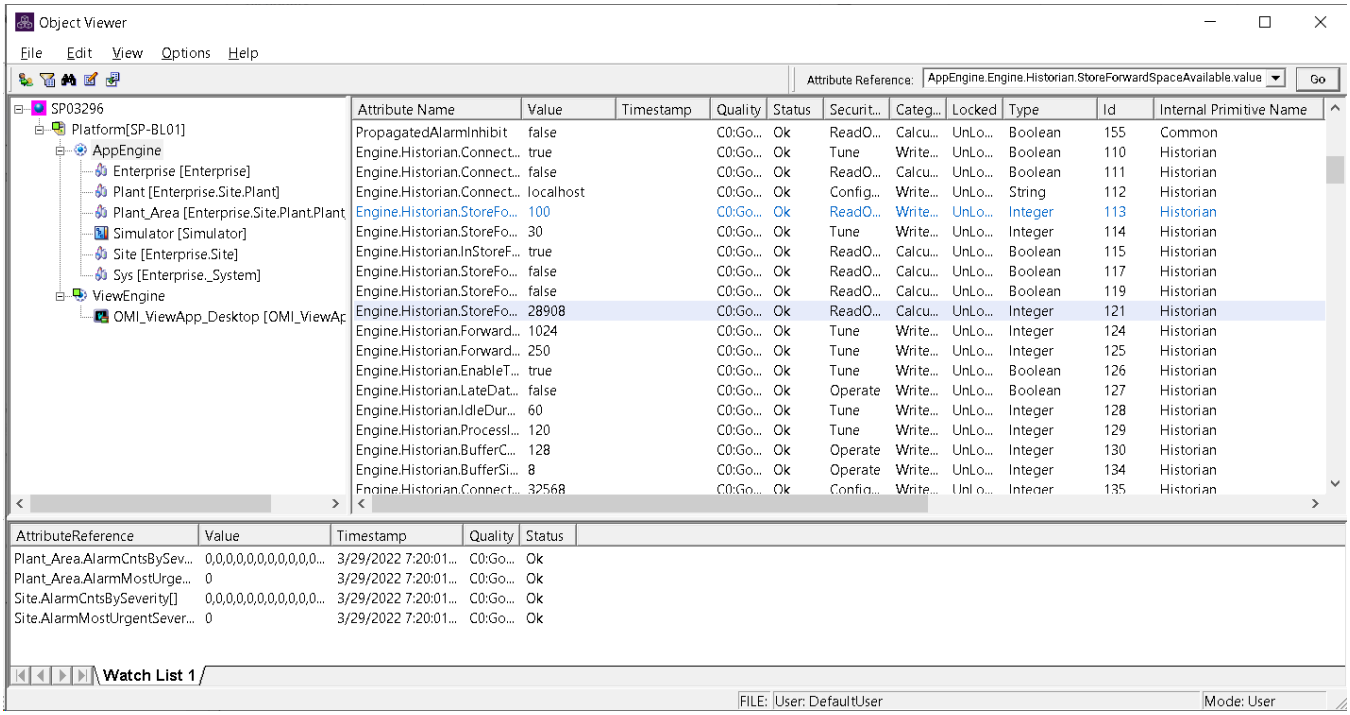
Object Viewer advanced diagnostic information

Object Viewer provides performance and diagnostic information about ApplicationObjects. You can use Object Viewer to show performance parameters, resource consumption, and reliability measurements of objects you select.

In addition to viewing an object's data value, data quality, and the communication status of the object, you can modify some of the object's attributes for diagnostic testing. Modifications can include adjusting timing parameters and setting objects in an execution or idle mode.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Typically, you will open Object Viewer from the IDE or Windows Start menu. Since Object Viewer is a common component of Application Server, you can open it from any node within your Application Server environment. For more information, refer to the *Object Viewer User Guide* Object Viewer overview.



Advanced diagnostics

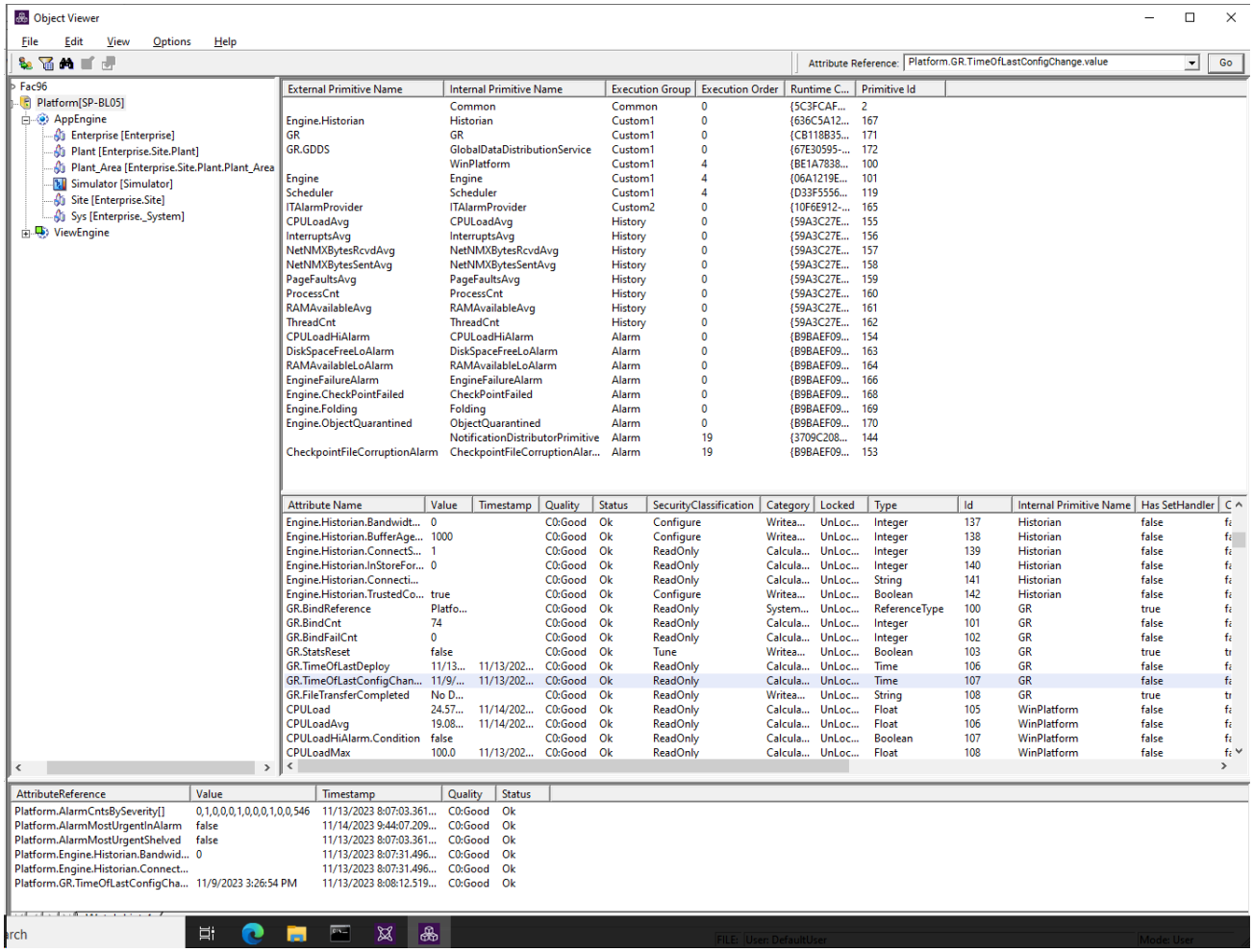
You can use Object Viewer to obtain advanced diagnostic information by running it from the command line. Use the switch /New to display additional diagnostic information, such as internal and external primitive names, execution order, run-time class IDs, and primitive ID.

To open Object Viewer from the command line:

1. Open a command prompt.
2. Navigate to:
   C:\Program Files (x86)\ArchestrA\Framework\Bin
3. Enter the command
   `aaObjectViewer.exe /New`

Note: Since Object Viewer is a run-time tool, there must be a deployed galaxy before you can run it.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.



© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Scripting Reference

The topics in this section contain:

- General information about using scripts in Application Server
- Descriptions of the QuickScript .NET functions, operators, variables, and control structures
- Some sample QuickScript .NET scripts

Common script environment

This section describes common styles, syntax, commands, and behaviors of scripts within AVEVA™ Application Server.

Script editing styles and syntax

Application Server supports two types of scripts:

- Simple scripts can perform assignments, comparisons, simple math functions, and similar actions. Simple scripts are described in this section.
- Complex scripts can perform logical operations using conditional branching with IF-THEN-ELSE type control structures.For more information about complex control structures, see the *Application Server Scripting Reference*.

Both single and multi-line comments are supported. Single-line comments start with a single quotation mark (’) at the beginning the line but do not require a single quotation mark at the end of the line. Multi-line comments are enclosed within brace characters ({ }), and can span multiple lines.

White space rules apply for space and indention. Indent using spaces, or the TAB key. Individual statements are indicated by a semicolon marking the end of the statement.

Required syntax for expressions and scripts

The syntax in scripts is similar to the algebraic syntax of a calculator. Most statements are presented using the following form:

```
a = (b - c) / (2 + x) * xyz;
```

This statement places the value of the expression to the right of the equal sign (=) in the variable location named "a."

- A single entity must appear to the left of the assignment

  operator =.
- The operands in an expression can be constants or variables.
- Statements must end with a semicolon (;).

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Entities can be concatenated by using the plus (+) operator. For example, if a data change script such as the one below is created, each time the value of "Number" changes, the indirect entity "Setpoint" changes accordingly:

```
Number=1;
Setpoint = "Setpoint" + Text(Number, "#");
```

Where the result is "Setpoint1."

Simple scripts

Simple scripts implement logic such as assignments, math, and functions. An example of this type of scripting is:

```
React_temp = 150;
ResultTag = (Sample1 + Sample2)/2;
{this is a comment}
```

Script execution types

This section describes the script execution types supported by Application Server and OMI.

- Startup scripts
- OnScan scripts
- Execute scripts
- OffScan scripts
- Shutdown scripts
- Deployment scripts

Scripting Redundant AppEngines

There are certain considerations that you must take into account when writing a script that will run on redundant AppEngines. This section outlines whether or not a script will run under various scenarios, including deploy, forced failover, system failure, system startup, and undeploy operations. Redundant engines can be set to run in either **Legacy Mode** or **Run Warm Mode**. The selected mode may change the circumstances under which a script will execute.

Run Warm Mode provides much faster failover performance, but there are internal differences between the redundancy modes in how the engines start and stop. Therefore, Startup and Shutdown scripts for redundant engines may operate differently, depending on which redundancy mode is selected.

Note: New redundant engines default to Run Warm Mode. Redundant engines in migrated galaxies default to Legacy Mode.

The following table summarize the circumstances under which each script execution type runs when redundancy is set to **Legacy Mode** (differences between the two modes are highlighted). The abbreviations used to denote the script execution types are as follows:

Abbreviations used below for Script execution types:

U - StartUp
N - OnScan
X - Execute
F - OffScan

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

D - ShutDown

Note that there are no differences for script execution between redundancy modes on the primary engine. There are differences in how scripts execute on the backup engine.

Legacy mode	Primary engine (on server 1)		Executes on server 1					Backup engine (on server 2)		Executes on server 2				
Action	**Initial state**	**End state**	**U**	**N**	**X**	**F**	**D**	**Initial state**	**End state**	**U**	**N**	**X**	**F**	**D**
Deploy	Down	Active OnScan	Y	Y	Y	N	N	Down	Standby	N	N	N	N	N
Forced Failover	Active OnScan	Standby	N	N	N	Y	Y	Standby	Active OnScan	Y	Y	Y	N	N
Server 1 Failure (hard shutdown)	Active OnScan	Down	N	N	N	N	N	Standby	Active OnScan	Y	Y	Y	N	N
Server 2 Failure (hard shutdown)	Active OnScan	Active OnScan	N	N	Y	N	N	Standby	Down	N	N	N	N	N
Graceful shutdown of Server 1	Active OnScan	Down	N	N	N	Y	Y	Standby	Active OffScan	Y	N	N	N	N
Graceful shutd	Active OnSc	Active OnSc	N	N	Y	N	N	Standby	Down	N	N	N	N	N

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

own of Server 2	an	an												
Start Server 1	Down (was OnScan)	Active OnScan	Y	Y	Y	N	N	Down	Down	N	N	N	N	N
Start Server 2 (Server 1 running)	Active OnScan	Active OnScan	N	N	Y	N	N	Down	Standby	N	N	N	N	N
Undeploy	Active OnScan	Down	N	N	N	Y	Y	Standby	Down	N	N	N	N	N

The following table summarizes the circumstances under which each script execution type runs when redundancy is set to **Run Warm Mode** (differences between the two modes are highlighted):

Run warm mode	**Primary engine (on server 1)**		**Executes on server 1**					**Backup engine (on server 2)**		**Executes on server 2**				
Action	**Initial state**	**End state**	**U**	**N**	**X**	**F**	**D**	**Initial state**	**End state**	**U**	**N**	**X**	**F**	**D**
Deploy	Down	Active OnScan	Y	Y	Y	N	N	Down	Standby	Y	N	N	N	N
Forced Failover	Active OnScan	Standby	N	N	N	Y	Y	Standby	Active OnScan	N	Y	Y	N	N
Server 1 Failure	Active OnScan	Down	N	N	N	N	N	Standby	Active OnScan	N	Y	Y	N	N

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Server 2 Failure (hard shutdown)	Active OnScan	Active OnScan	N	N	Y	N	N	Standby	Down	N	N	N	N	N
Graceful shutdown of Server 1	Active OnScan	Down	N	N	N	Y	Y	Standby	Active OffScan	N	N	N	N	N
Graceful shutdown of Server 2	Active OnScan	Active OnScan	N	N	Y	N	N	Standby	Down	N	N	N	N	Y
Start Server 1	Down (previously OnScan)	Active OnScan	Y	Y	Y	N	N	Down	Down	N	N	N	N	N
Start Server 2 (Server 1 running)	Active OnScan	Active OnScan	N	N	Y	N	N	Down	Standby	Y	N	N	N	N
Undeploy	Active OnScan	Down	N	N	N	Y	Y	Standby	Down	N	N	N	N	Y

Startup scripts

Startup scripts are called when an object containing the script is loaded into memory, such as during deployment, platform, or engine start.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Startup instantiates COM objects and .NET objects. Depending on load and other factors, assignments to object attributes from the Startup method may fail. Attributes that reside off-object are not available to the Startup method.

Startup scripts for redundant AppEngines

There are certain considerations that you must take into account when writing a Startup script that will run on redundant AppEngines. This section outlines whether or not a script will be executed under various scenarios, including deploy, forced failover, system failure, system startup, and undeploy operations.

Redundant engines can be set to run in either **Legacy Mode** or **Run Warm Mode**. Startup scripts for redundant engines may operate differently, depending on the selected redundancy mode.

Note: New redundant engines default to Run Warm Mode. Redundant engines in migrated galaxies default to Legacy Mode.

- **Legacy mode (RunWarm attribute is disabled):** In Legacy mode, the Standby engine does not start until failover occurs. This will result in longer failover times when compared with Run Warm Mode. Highlighted text indicates where there is a difference in script execution between Legacy mode and Warm Redundancy mode.

Legacy mode	**Primary engine (server 1)**		**Backup engine (server 2)**		**Startup script**
Action	**Initial state**	**End state**	**Initial state**	**End state**	**Script execution**
Deploy	Down	Active OnScan	Down	Standby	Startup scripts execute when the primary engine starts. Startup scripts do not execute on the backup engine as it starts and transitions to standby.
Forced Failover	Active OnScan	Standby	Standby	Active OnScan	Startup scripts execute when the backup engine starts.
Server 1 Failure (hard shutdown)	Active OnScan	Down	Standby	Active OnScan	Startup scripts execute when the backup engine transitions from standby to active.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Legacy mode	Primary engine (server 1)		Backup engine (server 2)		Startup script
Server 2 Failure (hard shutdown)	Active OnScan	Active OnScan	Standby	Down	Startup scripts do not execute in the event of a server 2 failure.
Graceful shutdown of Server 1 platform or engine using OCMC	Active OnScan	Down	Standby	Active Offscan	Startup scripts execute when the backup engine starts. The primary engine shuts down, Standby engine is started and goes to active but remains offscan.
Graceful shutdown of Server 2 platform or engine using OCMC	Active OnScan	Active OnScan	Standby	Down	Startup scripts do not execute. Shutdown of Server 2 has no affect on operations. Server 1 continues running onscan.
Start Server 1 only	Down	Active OnScan	Down	Down	Startup scripts execute when the Primary Engine on Server 1 starts.
Start Server 2 (Server 1 running)	Active OnScan	Active OnScan	Down	Standby	Startup scripts do not execute when the Backup Engine starts. Server 1 continues running onscan.
Undeploy	Active OnScan	Down	Standby	Down	Startup scripts do not execute during an undeploy operation.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

- **Run Warm mode (RunWarm attribute is enabled):** In Run Warm mode, Startup scripts do not execute during a failover in most circumstances, since both the Primary and Backup engines start concurrently. The Backup Engine on Server 2 will only execute Startup scripts if the Primary Engine on Server 1 is down. Highlighted text indicates where there is a difference in script execution between Legacy mode and Warm Redundancy mode.

Warm redundancy mode	**Primary engine (server 1)**		**Backup engine (server 2)**		**Startup script**
Action	**Initial state**	**End state**	**Initial state**	**End state**	**Script execution**
Deploy	Down	Active OnScan	Down	Standby	Startup scripts execute on both engines when the engines start (both engines start with warm redundancy).
Forced Failover	Active OnScan	Standby	Standby	Active OnScan	As the active engine transitions from active to standby, the active engine shuts down and restarts as the standby engine. This triggers the startup script to run again.
Server 1 Failure (hard shutdown)	Active OnScan	Down	Standby	Active OnScan	Startup scripts do not execute again because the backup engine has already started.
Server 2 Failure (hard shutdown)	Active OnScan	Active OnScan	Standby	Down	Startup scripts do not execute in the event of a server 2 failure. Server 1 continues to run.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Warm redundancy mode	Primary engine (server 1)		Backup engine (server 2)		Startup script
Graceful shutdown of Server 1 platform or engine using OCMC	Active OnScan	Down	Standby	Active OFFscan	Startup scripts do not execute again. The active engine shuts down, standby engine previously started and now goes to active but remains offscan.
Graceful shutdown of Server 2 platform or engine using OCMC	Active OnScan	Active OnScan	Standby	Down	Startup scripts do not execute. Shutdown of server 2 has no affect on operations. server 1 continues running onscan.
Start Server 1 only	Down	Active OnScan	Down	Down	Startup script executes when Primary Engine starts.
Start Server 2 (Server 1 running)	Active OnScan	Active OnScan	Down	Standby	Startup scripts execute when the backup engine starts on server 2 (runs as standby). Server 1 continues running onscan.
Undeploy	Active OnScan	Down	Standby	Down	Startup scripts do not execute during an undeploy operation.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

OnScan scripts

OnScan scripts are called the first time an AppEngine calls this object to execute after the object's scan state changes to OnScan. The OnScan method initiates local object attribute values and provides more flexibility in the creation of .NET or COM objects.

Attributes that are off-engine are not available to the OnScan method.

OnScan scripts for redundant AppEngines

This section outlines whether or not the script will run under various scenarios, including including deploy, forced failover, system failure, system startup, and undeploy operations.

The selected redundancy mode, Legacy or Run Warm, does not change the behavior of OnScan scripts for redundant engines. For both Legacy and Warm Redundancy modes:

- When failover occurs, OnScan scripts are triggered when the **Active** engine goes OnScan.
- OnScan scripts are NOT triggered when the **Standby** engine goes OnScan.

Both redundancy modes	**Primary engine (server 1)**		**Backup engine (server 2)**		**OnScan script**
Action	**Initial state**	**End state**	**Initial state**	**End state**	**Script execution**
Deploy	Down	Active OnScan	Down	Standby	Onscan scripts execute when the active engine transitions to onscan.
Forced Failover	Active OnScan	Standby	Standby	Active OnScan	Onscan scripts execute when the backup engine transitions to onscAn.
Server 1 Failure (hard shutdown)	Active OnScan	Down	Standby	Active OnScan	onscan scripts execute when the backup engine transitions to onscan.
Server 2 Failure (hard shutdown)	Active OnScan	Active OnScan	Standby	Down	Onscan scripts do not execute in the event of a

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Both redundancy modes	Primary engine (server 1)		Backup engine (server 2)		OnScan script
					server 2 failure (no state change for server 1).
Graceful shutdown of Server 1 platform or engine on Server 1 using OCMC	Active OnScan	Down	Standby	Active OffScan	Onscan scripts do not execute when the OCMC shuts down a platform or active engine on server 1. The standby engine on server 2 remains offscan.
Graceful shutdown of Server 2 platform or engine using OCMC	Active OnScan	Active OnScan	Standby	Down	Onscan scripts do not execute when the OCMC shuts down a platform or standby engine on backup server 2. The active engine on server 1 remains running onscan.
Start Server 1 only	Down	Active OnScan	Down	Down	Onscan scripts execute when the primary engine on server 1 starts and transitions to its prior state of active onscan.
Start Server 2 (Server 1 running)	Active OnScan	Active OnScan	Down	Standby	Onscan scripts do not execute when the backup engine starts (state of the active engine running on server 1 does not change).

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Both redundancy modes	Primary engine (server 1)		Backup engine (server 2)		OnScan script
Undeploy	Active OnScan	Down	Standby	Down	OnScan scripts do not execute during an undeploy operation.

Execute scripts

Execute scripts are called each time the AppEngine performs a scan and the object is OnScan.

The Execute script method is the workhorse of the scripting execution types. Use the Execute method for your run-time scripting to ensure that all attributes and values are available to the script.

If the **Quality** check-box is checked, the Execute method is similar to InTouch HMI scripts with the following conditional trigger types:

- Periodic: When going OnScan, a script with a periodic trigger executes immediately (at the next scheduled scan period of the AppEngine). It then executes periodically whenever the elapsed time evaluates as true.
- Data Change: Executes when a data value or quality changes between scans.

For the following trigger types, data changes between each scan are not evaluated, only the value at the beginning of each script is used for evaluation purposes. For example, if a Boolean attribute changes from True to False to True again during a scan cycle, this change is not evaluated as a data change as the value is True at the beginning of each scan cycle.

- OnTrue: Executes if the expression validates from a false on one scan to a true on the next scan.
- OnFalse: Executes if the expression validates from a true on one scan to a false on the next scan.

These scripts also have time-based considerations. A trigger period of 0 means that the script executes every scan.

Time-based scripts, WhileTrue, WhileFalse, and Periodic are evaluated and executed based on the elapsed time from a timestamp generated from the previous execution, not on an elapsed time counter. It is possible that a change in the system clock can change the interval between execution of these scripts.

- WhileTrue: Executes scan to scan as long as the expression validates as true at the beginning of the scan.
- WhileFalse: Executes scan to scan as long as the expression validates as false at the beginning of the scan.

For example, a periodic script is set to run every 60 minutes. The script executes at 11:13 AM. We expect it to execute 60 minutes later at 12:13 PM. However, a time synchronization event occurred and the node's time is adjusted from 11:33 AM to 11:30 AM.

The script still executes when the system time reaches 12:13 PM. But because of the time change, the actual (True) time period that elapsed between executions is 63 minutes.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Execute scripts for redundant AppEngines

This section outlines whether or not the script will run under various scenarios, including including deploy, forced failover, system failure, system startup, and undeploy operations.

The selected redundancy mode, Legacy or Run Warm, does not change the behavior of Execute scripts for redundant engines. For both Legacy and Warm Redundancy modes:

- The Active engine is triggered on execute.
- The Standby engine is NOT triggered on execute.

Both redundancy modes	**Primary engine (server 1)**		**Backup engine (server 2)**		**Execute script**
Action	**Initial state**	**End state**	**Initial state**	**End state**	**Script execution**
Deploy	Down	Active OnScan	Down	Standby	Execute scripts run after the active engine transitions to onscan, at the next scheduled scan period of the AppEngine.
Force Failover	Active OnScan	Standby	Standby	Active OnScan	Execute scripts run after the standby engine transitions to onscan, at the next scheduled scan period of the AppEngine.
Server 1 Failure (hard shutdown)	Active OnScan	Down	Standby	Active OnScan	Execute scripts run after the standby engine transitions to onscan, at the next scheduled scan period of the AppEngine.
Server 2 Failure (hard shutdown)	Active OnScan	Active OnScan	Standby	Down	Execute scripts run at the next scheduled scan period of the

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Both redundancy modes	Primary engine (server 1)		Backup engine (server 2)		Execute script
					AppEngine.
Graceful shutdown of Server 1 platform or engine using OCMC	Active OnScan	Down	Standby	Active OffScan	Execute scripts do not run when the OCMC shuts down a platform or active engine on server 1. The standby engine on server 2 remains offscan.
Graceful shutdown of Server 2 platform or engine using OCMC	Active OnScan	Active OnScan	Standby	Down	Execute scripts do not run when the OCMC shuts down a platform or standby engine on backup server 2. The active engine on server 1 remains running onscan.
Start Server 1 only	Down	Active OnScan	Down	Down	Execute scripts run after the active engine transitions to onscan, at the next scheduled scan period of the AppEngine.
Start Server 2 (Server 1 running)	Active OnScan	Active OnScan	Down	Standby	Execute scripts run at the next scheduled scan period of the AppEngine.
Undeploy	Active OnScan	Down	Standby	Down	Execute scripts do not run during an undeploy operation.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

OffScan scripts

OffScan scripts are called when the object is taken OffScan. This script type is primarily used to clean up the object and account for any needs to address as a result of the object no longer executing.

If an object is taken OffScan, either directly, or indirectly because its engine is taken OffScan, all in-progress asynchronous scripts for that object are requested to shut down by setting a Boolean shutdown attribute for the script to true. A well-written script checks this attribute before and after time-consuming operations. If the script takes more than 30 seconds to complete, a warning appears in the logger that the script is not responding to the shutdown command. However, the script is allowed to complete and is not terminated by force. This all takes place on the engine's main thread and could potentially hang the engine. During this time, the script might also time out and as a result exit before executing all its logic.

OffScan scripts for redundant AppEngines

This section outlines whether or not the script will run under various scenarios, including including deploy, forced failover, system failure, system startup, and undeploy operations.

The selected redundancy mode, Legacy or Run Warm, does not change the behavior of OffScan scripts for redundant engines. For both Legacy and Warm Redundancy modes:

- When failover occurs, OffScan scripts are triggered when the **Active** engine goes OffScan.
- OffScan scripts are NOT triggered when the **Standby** engine goes OffScan.

Both redundancy modes	**Primary engine (server 1)**		**Backup engine (server 2)**		**OffScan script**
Action	**Initial state**	**End state**	**Initial state**	**End state**	**Script execution**
Deploy	Down	Active OnScan	Down	Standby	Offscan scripts do not execute during a deploy operation.
Forced Failover	Active OnScan	Standby	Standby	Active OnScan	Offscan scripts execute when the backup engine transitions to offscan.
Server 1 Failure (hard shutdown)	Active OnScan	Down	Standby	Active OnScan	Offscan scripts do not execute when server 1 has a hard shutdown.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Both redundancy modes	Primary engine (server 1)		Backup engine (server 2)		OffScan script
Server 2 Failure (hard shutdown)	Active OnScan	Active OnScan	Standby	Down	Offscan scripts do not execute in the event of a server 2 failure (no state change for server 1).
Graceful shutdown of Server 1 platform or engine using OCMC	Active OnScan	Down	Standby	Active OffScan	Offscan scripts execute when the OCMC shuts down a platform or active engine on server 1. The standby engine on server 2 transitions from standby to active offscan.
Graceful shutdown of Server 2 platform or engine using OCMC	Active OnScan	Active OnScan	Standby	Down	Offscan scripts do not execute when the OCMC shuts down a platform or standby engine on backup server 2.
Start Server 1 only	Down	Active OnScan	Down	Down	Offscan scripts do not execute when the primary engine on server 1 starts and transitions to its prior state of active onscan.
Start Server 2 (Server 1 running)	Active OnScan	Active OnScan	Down	Standby	Offscan scripts do not execute when the backup engine starts (state of the active engine running

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Both redundancy modes	Primary engine (server 1)		Backup engine (server 2)		OffScan script
					on server 1 does not change).
Undeploy	Active OnScan	Down	Standby	Down	Offscan scripts execute during an undeploy operation when the active engine goes offscan before it shuts down.

Shutdown scripts

Shutdown scripts are called when the object is about to be removed from memory, usually as a result of the AppEngine stopping. Shutdown scripts are primarily used to destroy COM objects and .NET objects and to free memory.

Shutdown scripts for redundant AppEngines

There are certain considerations that you must take into account when writing a Shutdown script that will run on redundant AppEngines. This section outlines whether or not a script will be executed under various scenarios, including including deploy, forced failover, system failure, system startup, and undeploy operations.

Redundant engines can be set to run in either **Legacy Mode** or **Run Warm Mode**. Shutdown scripts for redundant engines may operate differently, depending on the selected redundancy mode.

Note: The warm redundancy feature was introduced in System Platform 2020 R2 SP1. New redundant engines default to Run Warm Mode. Galaxies with redundant engines that were created prior to System Platform 2020 R2 SP1 and migrated to the current System Platform version default to Legacy Mode.

Shutdown scripts for redundant engines may operate differently, depending on which redundancy mode is selected.

- **Legacy mode (RunWarm attribute is disabled):** In Legacy mode, the Standby Engine does not start until failover occurs. This will result in longer failover times when compared with Run Warm Mode. Highlighted text indicates where there is a difference in script execution between Legacy mode and Warm Redundancy mode.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Legacy mode	Primary engine (server 1)		Backup engine (server 2)		Shutdown script
Action	**Initial state**	**End state**	**Initial state**	**End state**	**Script execution**
Deploy	Down	Active OnScan	Down	Standby	Shutdown scripts do not execute during a deploy operation.
Forced Failover	Active OnScan	Standby	Standby	Active OnScan	Shutdown scripts execute when the primary engine shuts down.
Server 1 Failure (hard shutdown)	Active OnScan	Down	Standby	Active OnScan	Shutdown scripts do not execute in the event of a hard shutdown.
Server 2 Failure (hard shutdown)	Active OnScan	Active OnScan	Standby	Down	Shutdown scripts do not execute in the event of a hard shutdown.
Graceful shutdown of Server 1 platform or engine using OCMC	Active OnScan	Down	Standby	Active OFFscan	Shutdown scripts execute when the Primary Engine shuts down.
Graceful shutdown of Server 2 platform or engine using OCMC	Active OnScan	Active OnScan	Standby	Down	Shutdown scripts do not execute when server 2 is shut down. because the backup engine was never started.
Start Server 1 only	Down	Active OnScan	Down	Down	Shutdown scripts do not execute.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Legacy mode	Primary engine (server 1)		Backup engine (server 2)		Shutdown script
Start Server 2 (Server 1 running)	Active OnScan	Active OnScan	Down	Standby	Shutdown scripts do not execute.
Undeploy	Active OnScan	Down	Standby	Down	Shutdown scripts execute during an undeploy operation as the active engine is shut down. In legacy mode, the backup engine was never started.

- **Run warm mode (RunWarm attribute is enabled):** In Run Warm mode, the Standby Engine executes Startup scripts until it becomes active. Highlighted text indicates where there is a difference in script execution between Legacy mode and Warm Redundancy mode.

Warm redundancy mode	Primary engine (server 1)		Backup engine (server 2)		Shutdown script
Action	**Initial state**	**End state**	**Initial state**	**End state**	**Script execution**
Deploy	Down	Active OnScan	Down	Standby	Shutdown scripts do not execute during a deploy operation.
Forced Failover	Active OnScan	Standby	Standby	Active OnScan	Shutdown scripts execute when the primary engine shuts down.
Server 1 Failure (hard shutdown)	Active OnScan	Down	Standby	Active OnScan	Shutdown scripts do not execute in the event of a hard shutdown.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Warm redundancy mode	Primary engine (server 1)		Backup engine (server 2)		Shutdown script
Server 2 Failure (hard shutdown)	Active OnScan	Active OnScan	Standby	Down	Shutdown scripts do not execute in the event of a hard shutdown.
Graceful shutdown of Server 1 platform or engine using OCMC	Active OnScan	Down	Standby	Active Offscan	Shutdown scripts execute when the primary engine shuts down.
Graceful shutdown of Server 2 platform or engine using OCMC	Active OnScan	Active OnScan	Standby	Down	Shutdown scripts execute when server 2 is shut down because in run warm mode, the backup engine was started previously.
Start Server 1 only	Down	Active OnScan	Down	Down	Shutdown scripts do not execute.
Start Server 2 (Server 1 running)	Active OnScan	Active OnScan	Down	Standby	Shutdown scripts do not execute.
Undeploy	Active OnScan	Down	Standby	Down	Shutdown scripts execute during an undeploy operation as the servers shut down. In run warm mode, both engines were started and are now shut down.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Deployment scripts

Deploying objects is both a critical and a load-intensive process for a Galaxy. Implementing scripting in the Startup and OnScan methods can adversely affect a Galaxy's deployment and redundancy performance.

While objects are being deployed, their Startup and, if deployed OnScan scripts are executed. These scripts must complete within the deployment time-out period for the deployment to be successful.

Placing large numbers of scripts, or scripts that require heavy processing power into the Startup or OnScan script methods can slow or cause a deployment or failover to fail. In addition to the load that is placed on the system at deployment time, the type of scripting done in the Startup and OnScan methods is also important because these scripts execute in a sequence.

Deployment scripts for redundant AppEngines

When writing a deployment script that will run on redundant AppEngines, be sure to account for the mode in which the redundant engines will run. Redundant engines can be set to run in either **Legacy Mode** or Run **Warm Mode**.

- The selected redundancy mode, Legacy or Run Warm, does not change the behavior of OnScan scripts for redundant engines.
- The selected redundancy mode may change the behavior of Startup scripts. See Startup scripts for more information about the differences in script execution between modes.

  Note: The warm redundancy feature was introduced in System Platform 2020 R2 SP1. New redundant engines default to Run Warm Mode. Galaxies with redundant engines that were created prior to System Platform 2020 R2 SP1 and migrated to the current System Platform version default to Legacy Mode.

During deployment and restart, the Startup and OnScan script methods do not execute objects based on execution order. Objects are started up and placed on scan based on their alphanumeric tag name within their hosting Area.

Follow the recommendation below for each type of script method to help determine what scripting practices to follow in each script method.

Do not place the following types of scripting in the Startup or OnScan methods:

- Database access
- File system access to .csv, .xml, .txt, and other file types
- Off-object referencing
- Dynamic referencing

QuickScript editor features

The QuickScript editor provides a number of features to enhance scripting speed and accuracy.

Color indicators for script elements

The QuickScript .NET editor uses different text colors to identify different script elements. The following table

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

shows the text colors associated with script elements.

Element	Color
Keywords	Blue Syntax highlighted while typing.
Comments (both single line and multi-line)	Green Syntax highlighted while typing.
Strings	Purple Syntax highlighted while typing.
Function names, numeric constants, operators, semicolons, dim variables, alias variables, and so on	Black See descriptions for Attribute names and Reserved words.
Attributes, InTouch Tags, Reference Strings	Maroon, bold face
Reserved words	Red, non-bold face
.NET type names	Teal, non-bold face

Autocomplete

QuickScript autocomplete incorporates several features for use while authoring object and client scripts:

- Provides an autocomplete Attribute reference when you type a generic object name, such as "me." Run-time attributes appear in an autocomplete list box. Typing "InTouch:" displays an autocomplete list of tagnames from the most recently selected ViewApp template.
- Provides method parameter help in an autocomplete list box including context-specific suggestions covering definitions, keywords, script elements, and programmatic constructs such as try ... catch or while ... endwhile.
- Automatic word completion of Attribute references, methods, programmatic constructs, and other script elements.

These features serve as convenient documentation of method parameters and scripting syntax as well as an enhanced input method.

Autocomplete displays a context-sensitive list of options for script elements, keywords, object and attribute names, and programmatic constructs. Press Ctrl+space to display all available autocomplete options and variables for the selected location in the script. You can identify the context from the icons displayed with the list items.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Icon	Represents
	MxBoolean attribute
	MxInteger attribute
	MxFloat attribute
	MxDouble attribute
	MxString attribute
	MxTime attribute
	MxElapsedTime attribute
	MxReference attribute
	MxStatus attribute
	MxDataTypeEnum attribute
	MxSecurityClassification attribute
	MxDataQuality attribute
	MxQualifiedEnum attribute
	MxQualifiedStruct attribute
	MxInternationalizedString attribute
	.Net Method
	.Net Property

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Icon	Represents
	.Net Field or Variable
	.Net Namespace
	.Net Struct
	.Net Class
	.Net Interface
	.Net Enumeration
	.Net Enum Value
	QuickScript Keyword
	Contained object name, or any partial attribute name such as a attribute, field attribute, or primitive that has a dot in the name, or any attribute of Mx type MxNone, or if there are several type choices among objects and attributes. If the attribute cannot be exactly or unambiguously returned, this icon will appear. Partial name example: For me.alarm.a1, typing "me.alar" will show the blue ball icon for alarm. MxNone example: input/output extension attribute WriteValue.
	Rectangle
	Rounded rectangle
	Line
	Horizontal or vertical line

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Icon	Represents
	Text
	Ellipse
	Curve
	Closed curve
	Button
	Polygon
	Polyline
	Connect
	Image
	Group or embedded graphic
	Alarm control
	Edit box
	Arc
	Pie
	Chord
	Circle
	Status
	Radio buttons

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Icon	Represents
	Checkbox
	Edit box
	Combo box
	Calendar
	Date picker
	List box

Accept autocomplete suggestions

Insert an item at the editor caret from the autocomplete list box—without an end line or tab appended—by doing one of the following:

- Double-click the item.
- Highlight (select) the item and press the **Enter** key or the **Tab** key.

Type a space, period, comma, open or closed parenthesis, or other punctuation used in the QuickScript .NET programming language (: ; [ ] = < > - + / *), and the item highlighted in the autocomplete list box will be inserted at the editor caret with the additional character appended.

Multi-level undo and redo

You can selectively undo a history of changes to your script. The number of changes that can be undone is limited only by the amount of available memory.

An undone change can be redone. Redo mirrors undo changes.

A single undo typically is comprised of sequences of typing or deleting, which can be interrupted by interaction with an autocomplete list or by moving the cursor with the mouse, or by clicking elsewhere in the script.

All pending undo and redo actions will be lost if you close the object editor, switch to another script within the object editor, or switch among Startup, OnScan, Execute, OffScan, and Shutdown scripts.

Dynamic reference considerations

Dynamic reference scripting is one the biggest causes of deployment failures of StartUp and OnScan execution types.

Rather than placing dynamic referencing scripts in the Startup or OnScan methods, perform dynamic referencing

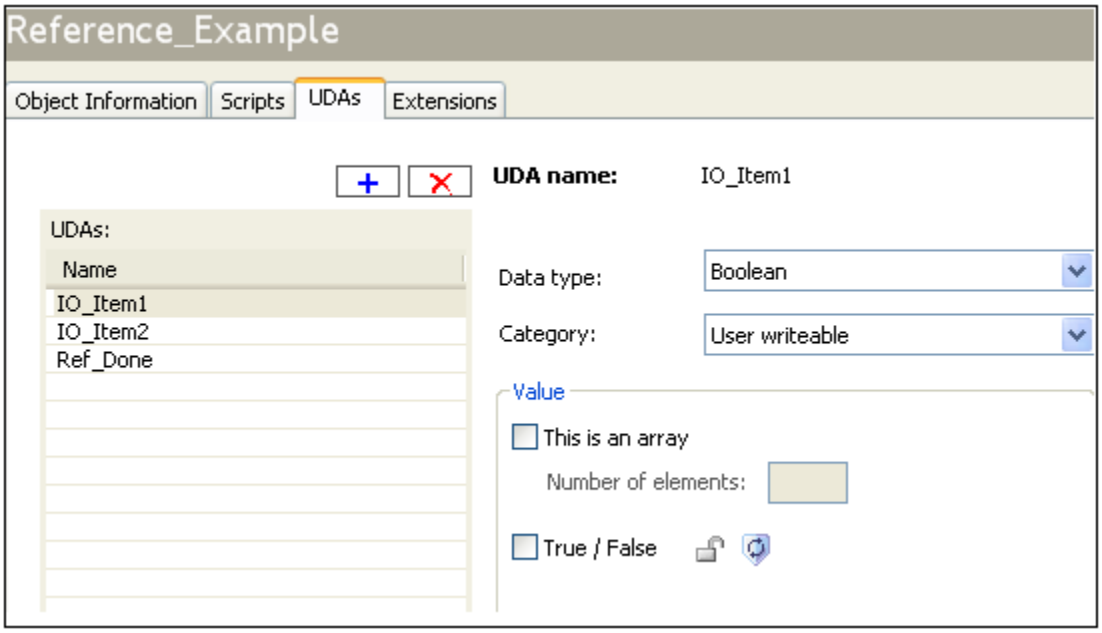
© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

in the Execute method. There are several advantages to using the Execute method with dynamic reference scripting:

- Deployment is faster.
- Deployment is more reliable.
- Deterministic execution order is guaranteed.
- Off-object and off-engine attributes are available.
- After a failover occurs, the startup of the redundant engine is more stable and can be faster.

To create a simple dynamic reference script example

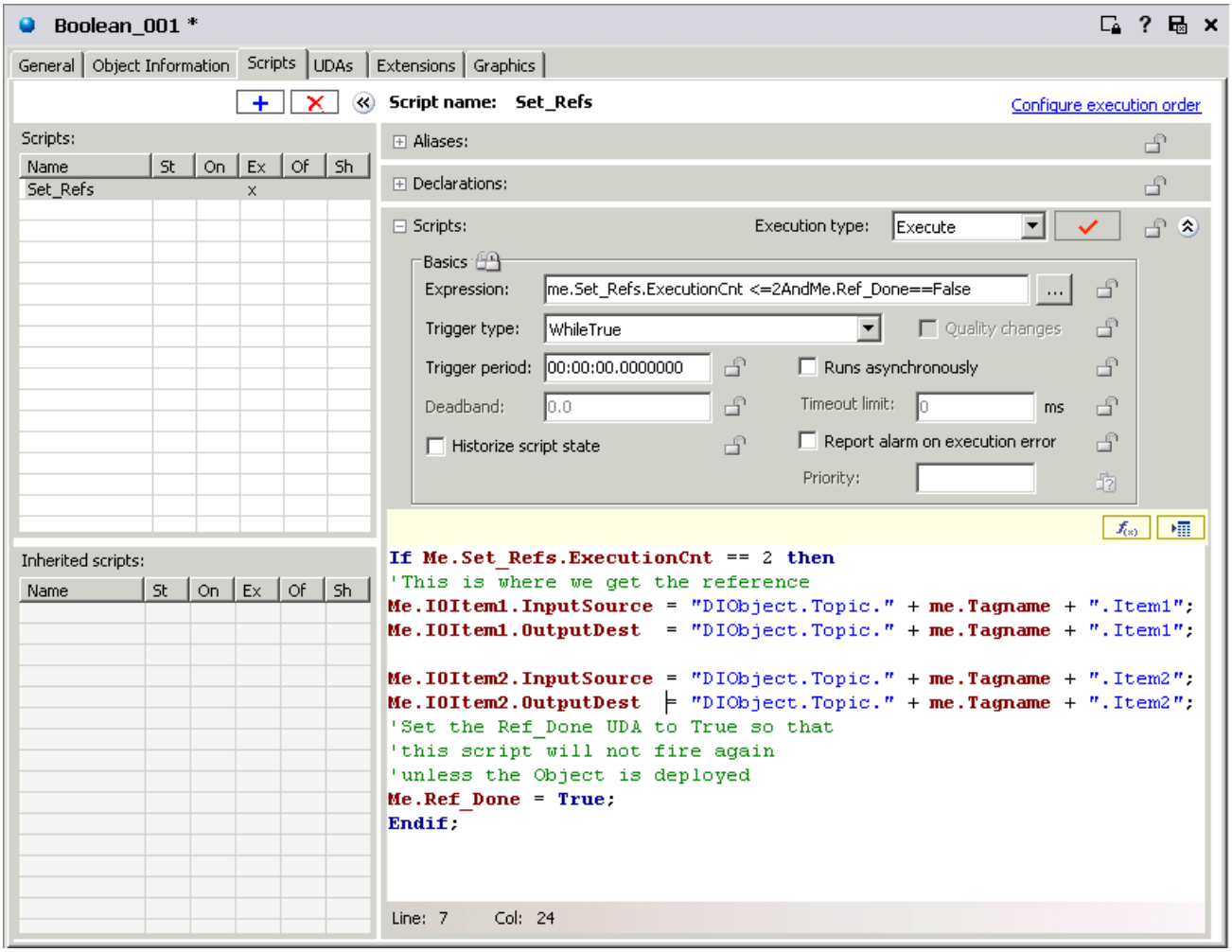
1. Create a Boolean attribute.



   The attribute shows if the referencing script is complete. In this example you create `Ref_Done`. `IO_Item1` and `IO_Item2` are the I/O points referenced in this example.

2. Create the script. The script in this example is called `Set_Refs`. The script has a trigger type of `WhileTrue` with a `0` trigger period.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.



The script is shown below:

```
If Me.Set_Refs.ExecutionCnt == 2 then
Me.IOItem1.InputSource = "DIObject.Topic." + me.Tagname + ".Item1";
Me.IOItem1.OutputDest = "DIObject.Topic." + me.Tagname + ".Item1";
Me.IOItem2.InputSource = "DIObject.Topic." + me.Tagname + ".Item2";
Me.IOItem2.OutputDest = "DIObject.Topic." + me.Tagname + ".Item2";
Me.Ref_Done = True;
Endif;
```

This script allows the system to stabilize after going on scan before setting the references. The script executes on the first two scans of the object when the Boolean attribute Ref_Done is false.

As the script is executed, a check is made against the execution count. If the count equals 2, the script performs the referencing operations. After the reference attributes are set on the attributes, the Ref_Done attribute is set to True. At this point the expression for the script is no longer true.

The three attributes set in this script are checkpointed, eliminating the need to run this script except on deployment. The next time the object is started, placed on scan, or failed over, there is no need to recreate the references to the items.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Run-time client script behavior

In Advanced Communication Management, script references to InTouch tags and object attributes are suspended from receiving data changes when the application window containing embedded ArchestrA® objects is minimized in InTouch WindowViewer. Suspending data updates to hidden objects reduces the amount of network traffic and improves the overall performance of a client application.

While Showing scripts of embedded graphics do not execute during the period when the window containing the graphics is minimized. Script execution resumes after restoring or maximizing a window that had been previously closed or minimized.

Open a client application window

In Advanced Communication Management, when a client application window containing embedded ArchestrA objects opens in WindowViewer, the following script events occur:

- Register all ArchestrA and InTouch references used in embedded graphic scripts, if not registered already.
- Advise all ArchestrA and InTouch references in embedded graphic scripts within the window, if not advised already.
- Execute the OnShow script on all embedded graphic scripts within the window.
- Execute named scripts if their trigger conditions are met.

Close a client application window

In Advanced Communication Management, when a client application window containing embedded ArchestrA objects is closed, the following script events occur:

- Execute OnHide scripts of all embedded graphics within the window.
- Stop running client scripts.
- Unadvise all ArchestrA and InTouch references in the Window if there are no other open windows using the references.
- Unregister all ArchestrA and InTouch references in the Window if there are no other open windows using the references.

Minimize a client application window

In Advanced Communication Management, when an open window containing embedded ArchestrA objects is minimized in WindowViewer, the following script events occur:

- Stop running client scripts associated with ArchestrA objects embedded in the window.
- Unadvise all ArchestrA and InTouch references in the Window if there are no other open windows using the references.
- OnHide scripts of embedded graphics do not execute when a window is minimized.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Maximize or restore a client application window

In Advanced Communication Management, after maximizing or restoring a window from WindowViewer that had been previously minimized or closed, the following script events occur:

- Advise all ArchestrA and InTouch script references in the window, if not advised already.
- Execute named scripts if their trigger conditions are met.

Visual indication of script errors

Verification errors in script text are marked with a red "squiggly" underline. The underline appears after approximately 2.5 seconds of keyboard inactivity.

Hovering over the error with the mouse cursor will display the error message as a tooltip. The tooltip error message is identical to the message shown when clicking the script verification button.

Note: In addition to error tooltips, the script editor will also display the variable name and type in a tooltip when hovering over a variable name in the script.

In some cases, more than one error will be underlined. This is not always possible because some errors prevent the compiler from continuing past the error.

Line numbers

The script editor displays line numbers in the left margin.

- Line numbers of up to four digits will display when the script editor is not zoomed.
- The line number may appear clipped for scripts longer than 9999 lines or when the script editor is zoomed.
- Use the right-click context menu **Go To** function to go to a specific line in the script.

Log functions

QuickScript .NET functions include several log functions to capture and display information in the logger under different log flags.

- LogCustom()
- LogError()
- LogMessage()
- LogTrace()
- LogWarning()

Important: To use the LogCustom function, you must enable Log Custom in the Operations Control Management Console (OCMC) Log Flag Editor. To use the LogTrace function, you must enable Log Trace in the OCMC Log Flag Editor.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

QuickScript .NET functions

For information about other functions in this category, see third-party documentation.

Keep in mind the following limitations when you use the script functions:

- Be aware of the .NET datatypes.
- Starting a GUI application from within a server script is not supported.
- Although QuickScript supports import libraries built with .NET CLR version 2.0.50727, it does not support any of the new language features introduced with .NET 2.0, such as generics.

Script functions

This section describes the script functions available in the HMI/SCADA development environment. The topics in this section are organized to reflect the organization of the functions in the Script Function Browser.

Other Microsoft .NET script functions, are not documented. Refer to Microsoft .NET documentation for descriptions of the functions.

aaMethods

aaMethod calls are executed in the OPC UA server through the OPC UA client, and are supported through the OPC client in the Gateway Communication Driver. See OPC UA Methods for details.

Graphic client functions

Use graphic client functions to hide and show graphics, open and close popup windows, log in and log off users, or to query custom properties contained in a graphic.

EmbedContent()

Dynamically embeds Industrial Graphics inside another Industrial Graphic at the top of the Z-order for use in an OMI ViewApp at run time. This function is available within any Industrial Graphics client script.

Note: In System Platform 2023 R2, EmbedContent() is supported only in the OMI desktop client. This function is not supported in the OMI web client.

Category

Graphic Client

Syntax

To embed an Industrial Graphic within another graphic:

```
Dim graphicInfo as aaGraphic.GraphicInfo;
Dim cpValues [2] as aaGraphic.CustomPropertyValuePair;
```

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

```
cpValues[1] = new aaGraphic.CustomPropertyValuePair("CP1", 20, true);
cpValues[2] = new aaGraphic.CustomPropertyValuePair("CP2", Pump.PV.TagName, true);
graphicInfo.Identity = "123";
graphicInfo.GraphicName = "Pump_001.Symbol_001";
graphicInfo. CustomProperties = cpValues;
EmbedContent(graphicInfo);
```

Parameter

GraphicInfo

aaGraphic.GraphicInfo properties

The EmbedContent() function supports a subset of aaGraphic.GraphicInfo properties. The supported properties are as follows:

Identity	*The **Identity** property is a unique name to identify embedded content. It is unique across the entire OMI application at runtime.* *The **Identity** name cannot contain more than 329 characters and must contain at least one letter. Valid characters are alphanumeric and special characters ($, #, _).* *This is a **required** property of data type **string**.* • *If you do not specify the **Identity** name, a warning message is logged in the Logger, and the **EmbedContent** call fails.* • *If the **EmbedContent** call is first made with one **Identity** and then a a second call is made with a different **Identity** but the same **GraphicName**, the same symbol is embedded with a new identity.* • *If **EmbedContent** is called again with the same **Identity**, it replaces the existing embedded content that used the **Identity** in the previous call.* *The same **Identity** is used for the RemoveContent() script function.* *Example:* `graphicInfo.Identity = "i1";`
GraphicName	***GraphicName** is the name of the graphic to show.* ***GraphicName** cannot contain more than 329 characters and must contain at least one letter. Valid characters are alphanumeric and special characters ($, #, _).* *This is a **required** property of data type **string**.*

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

	Default value: ***none***
	 Browse using the ***Galaxy Browser****, or directly type the graphic name. The* ***GraphicName*** *can come from:*
	 • *Visualization folder, for example:* `"Symbol_001"`
	 • *Instances, absolute or hierarchical, for example:* `"Userdefined_001.Symbol1"`, `"Userdefined_001.Pump_001.S1"`
	 • *Relative reference, for example:* `"Me.Symbol_001"`
	 Use an absolute name to specify the graphic name and owning object if you are using an Object Wizard with Symbol Wizard custom property selections. This allows the correct graphic configuration to be shown for the instance. See ***OwningObject****, below, for more information.*
	 If you type any invalid character or exceed the character limit, the system shows a warning message at run time. There is no validation at design time.
	 The graphic name can be a concatenation of constant strings and reference strings. For example:
	 `"Pump_001" + ".Graphic_001";`
	 `cp1 + ".Graphic_001"`
	 `where the value of cp1 = "Pump_001"`
	 – or
	 `Obj1.Str1 + ".Symbol_001"`
	 where the value of Obj.Str1 = "Pump_001".
	 Examples:
	 Visualization Folder reference
	 `graphicInfo.GraphicName = "S1";`
	 Absolute Reference
	 `graphicInfo.GraphicName = "OwningObjectName.SymbolName";`

OwningObject	*The name of the owning object of the graphic shown by the* ***EmbedContent()*** *script function.* *It is the owning object of the graphic shown by the* ***EmbedContent*** *script function. It can be a concatenation of constant strings and reference strings.* ***OwningObject*** *can be browsed by the* ***Galaxy Browser*** *or or you can type the name of the object.*

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

	The browser mode can be Tag name or Hierarchical Name. If selected from the Browser, double quotes are added to the selected automation object name. *This is is an **optional** property of data type string.* *Default value: **empty*** *Example:* `graphicInfo.OwningObject = "UserDefined_001";`
X	***X** specifies the horizontal position of the embedded content. This value is relative to the symbol in which the **EmbeddedContent()** function is called.* *This is an **optional** property of data type **integer**.* *Default Value: **0*** *Valid Range: -2,147,483,648 through 2,147,483,647* *If **X** is beyond the integer range, an overflow message appears in the Logger at run time.* *Example:* `graphicInfo.X = 100;`
Y	***Y** specifies the vertical position of the embedded content. This value is relative to the symbol in which the **EmbeddedContent()** function is called.* *This is an **optional** property of data type **integer**.* *Default Value: **0*** *Valid Range: -2,147,483,648 through 2,147,483,647* *If **Y** is beyond integer range, a proper overflow message will appear in the Logger at run time.* *Example:* `graphicInfo.Y = 100;`
Width	***Width** specifies the width of the embedded symbol. Default value is the actual value coming from symbol design-time. The valid range is from 0 to 10000. If **Width** is less than 0 or greater than 10000, an "Out of range" warning is logged at runtime. No validation is done at design time.* *This is an **optional** property of data type **integer**.* *Default Value: **100*** *Valid Range: 0–10000* *Both **Width** and **Height** must be specified in order to take effect. If only the height or width parameter is provided, the provided value is ignored and instead, the values are taken from the symbol as designed.*

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Example:

```
graphicInfo.width = 500;
```

Height

***Height** specifies the height of the embedded graphic. Default value is the actual value coming from graphic design-time. The valid range is from 0 to 10000. If **Height** is less than 0 or greater than 10000, an "Out of range" warning is logged at runtime. No validation is done at design time.*

*This is an **optional** property of data type **integer**.*

Default Value: 100

Valid Range: 0–10000

*Both **Width** and **Height** must be specified in order to take effect. If only the height or width parameter is provided, the provided value is ignored and instead, the values are taken from the symbol as designed.*

Example:

```
graphicInfo.height = 350;
```

CustomProperties

***CustomProperties** is a ValuePair[] array that is used to override the custom properties of the graphic being shown.*

*This is an **optional** property of data type **ValuePair[]**.*

The first three parameters are:

- *Custom property name*
- *value*
- *IsConstant*

  The collection of custom property name, value, and IsConstant sets custom properties in the symbol being shown. Both the custom property and the value could be constant string, reference or concatenation of strings.

  The first three parameters are custom property name, value, and IsConstant.

  Both custom property and the value can be a constant string, reference, or concatenation of strings.

  *If the parameter **IsConstant** = True, the value is treated as a constant. Otherwise, the value is treated as a reference.*

  The array index starts at 1.

  Example:

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

```
Dim cpValues [4] as
aaGraphic.CustomPropertyValuePair;
cpValues[1] = new
aaGraphic.CustomPropertyValuePair("CP
1", 20, true);
cpValues[2] = new
aaGraphic.CustomPropertyValuePair("CP
2", Pump.PV.TagName, true);
cpValues[3] = new
aaGraphic.CustomPropertyValuePair("CP
3", "CP"+var1, CP2 + "001" +
".Speed", true);
cpValues[4] = new
aaGraphic.CustomPropertyValuePair("CP
3", "<HMIName>:Tag1", false);
graphicInfo.CustomProperties =
cpValues;
```

Summary of supported GraphicInfo properties

Property Name	Required (y/n)	Data Type	Default Value
Identity	Yes	String	None
GraphicName	Yes	String	None
OwningObject	No	String	Empty
X	No	Integer	0
Y	No	Integer	0
Width	No	Integer	100
Height	No	Integer	100
CustomProperties	No	ValuePair[] array	Empty

Remarks

- Any parameter that has a default value in the GraphicInfo is optional. If no input value specified for these parameters, the default values are used at run time. Any parameter except the Enum data type can be a constant, reference, or expression.
- String properties can be a concatenation of strings and/or custom properties. Detailed information about these properties is listed below.
- For more information, see "Working with the Show/Hide Graphics Script Functions" in the *Industrial Graphic Editor User Guide*.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Example for EmbedContent

```
Dim graphicInfo as aaGraphic.GraphicInfo;
Dim cpValues[4] as aaGraphic.CustomPropertyValuePair;
cpValues[1] = new aaGraphic.CustomPropertyValuePair("CP1", 20, true);
cpValues[2] = new aaGraphic.CustomPropertyValuePair("CP2", "Pump.PV.TagName", true);
cpValues[3] = new aaGraphic.CustomPropertyValuePair("CP"+var1, CP2+"001" + ".Speed", true
cpValues[4] = new aaGraphic.CustomPropertyVlauePair(CP3", InTouch:Tag1",false
graphicInfo.Identity = "123";
graphicInfo.GraphicName = "Pump_001.Symbol_001";
graphicInfo.CustomProperties = cpValues;
graphicInfo.OwningObject = "UserDefined_001";
EmbedContent( graphicInfo );
```

Where "123" is string Identity and the graphic "Pump_001.Symbol_001" contains custom properties CP, CP1, CP2, and CP3.

See Also

EmbedMultipleContent()

ShowGraphic()

HideContent()

GetCPQuality()

Returns the Quality value of a custom property. This function is available within any Industrial Graphics client script, but may not be supported by your HMI. For more information, consult your HMI documentation.

Syntax

```
Int GetCPQuality(String name)
```

Where *String name* is the name of the custom property whose quality is to be retrieved.

This script function takes the name of a custom property on the graphic. This argument is of type string and it can be a reference or a constant.

If the custom property is type constant, GOOD is the quality always returned.

For use with custom properties only. It does not apply to HMI tags.

Return value

The GetCPQuality() script function returns a value 0-255 of type Integer, as per the OPC quality standard. 192 is GOOD.

Example

```
cp2 = GetCPQuality("cp1");
```

Where cp1 and cp2 are custom properties and the data type of cp2 is Integer.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

GetCPTimeStamp()

Returns the time stamp of a custom property. This function is available within any Industrial Graphics client script.

Syntax

```
DateTime GetCPTimeStamp(String name)
```

Where *String name* is the name of the custom property whose time stamp is to be retrieved.

This script function takes the name of a custom property on the graphic. This argument is of type string and it can be a reference or a constant.

For use with custom properties only. It does not apply to HMI tags.

Return value

The GetCPTimeStamp() script function returns the time stamp of the custom property's current value of type DateTime. If the custom property value is a constant, then the return value is the time the value was created.

Example

```
cp2 = GetCPTimeStamp("cp1");
```

Where cp1 and cp2 are custom properties and the data type of cp2 is DateTime.

GetReferences()

Returns an XML string of all configured references in the runtime. The string can be saved to an XML file by using a .NET object.

Category

Miscellaneous

Syntax

```
stringXml = GetReferences("GraphicName");
```

Parameter

GraphicName

Description

Name of the graphic for which references are required.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Default

Empty

Parameter options

The following arguments can be specified for the function:

Empty string ("")	*When GetReferences() is called with an empty string argument, GetReferences() will export the references in the graphic where GetReferences() is called.* *For example, the graphic "s_itag1" is configured with a button that has an Action Script configured with GetReferences(""). When the button is clicked at runtime, it exports the references configured in the "s_itag1" graphic.*
Graphic name used in the InTouch Window	*You can use an action script to export the references within the same graphic. In addition, the scripts can be run from another embedded graphic by specifying the graphic name in the InTouch window. The graphic name and reference VTQ in the graphic will be exported.*
Graphic element with hierarchy	*An example of a graphic element that includes the hierarchy is as follows:* *Symbol_c contains a text element – Text1.* *The method call would be:* `sXML = GetReferences ("Symbol_c1.Text1");`

Return value

An XML string with the following exported references is returned:

- Public Custom Properties
- Private Custom Properties with overrides
- InTouch tags
- Attribute() with InTouch tags
- VTQS, if the WinPlatform is deployed
- MxReferences with absolute reference
- MxReferences with resolved relative references (Me.Value)
- Graphic references used in the ShowGraphic() scripts function
- Popup graphic launched by ShowSymbol animation
- References configured in the SetCustomPropertyValue() scripts function.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Examples

Example 1

```
dim sXml as string;
sXml = GetReferences("");
dim xmlDoc as System.Xml.XmlDocument;
xmlDoc = new System.Xml.XmlDocument();
xmlDoc.LoadXml(sXml);
xmlDoc.Save("C:\\tmp\\StringXML.xml");
```

Example 2

```
dim sXml as string;
sXml = GetReferences("s_itag1");
dim xmlDoc as System.Xml.XmlDocument;
xmlDoc = new System.Xml.XmlDocument();
xmlDoc.LoadXml(sXml);
xmlDoc.Save("C:\\tmp\\s_itag1.xml");
```

Example 3

```
dim sXml as string;
sXml = GetReferences("Tank1.Ellipse1");
dim xmlDoc as System.Xml.XmlDocument;
xmlDoc = new System.Xml.XmlDocument();
xmlDoc.LoadXml(sXml);
xmlDoc.Save("C:\\tmp\\Tank1Ellipse1.xml");
```

SetCustomPropertyValue()

Change the expression or reference of a custom property at run time.

Category

Graphic Client

Syntax

```
SetCustomPropertyValue(System.String name, System.String value, System.Boolean isConstant);
```

Parameters

Name	*The name of the custom property to be modified on the graphic. This is a string property and can be a reference or a constant.*
Value	*The new value to be set. **Value** is a string property and can be an expression, reference, or constant. If the value is given in quotes ("), then the value is considered a constant. If the value is given without quotes, then the value of the expression is considered*

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

	a reference.
isConstant	***IsConstant** is a **Boolean** flag that indicates whether the new value will be evaluated as a constant or a reference.* • *If set to True (1), then the new value will be treated as a constant.* • *If it is set to False (0), then the new value will be treated as a reference.* *This parameter only applies when the value parameter is a reference or constant and the custom property specified in the name parameter is a string or time type. This parameter has no meaning if the custom property is an integer, float, Boolean, or double type.* ***isConstant** does not override the type of input for the **value** parameter. The **value** parameter itself can be either a constant or a reference, depending on whether it is enclosed in quotes. The **isConstant** parameter is only determines how the actual value (coming from the V**alue** parameter) is evaluated.*

Additional Information

The whole expression or reference of the custom property is replaced with the new value, regardless if it is overridden or not. No partial replacement is supported.

Only public custom properties on the graphic can be changed.

When the method executes, it overrides any modifications done by previous IOSetRemoteReference() calls from a native InTouch script.

Examples

Configuring the custom property as a reference:

Motor_001 is an object with the string attribute name "State" that stores the current state of the motor ("Running" or "Stopped"). You also have an Industrial graphic that has the string data type custom property "MotorState." The following script code will set the MotorState custom property to Motor_001.State in run-time:

```
   GraphicA.SetCustomPropertyValue("MotorState","Motor_001.State",False);
```

As a result of the call, the function will set the string custom property GraphicA.MotorState to "Motor_001.State" as a reference. The string custom property GraphicA.MotorStatus will resolve that reference and update its value with the reference value ("Running" or "Stopped").

Configuring the custom property as a constant:

Motor_001 is an object with the Boolean attribute name "State" that reflects the current state of the motor

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

(True or False). You also have an Industrial graphic that has the string data type custom property "MotorState." The following script code causes the MotorState custom property to hold the state of equipment—"Running" or "Stopped"—as text based on the value returned for Motor_001:

```
IF Motor_001.State THEN
GraphicA.SetCustomPropertyValue("MotorState","Running",True);
ELSE
GraphicA.SetCustomPropertyValue("MotorState","Stopped",True);
ENDIF;
```

As a result of the call, the function will set the string custom property GraphicA.MotorState to "Running" or "Stopped," depending on the value of Motor_001.State.

HideContent()

Closes one or more matching content items within an AVEVA OMI ViewApp. Multiple content items can be closed if they match the parameters that are specified in the HideContent call. The HideContent() function uses a subset of the parameters that ShowContent() uses.

The HideContent() function works only within a single level of the layout, and the level is defined by the SearchScope parameter. By default, SearchScope is "Self," and searches within the layout that has invoked it. This function is available within any Industrial Graphics client script or AVEVA OMI layout script. SearchScope parameters other than "Self" constrain the search for content to only layouts that are directly associated with the Screen Profile, and not a nested layout. (A nested layout is a layout embedded or contained in a pane of another layout.)

Note: While the HideContent() function is available in object scripts through both IntelliSense and the IDE function browser, its use in object scripts is not supported.

Category

Graphic Client

Syntax

```
Dim contentInfo as aaContent.ContentInfo;
contentInfo.Content = “SA_Valve_2Way";
contentInfo.Name =”SA_Valve_2Way1”;
contentInfo.ContentType = “Level_3”;
contentInfo.PaneName =”Pane 2”;
contentInfo.ScreenName =”Primary”;
contentInfo.SearchScope = aaContent.SearchScope.Self;
HideContent( contentInfo );
```

Parameter

ContentInfo

The description of the content, along with the location of the content (screen and pane) to be hidden.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Data type

aaContent.ContentInfo

aaContentInfo properties

You must pass the ContentInfo parameter in the HideContent call (i.e., `HideContent ( contentInfo )`), even if you do not define any ContentInfo properties. No properties are required; the ContentInfo properties that can be specified optionally are:

Content	***Content** is a unique name for an item, either in the Visualization folder or associated with an asset, that specifies the content to be loaded into the pane. It can be a graphic, a layout, or external content.* *This is an optional property of data type **string**.* ***Content** is the name of the item within the Visualization folder or associated with an asset. The content names are the names shown in the Visualization folder. The Properties tab of the Layout and ViewApp editors lists content name as the Content property.* *Relative names, for example, "Me.S1," can also be used to designate content.* *The **Content** name must be unique. Application Server does not check for duplicated names. If **Content** is duplicated, all content with the same name is closed.* *If the same content item is used in multiple panes of the layout, and the "Content" property is specified by the HideContent() method, all instances of the content item are hidden. To hide a single instance of a content item that appears more than once in the layout, use the "Name" property instead.* *Example:* `contentInfo.Content = "UserDefinedObject_001.Symbol_001";`
Name	***Name** is the auto-generated (or user-edited) name of a unique content item. **Name** is created when the content item is added to a layout pane.* *This is an **optional** property of data type **string**.* ***Name** must be unique within a layout. **Name** can be duplicated in nested layouts, as long as the name is not duplicated within a nested single layout or the top level layout. The aaContentInfo **Name** property is shown in the **Properties** tab of the Layout Editor, and is auto-generated from the **Content** property, also*

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

	*shown in the Properties tab. You can edit the **Name** property.* *When **Name** is specified in a HideContent() call, only the uniquely-named content is closed. If **Content** is specified and the **Name** property is not specified, all items in the layout with the same content name are hidden. If both are specified, **Name** has precedence.* *Example:* `contentInfo.Name = "Symbol_011";`
ScreenName	***ScreenName** specifies the screen that contains the pane with the content to be closed.* *This is an **optional** property of data type **string**.* *ScreenNames are configured in the Screen Profile Editor. See Screen Profiles in the AVEVA OMI Help for additional information.* *Example:* `contentInfo.ScreenName = "Wall";`
PaneName	***PaneName** specifies the pane containing the content to be closed.* *This is an **optional** property of data type **string**.* *PaneNames are configured in the **Layout Editor**. See Layouts in the AVEVA OMI Help for additional information.* *Example:* `contentInfo.PaneName = "Pane1";`
ContentType	***ContentType** specifies the content type of the content to be closed, for example, "Overview," "Navigation," or "Faceplate."* *This is an **optional** property of data type **string**.* ***ContentType** is matched against the Content Type property that can be set for a pane in the Layout Editor. **ContentType** is used to override the actual type of the specified Content. If **ContentType** is not specified, Content is examined for its type of content. See Layouts in the AVEVA OM Help for additional information about content types.* *Example:* `contentInfo.ContentType = "Overview";`
SearchScope	***SearchScope** can be used when when **ScreenName** has not been specified. **SearchScope** sets which screen or screens will be searched for a pane that matches the*

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

	*specified **PaneName** or **ContentType**.* *This is **optional** property of data type **enumuration**.* *The default **SearchScope** is "Self."* *The following values are valid with **SearchScope:*** • ***Self** searches for matching content within the panes of the layout from which the **HideContent** call was made. If **SearchScope** is not specified, "Self" is the default. When **SearchScope** = Self, the layout that initiated the call is searched, whether it is the top level layout or a nested (embedded) layout.* *In contrast to **Self**, the remaining **SearchScope** values reference only the top level layout, not nested layouts.* • ***AllScreens** searches for matching content within the panes of all screens in the top level layout. The search starts with the source screen, then the primary screen, and then any remaining screens in alphabetical order.* • ***SourceScreen** searches for matching content only within the panes of the top level layout from which **HideContent** was called.* • ***PrimaryScreen** searches for matching content only within the panes of the top level layout of the screen designated in the Screen Profile as the primary screen.* *In a nested layout, **HideContent** searches for a matching pane within the layout that initiated the **HideContent** call, if **SearchScope** is **Self** or is not specified* *When **SearchScope** is **All**, **Source**, or **Primary**, or if the **ScreenName** is specified, **HideContent** searches for matching content within the top-level layout only* *Example:* `contentInfo.SearchScope = aaContent.SearchScope.PrimaryScreen;`
OwningObject	***OwningObject** sets the owning object of the content shown by the **ShowContent()** script function.* *This is an **optional** property of data type **string**.* ***OwningObject** can be used for relative referencing. It can be a concatenation of constant strings and reference strings.*

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

*You can browse for the **OwningObject** using the Display Automation Object Browser, or you can type the name of the owning object.*

*The **OwningObject** property sets references for the graphic, but is not associated with the **GraphicName** property, if the graphic is part of an Object Wizard. Therefore, if you are scripting a graphic with an owning object, specify the owning object name as part of the **GraphicName** property, as shown in the following example:*

Example:

```
UserDefined_001.Pump_001.
```

Summary of supported ContentInfo properties

Property Name	Required (y/n)	Data Type	Default Value
Content	No	String	None
Name	No	String	None
ScreenName	No	String	None
PaneName	No	String	None
ContentType	No	String	None
SearchScope	No	Enumeration	Self
OwningObject	No	String	None

Examples

```
Dim contentInfo as aaContent.ContentInfo;
contentInfo.Name = “Symbol21”;
HideContent( contentInfo );
```

Where "Symbol21" is the Name property of the content shown in the Layout Editor.

```
Dim contentInfo as aaContent.ContentInfo;
contentInfo.Content = “Symbol_001”;
HideContent( contentInfo );
```

Where "Symbol_001" is the name of the content as listed in the Visualization folder.

```
Dim contentInfo as aaContent.ContentInfo;
HideContent( contentInfo );
```

When ContentInfo does not define any properties in the HideContent call, the nested

layout that called it is hidden. In this case, it works identically to the HideSelf method. See HideSelf() for more information.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

If HideContent() with no ContentInfo properties is called from the top level of a layout, it has no effect; that is, the top level layout is not closed.

Terms

Content type: specifies the type of content represented by a pane.

Content: the name of a graphic, layout, or external content item as it is listed within the the Visualization folder. This is displayed as the Content property in the Layout Editor when the content item is added to a layout.

Name: the unique name assigned to an instance of a content item, when it is added to a layout. This is displayed as the Name property in the Layout Editor when the content item is added to a layout, and can be edited.

Layout: consists of one or more rectangular areas called panes that contain content shown in a ViewApp. A layout is associated with a screen, or it can be embedded within a pane of another layout.

Embedded or Nested Layout: In the context of ShowContent and HideContent, an embedded layout is a layout that is placed inside a pane of a containing layout. When SearchScope is "Self" (default), embedded layouts are searched for content that matches the parameters specified in the ShowContent/HideContent call.

Pane: rectangular area of a layout that can hold a single piece of content.

Primary screen: represents the main screen of a workstation that will show a running ViewApp

Screen Profile: defines the physical characteristics of one or more client workstation screens that will show a running ViewApp and how these screens are arranged with respect to each other.

Source screen: screen from which ShowContent or HideContent was called.

See also

ShowContent()

HideSelf()

HideGraphic()

Closes an open graphic pop-up window shown in the ShowGraphic() script with the given identity name.

The HideGraphic() function has been extended to close HMI Windows identified with a given identity name. This function is available within any Industrial Graphics client script.

Category

Graphic Client

Syntax

```
HideGraphic(string identity);
```

Parameter

Identity

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Description

The unique name of the instance that shows the graphic.

Examples

```
HideGraphic("i1");
```

Where "i1" is string Identity.

```
HideGraphic("<HMIName>:Window1");
```

Where "<HMIName>1" is the string identity.

See also

ShowGraphic(), HideSelf()

HideSelf()

Closes the displayed graphic or layout for which this script is configured. This script function is available within any Industrial Graphics client script.

Category

Graphic Client

Syntax

```
HideSelf();
```

Remarks

For an Industrial Graphics script, call the script function within the graphic to hide the popup.

Example

```
HideSelf();
```

See also

ShowGraphic(), HideGraphic()

Logoff()

Action script that automatically logs off the current user from an AVEVA OMI ViewApp. Logoff() is for use with AVEVA OMI only.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Action scripts are graphic animations that are triggered by a user action such as a mouse click.

Category

Miscellaneous

Syntax

```
LogOff() ;
```

Parameter

None

Trigger

On Left-Click/Key/Touch Down

Additional information

A log off button can be added that uses the Logoff() method to allow the user to log off from the HMI/SCADA application .

Example

```
Logoff() ;
```

See also

ShowLoginDialog()

RemoveEmbeddedContent()

Closes the matching multiple content items within a graphic in a AVEVA OMI ViewApp, if placed in the graphic using the EmbedMultipleContents() function. Multiple content items can be closed if they match the parameters that are specified in the RemoveEmbeddedContent() call. The RemoveEmbeddedContent() function uses a subset of the parameters that EmbedMultipleContents() uses.

Note: While the RemoveEmbeddedContent() function is available in object scripts through both IntelliSense and the IDE function browser, its use in object scripts is not supported.

Category

Graphic Client

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Syntax

```
Dim graphicCollection as aaGraphic.GraphicCollection;
graphicCollection.Identity = “<Identity>";
graphicCollection.GraphicName = “<SymbolName>";
```

Parameter

GraphicCollection

aaGraphic.GraphicCollection properties

Identity	*The **Identity** property is a unique name to identify embedded content. It is unique across the entire OMI application at runtime. Multiple Identity properties can be defined within the **GraphicCollection** parameter.*
	*Each **Identity** name cannot contain more than 329 characters and must contain at least one letter. Valid characters are alphanumeric and special characters ($, #, _).*
	*This is a **required** property of data type **string**.*
	Example:
	`graphicInfo1.Identity = "i1";`
	`graphicInfo2.Identity = "i2";`

GraphicName	***GraphicName** is the name of the graphic to remove.* ***GraphicName** cannot contain more than 329 characters and must contain at least one letter. Valid characters are alphanumeric and special characters ($, #, _).* *This is a **required** property of data type **string**.* *Default value: **none*** *Browse using the **Galaxy Browser**, or directly type the graphic name. The **GraphicName** can come from:* • *Visualization folder, for example: `"Symbol_001"`* • *Instances, absolute or hierarchical, for example: `"Userdefined_001.Symbol1"`, `"Userdefined_001.Pump_001.S1"`* • *Relative reference, for example: `"Me.Symbol_001"`* *Use an absolute name to specify the graphic name*

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

	*and owning object if you are using an Object Wizard with Symbol Wizard custom property selections. This allows the correct graphic configuration to be removed. See **OwningObject**, below, for more information.*
	If you type any invalid character or exceed the character limit, the system shows a warning message at run time. There is no validation at design time.
	The graphic name can be a concatenation of constant strings and reference strings. For example:
	`"Pump_001" + ".Graphic_001";`
	`cp1 + ".Graphic_001"`
	`where the value of cp1 = "Pump_001"`
	– or
	`Obj1.Str1 + ".Symbol_001"`
	where the value of Obj.Str1 = "Pump_001".
	Examples:
	Visualization Folder reference
	`graphicInfo1.GraphicName = "S1";`
	Absolute Reference
	`graphicInfo1.GraphicName = "OwningObjectName.SymbolName001";`
	`graphicInfo2.GraphicName = "OwningObjectName.SymbolName002";`

OwningObject	*The name of the owning object of the graphic to be removed by the **RemoveMultipleContent()** script function.* *It is the owning object of the graphic shown by the **EmbedMultipleContents** script function. It can be a concatenation of constant strings and reference strings. **OwningObject** can be browsed by the **Galaxy Browser** or or you can type the name of the object. The browser mode can be Tag name or Hierarchical Name. If selected from the Browser, double quotes are added to the selected automation object name.* *This is is an **optional** property of data type string.* *Default value: **empty*** *Example:* `graphicCollection.OwningObject = "UserDefined_001";`

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Remarks

- String properties can be a concatenation of strings and/or custom properties. Detailed information about these properties is listed below.
- For more information, see "Working with the Show/Hide Graphics Script Functions" in the *Industrial Graphic Editor User Guide*.

See also

EmbedMultipleContents()

HideContent()

ShowContent()

Loads a content item into an AVEVA OMI pane. This function is available within an Industrial Graphics client script or layout script to show the content of pane.

To load a graphic into a modal or modeless popup window, use ShowGraphic(). You can use ShowGraphic() for both AVEVA OMI and InTouch HMI ViewApps. The ShowContent() method is for AVEVA OMI only.

Note: While the ShowContent() function is available in object scripts through both IntelliSense and the IDE function browser, its use in object scripts is not supported.

Category

Graphic Client

Syntax

To show content, in this case a 2-way valve symbol, within a specified screen and pane:

```
Dim contentInfo as aaContent.ContentInfo;
contentInfo.Content = “SA_Valve_2Way”;
contentInfo.Name =”SA_Valve_2Way1”;
contentInfo.ContentType = “Level_3”;
contentInfo.PaneName =”Pane 2”;
contentInfo.ScreenName =”Primary>”;
contentInfo.SearchScope = aaContent.SearchScope.Self;
ShowContent( contentInfo );
```

Parameter

ContentInfo

Data type

aaContent.ContentInfo

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

aaContentInfo.ContentInfo Properties

ContentInfo is a predefined structure that contains the data members described in the following table.

String properties can be a concatenation of string and/or custom properties.

Note: See "Terms," below, for definitions of Content Type, Layout, Pane, Screen Profile, Primary Screen, and Source Screen.

Content	*The **Content** property is a unique name for an item, either in the Visualization folder or associated with an asset, that specifies the content to be loaded into the pane. Content can be a graphic, a layout, or external content.* *This is a **required** property of data type **string**.* ***Content** is the name of the item within the Visualization folder or associated with an asset. The content names are the names shown in the Visualization folder. The Properties tab of the Layout and ViewApp editors lists content name as the Content property.* *Relative names, for example, "Me.S1," can also be used to designate content.* *If the specified content is already shown, invoking **ShowContent** again closes the open content and reopens it.* *However, if the content is a graphic and you are using object wizards that include Symbol Wizard custom property selections, and the graphic has an owning object, use the graphic's absolute name. This allows the correct graphic configuration to be shown for the instance. See Owning Object, below, for more information.* ***Content** name must be unique. Application Server does not check for duplicated names. If **Content** is duplicated, open content with the same name is closed, and the content with the duplicated name is opened in its place. PropertyOverrides and other ContentInfo parameters are updated with any new specified values.* *If the same content item is used in multiple panes of the layout, and the **Content** property is specified by the **HideContent()** method, all instances of the content item are hidden. To specify a single instance, use the **Name** property instead.* *Example:* `contentInfo.Content =`

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

	`"UserDefinedObject_001.Symbol_001";`
Name	***Name** is an auto-generated (or user-edited) identifier of a unique content item. The name is created when the content item is added to a layout pane.* *This is an **optional** property of data type **string**.* ***Name** must be unique within a layout. **Name** can be duplicated in nested layouts, as long as the name is not duplicated within a nested single layout or the top level layout.* *If content with the same **Name** is open within the **SearchScope**, the matching, open content is closed. A new instance of the matching content opens in the pane specified by the **ShowContent** call.* *When **Name** is specified in a **HideContent** call, only the uniquely-named content is closed. If **Content** is specified and the **Name** property is not specified, all items in the layout with the same content name are hidden.* *Example:* `contentInfo.Name = "Symbol_011";`
ScreenName	***ScreenName** specifies the screen that contains a pane in which to place the content.* *This is an **optional** property of data type **string**.* *ScreenNames are configured in the **Screen Profile Editor**. See Screen Profiles in the AVEVA OMI Help for additional information.* *Example:* `contentInfo.ScreenName = "Wall";`
PaneName	***PaneName** specifies the pane in which to place the content.* *This is an **optional** property of data type **string**.* *PaneNames are configured in the **Layout Editor**. See Layouts in the AVEVA OMI Help for additional information.* *Example:* `contentInfo.PaneName = "Pane1";`
ContentType	***ContentType** specifies the content type, for example, "Overview," "Navigation," or "Faceplate."* *This is an **optional** property of data type **string**.* ***ContentType** is matched against the Content Type parameter that can be set for a pane in the **Layout***

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

	Editor. ***ContentType*** *is used to override the actual type of the specified* ***Content***. *If* ***ContentType*** *is not specified,* ***Content*** *is examined for its type of content. See Layouts in the AVEVA OMI Help for additional information about content types.* *Example:* `contentInfo.ContentType = "Overview";`
SearchScope	***SearchScope*** *specifies which screen or screens will be searched for a pane that matches the specified* ***PaneName*** *or* ***ContentType*** *when* ***ScreenName*** *has not been specified.* *This is an* ***optional*** *property of data type* ***enum***. *The default value for* ***SearchScope*** *is "Self."* ***SearchScope*** *is an enum with the following values:* • *"Self" searches for matching content within the panes of the NESTED layout (an embedded layout) from which the* ***ShowContent*** *call was made. If* ***SearchScope*** *is not specified, "Self" is the default.* *In contrast to "Self," the remaining* ***SearchScope*** *values reference only the top level layout, not nested layouts.* • *"AllScreens" searches for matching content within the panes of all screens in the top level layout. The search starts with the source screen, then the primary screen, and then any remaining screens in alphabetical order.* • *"SourceScreen" searches for matching content only within the panes of the top level layout from which ShowContent was called.* • *PrimaryScreen searches for matching content only within the panes of the screen designated in the Screen Profile as the primary screen.* ***Nested layouts:*** • *If* ***SearchScope*** *is "Self" or is not specified, and* ***ShowContent*** *is called from a nested (embedded) layout,* ***ShowContent*** *searches for a matching pane within the nested layout.* • *When* ***SearchScope*** *is "All," "Source," or "Primary," or if the* ***ScreenName*** *is specified,* ***ShowContent*** *searches for matching content within the top-level layout only.* *Example:*

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

```
contentInfo.SearchScope =
aaContent.SearchScope.PrimaryScreen;
```

PropertyOverrideValue

PropertyOverrideValue *is a ValuePair[] array that is used to set custom property overrides if a graphic has been specified by the* ***Content*** *property. Each override must include the custom property name and the override value.*

This is an ***optional*** *property of data type* ***ValuePair[]****.*

Property overrides are specified as a key-value pair, with the property name enclosed in quotes.

Example:

```
Dim cpValues [2] as
aaContent.PropertyOverrideValue;
cpValues[1] = new
aaContent.PropertyOverrideValue("CP1",
"20", true);
cpValues[2] = new
aaContent.PropertyOverrideValue("CP2",
"Pump.PV.TagName", false);
```

OwningObject

OwningObject *sets the owning object of the content shown by the* ***ShowContent()*** *script function.*

This is an ***optional*** *property of data type* ***string****.*

OwningObject *can be a concatenation of constant strings and reference strings, and can be browsed using the Display Automation Object Browser. You can also type the name of the owning object.*

The ***OwningObject*** *property sets references for the graphic, but is not associated with the* ***GraphicName*** *property, if the graphic is part of an Object Wizard. Therefore, if you are scripting a graphic with an owning object, specify the owning object name as part of the* ***GraphicName*** *property, for example,* `UserDefined_001.Pump_001`*.*

Example:

```
contentInfo.OwningObject = "Enterprise";
```

Summary of ContentInfo properties

Property Name	Required (y/n)	Data Type	Default Value
Content	Yes	String	None
Name	No	String	None

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Property Name	Required (y/n)	Data Type	Default Value
ScreenName	No	String	None
PaneName	No	String	None
ContentType	No	String	None
SeachScope	No	Enum	Self
PropertyOverrides	No	ValuePair[] array	None
OwningObject	No	String	None

Terms

Content type: specifies the type of content represented by a pane.

Content: the name of a graphic, layout, or external content item as it is listed within the the Visualization folder. This is displayed as the Content property in the Layout Editor when the content item is added to a layout.

Name: the unique name assigned to an instance of a content item, when it is added to a layout. This is displayed as the Name property in the Layout Editor when the content item is added to a layout, and can be edited.

Layout: consists of one or more rectangular areas called panes that contain content shown in a ViewApp. A layout is associated with a screen, or it can be embedded within a pane of another layout.

Embedded or Nested Layout: In the context of ShowContent and HideContent, an embedded layout is a layout that is placed inside a pane of a containing layout. When SearchScope is "Self" (default), embedded layouts are searched for content that matches the parameters specified in the ShowContent/HideContent call.

Pane: rectangular area of a layout that can hold a single piece of content.

Primary screen: represents the main screen of a workstation that will show a running ViewApp

Screen Profile: defines the physical characteristics of one or more client workstation screens that will show a running ViewApp and how these screens are arranged with respect to each other.

Source screen: screen from which ShowContent or HideContent was called.

Example

Show content in a pane

```
dim contentInfo as aaContent.ContentInfo;

Dim cpValues [2] as aaContent.PropertyOverrideValue;
cpValues[1] = new aaContent.PropertyOverrideValue("CP1", "20", true);
cpValues[2] = new aaContent.PropertyOverrideValue("CP2", "Pump.PV.TagName", false);

contentInfo.Content = "Symbol1";
contentInfo.Name = "S12";
contentInfo.ContentType = "Overview";
contentInfo.OwningObject = "Enterprise";
contentInfo.PaneName = "Pane 1";
contentInfo.ScreenName = "Wall";
contentInfo.PropertyOverrideValues = cpValues;
```

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

```
contentInfo.SearchScope = aaContent.SearchScope.PrimaryScreen;

ShowContent ( contentInfo );
```

See also

HideContent()

ShowGraphic()

ShowGraphic()

Shows a graphic within a pop-up window. The ShowGraphic() function has been extended to call InTouch HMI Windows. This function is available within any Industrial Graphics client script.

Category

Graphic Client

Syntax

To show a graphic within a pop-up window:

```
Dim graphicInfo as aaGraphic.GraphicInfo;
graphicInfo.Identity = "<Identity>";
graphicInfo.GraphicName = "<SymbolName>";
ShowGraphic( graphicInfo );
```

To call an HMI window

```
Dim graphicInfo as aaGraphic.GraphicInfo;
graphicInfo0.Identity = "<<HMIName>:WindowName>";
ShowGraphic( graphicInfo );
```

Parameter

GraphicInfo

Data Type

aaGraphic.GraphicInfo

aaGraphic.GraphicInfo properties

GraphicInfo is a predefined structure that contains the data members described below.

Identity	*The **Identity** property is a unique name that identifies which instance has opened the graphic.* ***Identity** cannot contain more than 329 characters and*

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

	must contain at least one letter. Valid characters are alphanumeric and special characters ($, #, _). *This is a **required** property of data type **string**.* *Default value: **none*** *Example:* `graphicInfo.Identity = "i1";`
GraphicName	***GraphicName** is the name of the graphic to show.* ***GraphicName** cannot contain more than 329 characters and must contain at least one letter. Valid characters are alphanumeric and special characters ($, #, _).* *This is a **required** property of data type **string**.* *Default value: **none*** *Browse using your HMI/SCADA application browser, such as the **Display Galaxy Browser** or equivalent in your HMI/SCADA software, or directly type the graphic name. The application name can come from:* • *Visualization folder, for example: `"Symbol_001"`* • *Instances, absolute or hierarchical, for example: `"Userdefined_001.Symbol1"`, `"Userdefined_001.Pump_001.S1"`* • *Relative reference, for example: `"Me.Symbol_001"`* *Use an absolute name to specify the **GraphicName** and **OwningObject** if you are using an Object Wizard with Symbol Wizard custom property selections. This allows the correct graphic configuration to be shown for the instance. See **OwningObject**, below, for more information.* *If you type any invalid character or exceed the character limit, the system shows a warning message at run time. There is no validation at design time.* *The graphic name can be a concatenation of constant strings and reference strings. For example:* `"Pump_001" + ".Symbol_001"; cp1 + ".Symbol_001", where the value of cp1 = "Pump_001"; or Obj1.Str1 + ".Symbol_001", where the value of Obj.Str1 = "Pump_001".` *Examples:*

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

	To specify a Visualization folder reference:
	`graphicInfo.GraphicName = "S1";`
	To specify an absolute reference:
	`graphicInfo.GraphicName = "OwningObjectName.SymbolName";`

OwningObject	*The name of the owning object of the graphic shown by the **ShowGraphic()** script function.*
	***OwningObject** is an optional property. It can be a concatenation of constant strings and reference strings. **OwningObject** can be browsed using the **Display Automation Object Browser**, or you can type the name of the owning object.*
	*The **OwningObject** property sets references for the graphic, but is not associated with the **GraphicName** property if the graphic is part of an Object Wizard. Therefore, if you are scripting a graphic with an owning object, specify the owning object name as part of the **GraphicName** property, for example:*
	`UserDefined_001.Pump_001.`
	*This is an **optional** property of data type **string**.*
	*Default Value: **empty***
	Example:
	`graphicInfo.OwningObject = "UserDefined_001";`

HasTitleBar	***HasTitleBar d**etermines if the graphic is shown with a title bar.* *This is an **optional** property of data type **Boolean**.* *Default value: **True*** *Example:* `graphicInfo.HasTitleBar = false;`
WindowTitle	***WindowTitle** specifies the title shown in the window title bar.* *This is an **optional** property of data type **string**.* *Default value: **Empty*** *Valid range: Limit of 1024 characters.* ***WindowTitle** can be a constant string, a reference, or an expression. If you change the owning object for an AutomationObject graphic, **WindowTitle** is updated accordingly. If **WindowTitle** is empty, the value of the **Identity** property is shown on the title bar.*

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

	Example: *This is an **optional** property of data type **Boolean**.* `graphicInfo.WindowTitle = "Graphic01";`

WindowType	***WindowType** specifies whether window type is modal or modeless.*
	*This is an **optional** property of data type **enum**.*
	*Default value: **Modeless***
	Valid range: 0 or 1 where
	• *0 = Modal*
	• *1 = Modeless.*
	Examples:
	`graphicInfo.WindowType = aaGraphic.WindowType.<windowtype>;`
	`graphicInfo.WindowType = 1;`

HasCloseButton	***HasCloseButton** determines if the pop-up window has a close button.* *This is an **optional** property of data type **Boolean**.* *Default value: **True*** *Example:* `graphicInfo.HasCloseButton = false;`
Resizable	***Resizable** determines if the pop-up window is resizable.* *This is an **optional** property of data type **Boolean**.* *Default value: **False*** *Example:* `graphicInfo.Resizable = true;`
WindowLocation	***WindowLocation** specifies the location of the pop-up window.* *This is an **optional** property of data type **enum**.* *Default value: **0** (center)* *Valid range: 0 through 12, where the window location is set by the corresponding enumeration:* *0 = Center* *1 = Above* *2 = TopLeftCorner* *3 = Top* *4 = TopRightCorner* *5= LeftOf*

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

	6= LeftSide
	7= RightSide
	8 = RightOf
	9 = BottomLeftCorner
	10 = Bottom
	11 = BottomRightCorner
	12 = Below
	*If you select **Desktop** as the **WindowRelativePosition** (see below), Above (1), LeftOf (6), RightOf (8), and Below (12) are invalid. For more information about the behavior of the WindowLocation parameter, see Working with the Show/Hide Graphics Script Functions in the Creating and Managing Industrial Graphics User Guide.*
	Examples:
	`graphicInfo.WindowLocation = aaGraphic.WindowLocation.<WindowLocation>;`
	`graphicInfo.WindowLocation = 1;`

WindowRelative Position	***WindowRelativePosition s**pecifies the relative position of the pop-up window.*
	*This is an **optional** property of data type **enum**.*
	*Default value: **0** (desktop)*
	Valid range: 0 through 8, where the window relative position is set by the corresponding enumeration.
	0 = Desktop
	1 = Window
	2 = ClientArea
	3 = ParentGraphic
	4 = ParentElement
	5 = Mouse
	6 = DesktopXY
	7 = WindowXY
	8 = ClientAreaXY
	Examples:
	`graphicInfo.WindowRelativePosition = aaGraphic.WindowRelativePosition.<WindowRelativePosition>;`
	`graphicInfo.WindowRelativePosition = 1;`

RelativeTo	***RelativeTo** specifies the size of the pop-up window relative to the graphic, desktop, or customized width and height.* *This is an **optional** property of data type **enum**.* *Default value: **0** (Graphic)*

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

	Valid range: 0, 1, 2 where
	0 = Graphic
	1 = Desktop
	2 = CustomizedWidthHeight
	*If you enter **aaGraphic.RelativeTo.CustomizedWidthHeight,** you can include the values of the height and width in the script. Otherwise, the default values are used.*
	Examples:
	`graphicInfo.RelativeTo = aaGraphic.RelativeTo.<RelativeTo>;`
	`graphicInfo.RelativeTo = 1;`

X	***X** specifies the horizontal position of the pop-up window.* *This is an **optional** property of data type **integer**.* *Default value: **0*** *Valid range: -2,147,483,648 through 2,147,483,647* *If **X** is beyond the integer range, an overflow message appears in the Logger at run time.* *This parameter is applicable only if the value of the **WindowRelativePosition** parameter is **DesktopXY**, **WindowXY**, or **ClientAreaXY**. Unlike the ShowSymbol animation, there is no boundary for this value.* *Example:* `graphicInfo.X = 100;`
Y	***Y** specifies the vertical position of the pop-up window.* *This is an **optional** property of data type **integer**.* *Default value: **0*** *Valid range: -2,147,483,648 through 2,147,483,647* *If **Y** is beyond integer range, a proper overflow message will appear in the Logger at run time. This value is applicable only if **WindowRelativePosition** is **DesktopXY**, **WindowXY**, or **ClientAreaXY**. Unlike the ShowSymbol animation, there is no boundary for this value.* *Example:* `graphicInfo.Y = 100;`
Width	***Width** specifies the width of the pop-up window.* *This is an **optional** property of data type **integer**.* *Default value: 100* *Valid range: 0–10000*

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

	*Applicable only if **RelativeTo** is **CustomizedWidthHeight.** You can specify either the height or the width of the pop-up window. The system calculates the other, based on the aspect ratio of the graphic. If you enter an out-of-boundary value, the system shows an "Out of range" message at run time. If the value > 10000, it is set at 10000. If the value < 0, it is set at 0.* *Example:* `graphicInfo.width = 500;`
Height	***Height** specifies the height of the pop-up window.* *This is an **optional** property of data type **integer**.* *Default value: 100* *Valid range: 0–10000* ***Height** is applicable only if **RelativeTo** is the value of the **CustomizedWidthHeight** parameter. You can specify either the height or the width of the pop-up window. The system calculates the other, based on the aspect ratio of the graphic. If you enter an out-of-boundary value, the system shows an "Out of range" message at run time. If the value > 10000, it is set at 10000. If the value < 0, it is set at 0.* *Example:* `graphicInfo.height = 500;`
TopMost	***TopMost** sets a value that indicates whether the **ShowGraphic** appears in the top most z-order window. A **ShowGraphic** whose **Topmost** property is set to true appears above all windows whose **TopMost** properties are set to false (same as Windows Task Manager).* *This is an **optional** property of data type **Boolean**.* *Default value: **False*** *In a group of windows that have the **Topmost** property set to true, the active window is the topmost window. Do not create scripts that launch a non-TopMost Modal dialog from a TopMost dialog. Users will not be able to interact with the View if the Modal dialog is completely hidden by any TopMost window.* *Example:* `graphicInfo.TopMost = true;`
ScalePercentage	***ScalePercentage s**ets the scaling percentage of the pop-up window and the graphic it contains.* *This is an **optional** property of data type **integer**.*

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

	Default value: ***100*** *Valid range: 0–1000* *If you enter an out-of-boundary value, the system shows an "Out of range" message at run time. If the value > 1000, it is set at 1000. If the value < 0, it is set at 0.* *Example:* `graphicInfo.ScalePercentage = 150;`
KeepOnMonitor	***KeepOnMonitor*** *specifies that a pop-up window appears entirely within the boundaries of an application window.* *This is an* ***optional*** *property of data type* ***Boolean****.* *Default value:* ***True*** *Example:* `graphicInfo.KeepOnMonitor = true;`
StretchGraphicToFit WindowSize	***StretchGraphicToFitWindowSize*** *determines if the graphic is scaled to the current size of the pop-up window.* *This is an* ***optional*** *property of data type* ***Boolean****.* *Default Value:* ***True*** *Applicable only if the value of the* ***ScalePercentage*** *parameter is greater than 100.* *Example:* `graphicInfo.StretchGraphicToFitWindowSize = false;`
StretchWindowTo ScreenWidth	***StretchWindowToScreenWidth*** *determines if the pop-up window is scaled to the same width as the screen.* *This is an* ***optional*** *property of data type* ***Boolean****.* *Default value:* ***False*** *Applicable only if the WindowRelativePosition parameter is Desktop, Window, Client Area, ParentGraphic, or ParentElement.* *Example:* `graphicInfo.StretchWindowToScreenWidth = true;`

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

StretchWindowTo ScreenHeight	***StretchWindowToScreenHeight*** *determines if the pop-up window is scaled to the same height as the screen.* *This is an* ***optional*** *property of data type* ***Boolean****.* *Default value:* ***False*** *Applicable only if the* ***WindowRelativePosition*** *parameter is set to Desktop, Window, Client Area, ParentGraphic, or ParentElement.* *Example:* `graphicInfo.StretchWindowToScreenHeight = true;`

CustomProperties	***CustomProperties*** *is a ValuePair[] array that is used to set the custom properties of the graphic being shown.*
	This is an ***optional*** *property of data type* ***ValuePair[]****.*
	The first three parameters are:
	• *Custom property name*
	• *value*
	• *,IsConstant*
	Both custom property and the value can be a constant string, reference, or concatenation of strings. If the parameter IsConstant = True, the value is treated as a constant. Otherwise, the value is treated as a reference. The array index starts at 1.
	Example:
	`Dim cpValues [4] as aaGraphic.CustomPropertyValuePair;`
	`cpValues[1] = new aaGraphic.CustomPropertyValuePair("CP 1", 20, true);`
	`cpValues[2] = new aaGraphic.CustomPropertyValuePair("CP 2", Pump.PV.TagName, true);`
	`cpValues[3] = new aaGraphic.CustomPropertyValuePair("CP 3", "CP"+var1, CP2 + "001" + ".Speed", true);`
	`cpValues[4] = new aaGraphic.CustomPropertyValuePair("CP 3", "<HMIName>:Tag1", false);`
	`graphicInfo.CustomProperties = cpValues;`

Summary of GraphicInfo properties

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Property Name	Required	Data Type	Default Value
Identity	Yes	String	None
GraphicName	Yes	String	None
OwningObject	No	String	None
HasTitleBar	No	Boolean	Empty
WindowTitle	No	String	Empty
WindowType	No	Enum	Modeless
HasCloseButton	No	Boolean	True
Resizable	No	Boolean	False
WindowLocation	No	Enum	Center
WindowRelativePosition	No	Enum	Desktop
RelativeTo	No	Enum	Graphic
X	No	Integer	0
Y	No	Integer	0
Width	No	Integer	100
Height	No	Integer	100
TopMost	No	Boolean	False
ScalePercentage	No	Integer	100
KeepOnMonitor	No	Boolean	True
StretchGraphicToFitWindowSize	No	Boolean	True
StretchWindowToScreenWidth	No	Boolean	False
StretchWindowToScreenHeight	No	Boolean	False
CustomProperties	No	ValuePair[] array	Empty

Remarks

- Any parameter that has default value in the GraphicInfo is optional. If no input value specified for these

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

parameters, the default values are used at run time.

- Any parameter except the Enum data type can be a constant, reference, or expression.
- String properties can be a concatenation of strings and/or custom properties.
- For more information, see "Working with the Show/Hide Graphics Script Functions" in the *Industrial Graphic Editor User Guide*.

Examples for ShowGraphic

Basic script example:

```
Dim graphicInfo as aaGraphic.GraphicInfo;
graphicInfo.Identity = "Script_001";
graphicInfo.GraphicName = "Symbol_001";
ShowGraphic( graphicInfo );
```

Advanced script example:

```
Dim graphicInfo as aaGraphic.GraphicInfo;
Dim cpValues [2] as aaGraphic.CustomPropertyValuePair;
cpValues[1] = new aaGraphic.CustomPropertyValuePair("CP1", 20, true);
cpValues[2] = new aaGraphic.CustomPropertyValuePair("CP2", "Pump.PV.TagName", false);
graphicInfo.Identity = "i1";
graphicInfo.GraphicName = "S1";
graphicInfo.OwningObject = "UserDefined_001";
graphicInfo.WindowTitle = "Graphic01";
graphicInfo.Resizable = false;
graphicInfo.CustomProperties=cpValues;
ShowGraphic( graphicInfo );
```

Where "i1" is string Identity and the graphic "S1" contains custom property CP1 and CP2.

Show graphic within a pop-up window

```
ShowGraphic (graphicInfo);
```

Show an HMI window

```
Dim graphicInfo0 as aaGraphic.GraphicInfo;
graphicInfo0.Identity = "<HMIName>:Window1";
ShowGraphic( graphicInfo0 );
```

See also

HideGraphic()

HideSelf()

ShowContent()

ShowLoginDialog()

Action script that shows a login dialog box in an AVEVA OMI ViewApp with fields to enter a username and password. A typical login interface includes a login button that is selected by the user to show the **Login** dialog box with fields to enter a username and password. ShowLoginDialog() is for use with AVEVA OMI only.

Action scripts are graphic animations that are triggered by a user action such as a mouse click.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Category

Miscellaneous

Syntax

```
ShowLoginDialog() ;
```

Parameter

None

Trigger

On Left-Click/Key/Touch Down

Additional information

A log off button can be added that uses the Logoff() method to allow the user to log off from the HMI/SCADA application.

Example

```
ShowLoginDialog() ;
```

See also

Logoff()

InTouch functions

The following InTouch functions can be used within the script editor contained in the **Industrial Graphic Editor**. In all functions that specify tag names as parameters, you can use InTouch tags from your InTouch application.

InTouch script functions can be used *only* in graphic scripts. InTouch script functions do not work in ArchestrA object scripts. Even though the object script will not work, no error or warning is generated.

Note: Using the **Convert to Industrial Graphic** option in InTouch scripts may wrongly append the term *"InTouch:"* to the script function name. To avoid errors, remove the term *"InTouch:"* from the script function name.

AddPermission() function

Assigns a certain InTouch access level to a given user group on the local system or on the domain. When a user belonging to that group logs on to the InTouch HMI after the AddPermission() function is called, he or she receives the specified access level.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Category

security

Syntax

```
DiscreteTag=AddPermission( "Domain", "Group", AccessLevel);
```

Arguments

Domain

Name of the domain or local computer in which the group is located.

Group

Windows user group.

AccessLevel

InTouch access level that you want to associate with the given group.

Remarks

Valid for operating system security only. When this function is called, it checks for the presence of the specified group in the specified domain or workgroup. If successful, TRUE is returned, and the specified Access Level is associated with the group for subsequent user log ons. In all other cases, (that is, if an invalid value is specified for any of the arguments) FALSE is returned.

This function is typically configured to run on application startup. It does not affect users that are currently logged on. Only users that log on after AddPermission() is successfully called receive the access level associated with their group.

Examples

```
DiscreteTag=AddPermission( "corporate_hq", "InTouchAdmins", 9000);
DiscreteTag=AddPermission( "johns01", "InTouchUsers", 5000);
```

Operations Control connected experience

The AddPermission() method accepts only two parameters in Operations Control connected experience:

- AVEVA Connect Group
- Access Level

Script function for AddPermission() in Operations Control connected experience:

```
DiscreteTag=AddPermission("", "AVEVA Connect group", AccessLevel);
```

If a runtime user is a member of multiple groups from CONNECT, the access level will be determined by the group with the highest access level.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

See also

AttemptInvisibleLogon() function
InvisibleVerifyCredentials() function
IsAssignedRole() function
PostLogonDialog() function
QueryGroupMembership() function

AttemptInvisibleLogon() function

The AttemptInvisibleLogon() function can be used in a script to log on a user to InTouch using the supplied credentials. The user is not required to enter a password or user ID.

Category

security

Syntax

```
DiscreteTag=AttemptInvisibleLogon( "UserId", "Password", "Domain" );
```

Arguments

UserId

A valid user account name.

Password

Password of the user.

Domain

Name of the local computer, workgroup, or domain to which the user belongs. This argument applies only if the current security type is operating system-based.

Return value

Returns TRUE if authentication is successful. Otherwise, it returns FALSE.

Remarks

An attempt is made to log on to the InTouch HMI using the supplied credentials.

- If the logon attempt succeeds, then TRUE is returned and the $OperatorDomain, $OperatorName, $AccessLevel, and $Operator system tags are updated accordingly.
- If the log on attempt fails, then FALSE is returned, and the currently logged on user (if any) continues to be the current user.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

The *Domain* argument is only valid for operating system-based security. If ArchestrA security mode is in use and if ArchestrA security is in turn using operating system-based security, the *UserId* argument should contain the fully qualified user name with domain name or computer name.

Examples

When security is operating system-based:

```
DiscreteTag=AttemptInvisibleLogon("UserId", "Password", "Domain" );
```

When security is either InTouch-based or ArchestrA-based:

```
DiscreteTag=AttemptInvisibleLogon("UserId", "Password", "" );
```

See also

AddPermission() function
InvisibleVerifyCredentials() function
IsAssignedRole() function
PostLogonDialog() function
QueryGroupMembership() function

ChangePassword() function

Shows the **Change Password** dialog box, allowing the logged on operator to change his/her password.

Category

security

Syntax

```
[Result=]ChangePassword();
```

Return Value

Returns one of the following integer values:

0 = Cancel was pressed.

1 = OK was pressed.

Remarks

If the operator uses a touch screen, the operator can use the alphanumeric keyboard to enter the new password.

Example

The following script can be placed on a button or called from a condition script or data change script.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

```
Errmsg=ChangePassword();
```

EnableDisableKeys() function

Enables/disables key filters for the Alt, Escape, and Windows keys.

Category

View

Syntax

```
EnableDisableKeys(AltKey, EscKey, WinKey);
```

Parameters

AltKey

Integer to enable or disable key filters for the Alt key:

1 = enable filter (disable Alt key)

0 = disable filter (enable Alt key)

EscKey

Integer to enable or disable key filters for the Escape key:

1 = enable filter (disable Esc key)

0 = disable filter (enable Esc key)

WinKey

Integer to enable or disable key filters for the Windows key:

1 = enable filter (disable Win key)

0 = disable filter (enable Win key)

Remarks

Disabling the Alt key also disables the Win+L key combination (for locking the Windows desktop). Win+L is the shortcut for another combination of keys that involves the Alt key. Thus, disabling the Alt key also disables the shortcut for locking the Windows desktop.

Disabling the Esc key disables it for all actions.

Example(s)

```
EnableDisableKeys(0,0,0); // enable all three keys
EnableDisableKeys(1,1,1); // disable all three keys
EnableDisableKeys(0,0,1); // enable Alt and Escape keys, disable Windows key.
```

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

FileCopy() function

Copies a source file to a destination file and returns a status result. This function may take a longer time to execute and is executed in multiple stages:

1. FileCopy() function is called and an immediate result is returned, indicating success or failure of the file copy initialization.
2. FileCopy() function executes the copy procedure in the background, and InTouch scripting continues execution while the file copying is in progress. You can monitor the file copying progress with an integer tag.
3. FileCopy() function returns a file copy result, indicating success or failure of the file copy procedure.

If the destination folder is not available (i.e. another computer on the network), the function waits for up to 10 seconds to time out, and then posts a message in the Logger.

Note: Do not use the FileCopy() function in asynchronous QuickFunctions.

Syntax

```
result = FileCopy (sourcefile, destfile, progresstag)
```

Parameters

sourcefile

Full path and file name of the file to be copied. A literal string value, message tagname, or string expression. You can use the wildcard characters (* and ?) in this parameter to copy just files matching a specified criteria. The path name can also be a UNC path name.

destfile

Full path and file name (or just path name) of the destination. A literal string value, message tagname, or string expression. The path name can also be a UNC path.

progresstag

Name of an integer tag enclosed in double quotes that will contain a value indicating the file copy progress. A literal string value, message tagname (such as a message tag containing the value "IntTag.Name") or string expression. The values have following meaning:
0 - FileCopy() procedure is still in progress.
1 - FileCopy() procedure has completed successfully.
-1 - FileCopy() procedure completed with errors.

Return value

A value of -1, 0, or 1 indicating the following:

1 - FileCopy() function successfully called.
0 - Error when calling the FileCopy() function because another FileCopy() procedure is already in progress.
-1 - Error when calling the FileCopy() function because of a non-existent source file or the destination is read only.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Example(s)

This script copies the file c:\MyData\output.log to the directory d:\archive and renames the file to output.txt. The progress of the file copy is written to the integer tag Monitor.

```
Status=FileCopy("c:\MyData\output.log","d:\archive\output.txt","Monitor");
```

This script copies all files with file ending .txt in the c:\ root directory to the destination directory c:\Backup.

```
Status=FileCopy("c:\*.txt", "c:\Backup", "Monitor");
```

This script copies a file whose full path and file name is contained in the message tag LogFile to the destination directory c:\results\ and renames it to logxxx.txt where xxx is a timestamp.

```
Status=FileCopy(LogFile, "c:\results\log" + $DateString + $TimeString + ".txt", "Monitor");
```

FileDelete() function

Deletes an individual file.

Syntax

```
result = FileDelete (filename)
```

Parameters

filename

The path name and file name of the file to delete. A literal string value, message tagname, or string expression. UNC path names are supported.

Remarks

Do not use the wildcard characters (* and ?) with the FileDelete() function and do not use the FileDelete() function in asynchronous QuickFunctions.

The FileDelete() function does not delete directories.

Return value

A value indicating success or failure of the file deletion:

1 - file is deleted successfully
0 - file is not deleted successfully. Possible causes are attempts to delete a read only or a non-existent file.

Example(s)

This script deletes the file c:\Data.txt and returns 1 if the file was found and deleted successfully.

```
Status=FileDelete("c:\Data.txt");
```

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

FileMove() function

Moves a source file to a destination file and returns a status result. It can be also used to rename a file. This function may take a longer time to execute and executes in multiple stages:

1. FileMove() function is called and an immediate result is returned, indicating success or failure of the file move initialization.
2. FileMove() function executes the move procedure in the background, InTouch scripting continues execution while the file moving is in progress. You can monitor the file moving progress with an integer tag.
3. FileMove() function returns a file move result, indicating success or failure of the file moving procedure.

Do not use the FileMove() function in asynchronous QuickFunctions.

Syntax

```
result = FileMove (sourcefile, destfile, progresstag)
```

Parameters

sourcefile

Full path and file name of the file to be moved. A literal string value, message tagname, or string expression. You can use the wildcard characters (* and ?) in this parameter to move just files matching a specified criteria. The path name can also be a UNC path name.

destfile

Full path and file name (or just path name) of the destination. A literal string value, message tagname, or string expression. The path name can also be a UNC path.

progresstag

Name of an integer tag enclosed in double quotes that will contain a value indicating the file moving progress. A literal string value, message tagname (such as a message tag containing the value "IntTag") or string expression. The values have following meaning:

0 - FileMove() procedure is still in progress

1 - FileMove() procedure has completed successfully

-1 - FileMove() procedure completed with errors

Return value

A value of-1, 0, or 1 indicating the following:

1 - FileMove() function successfully called
0 - Error when calling the FileMove() function because another FileMove() procedure is already in progress
-1 - Error when calling the FileMove() function. Possible errors are attempts to move a non-existent file.

Example(s)

This script moves the file c:\MyData\output.log to the directory d:\archive and renames the file to output.txt.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

The progress of the file moving is written to the integer tag Monitor.

```
Status=FileMove("c:\MyData\output.log","d:\archive\output.txt","Monitor");
```

This script moves all files with file ending .txt in the c:\ root directory to the destination directory c:\Backup.

```
Status=FileMove("c:\*.txt", "c:\Backup", "Monitor");
```

This script moves a file whose full path and file name is contained in the message tag LogFile to the destination directory c:\results\ and renames it to logxxx.txt where xxx is a timestamp.

```
Status=FileMove(LogFile, "c:\results\log" + $DateString + $TimeString + ".txt", "Monitor");
```

FileReadFields() function

Reads the values contained in a csv file into a series of tagnames. You can use this function to load a set of tagname values.

Commas are the only supported delimiter.

This function can only be used for synchronous calls.

Syntax

```
[result = ] FileReadFields (filename, offset, starttag, numberoffields)
```

Parameters

filename

Name of the csv file to read the data from. A literal string value, a message tagname or a string expression.

offset

Location (in bytes) in the file to start reading. A literal integer value, integer tagname, or integer expression.

starttag

Name of the first tagname that receives the first read data item. The tagname must be enclosed with double quotes and end in a number, such as "MyTag1". A literal string value, message tagname (such as a message tagname containing the value "MyTag1"), or a string expression.

numberoffields

Number of data items to read from the csv file. A literal integer value, integer tagname, or integer expression. The first data item is read into the tagname defined in the starttag parameter, subsequent data items into tagnames with the incremented numeral suffix of the starttag parameter (MyTag1, MyTag2, MyTag3, ...).

Return value

Optional new file offset (in byte) after reading the data. This can be used to read the next set of data.

Example(s)

This script reads the values "Flour" to RecipeTag1, 27.23 to RecipeTag2, 14 to RecipeTag3, and 1 to RecipeTag4, and returns the new file offset—if the csv file c:\set.csv contains the following data: Flour, 27.23,14,1 and if the following tags are defined: RecipeTag1:message, RecipeTag2:real, Recipe3:integer, RecipeTag4:discrete.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

```
FileReadFields("c:\set.csv",0,"RecipeTag1",4);
```

FileReadMessage() function

Reads a specified number of bytes (or one line) of string data from a file.

Syntax

```
[result = ] FileReadMessage (filename, offset, messagetag, charstoread)
```

Parameters

filename

Name of the file to read the data from. A literal string value, a message tagname, or a string expression.

offset

Location (in bytes) in the file to start reading from. A literal integer value, integer tagname, or integer expression.

messagetag

Message tagname that receives the first line or number of bytes from the file. Enclose the tagname with double quotes when using the function within the Industrial Graphics Editor Script Editor.

charstoread

Number of bytes to read from the file. Set it to 0 to read until the next line feed (LF) character. A literal integer value, integer tagname, or integer expression.

Return value

Contains the new byte position after the read. You can use this for subsequent reads from the file.

Example(s)

This script reads the first line of data in the file c:\Data\File.txt to the message tagname MsgTag.

```
FileReadMessage ("c:\Data\File.txt",0,MsgTag, 0);
FileReadMessage ("c:\Data\File.txt",0,"InTouch:MsgTag", 0);
```

FileWriteFields() function

Writes the values contained in a series of tagnames to a csv file. You can use this function to save a set of tagname values.

Commas are the only supported delimiter.

Syntax

```
[result = ] FileWriteFields (filename, offset, starttag, numberoffields)
```

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Parameters

filename

Name of the csv file to write the data to. A new file is created if it does not previously exist. A literal string value, a message tagname, or a string expression.

offset

Location (in bytes) in the file to start writing to. Use -1 to write to the end of the file (append). A literal integer value, integer tagname, or integer expression.

starttag

Name of the first tagname that contains the first data item to be written. The tagname must be enclosed with double quotes and end in a number, such as "MyTag1". A literal string value, message tagname (such as a message tagname containing the value "MyTag1") or a string expression.

numberoffields

Number of data items to write to the csv file. A literal integer value, integer tagname, or integer expression. The first data item is written from the tagname defined in the starttag parameter to the file, subsequent data items from tagnames with the incremented numeral suffix of the starttag parameter (MyTag1, MyTag2, MyTag3, ...).

Return value

Optional new file offset (in byte) after writing the data. This can be used to write the next set of data.

Example(s)

A series of InTouch tags is defined as follows:

Tagname	Data Type	Value
RecipeTag1	Message	Flour
RecipeTag2	Real	27.23
RecipeTag3	Integer	14
RecipeTag4	Discrete	1

This script writes the values contained in RecipeTag1 to RecipeTag4 to the csv file c:\set.csv.

```
FileWriteFields("c:\set.csv",0,"RecipeTag1",4);
```

So that the file c:\set.csv will contain the following data:

```
Flour,27.23,14,1
```

FileWriteMessage() function

Writes a specified number of bytes (or one line) of string data to a file.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Syntax

```
[result = ] FileWriteMessage (filename, offset, messagetag, linefeed)
```

Parameters

filename

Name of the file to write the data to. A literal string value, a message tagname, or a string expression.

offset

Location (in bytes) in the file to start writing to. Set it to -1 to write data to the end of the file (append). A literal integer value, integer tagname, or integer expression.

messagetag

Message tagname that contains the data to be written to the file.

linefeed

Specifies whether to write a line feed (LF) character after writing the data to the file. Set to 1 to write a line feed character; otherwise, set it to 0. A literal Boolean value, discrete tagname, or Boolean expression.

Return value

Contains the new byte position after the write. You can use this for subsequent writes to the file.

Example(s)

This script writes the value of a message tagname MsgTag to the end of the file c:\Data\File.txt.

```
FileWriteMessage("c:\Data\File.txt",-1,MsgTag,1);
```

GetAccessToken() Function

GetAccessToken () provides authentication token, which will be accepted by controls like Trend Control, Alarm Client Control and other InTouch controls, to support Single Sign-On functionality.

Syntax

The syntax of the script function is as follows:

```
resultCode = GetAccessToken();
```

Resultcode indicates the latest access token.

Example:

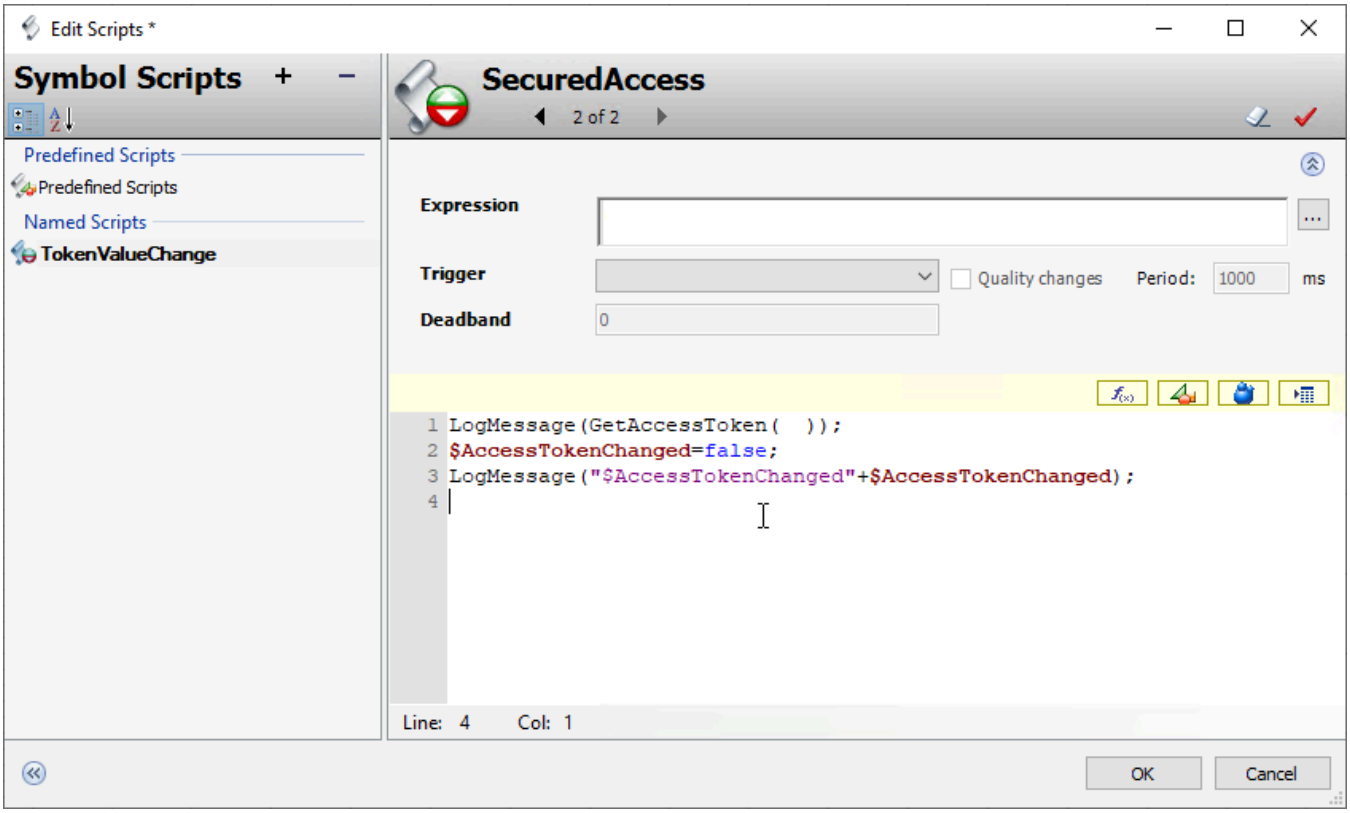
To get the token value renewed every time it expires, you can use the script as follows:

1. In the Symbol Script window, select **Add Script**.
2. Provide a name to the script.
3. Select the **Display Script Function Browser** icon.
4. In the **Script Browser** screen, under **InTouch,** select **GetAccessToken** to insert the script function or you can

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

manually type.

5. Select **OK**.



Return value

The GetAccessToken script function returns the access token with the a token value, if user is authenticated using CONNECT.

The GetAccessToken script function returns an empty string, if user is not authenticated using CONNECT.

GetAccountStatus() function

Returns the number of days until the user's password expires.

Category

security

Syntax

```
Result=GetAccountStatus(Domain, UserID);
```

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Arguments

Domain

Name of the domain or local computer in which the user account is located.

UserID

Windows user account name that is part of the local computer, workgroup, or domain.

Return value

This function also returns the following values:

Result	Description
-1	Account password expired
-2	Account password never expires
-3	Account locked out
-4	Account disabled
-5	Account info failed

Remarks

Use this script function with operating system-based security. Do not use this function with the ArchestrA security mode.

If the GetAccountStatus() function is used with ArchestrA security, the script attempts to retrieve the account information directly from the domain controller. This works as long as the ArchestrA Galaxy Repository is using operating system security with the same domain.

Example(s)

```
Status = GetAccountStatus("Corporate_HQ","Operator");
```

GetNodeName() function

Returns the node name of the computer.

Syntax

```
GetNodeName (messagetag, nodenum);
```

Parameters

messagetag

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Message tagname that will contain the node name. Enclose the tagname with double quotes when using the function within the Industrial Graphics Editor Script Editor.

nodenum

Number of characters to retrieve from the node name. A literal integer value, integer tagname, or integer expression in the range of 0 to 131.

Example(s)

This script retrieves the node name and assigns it to the NodeName message tagname.

```
GetNodeName(NodeName,131);
GetNodeName("InTouch:NodeName",131);
```

GetSecureAccessToken() Function

GetSecureAccessToken () provides secure authentication token, which will be accepted by controls like Trend Control, Alarm Client Control and other InTouch controls, to support Single Sign-On functionality.

Syntax

The syntax of the script function is as follows:

```
resultCode = GetSecureAccessToken();
```

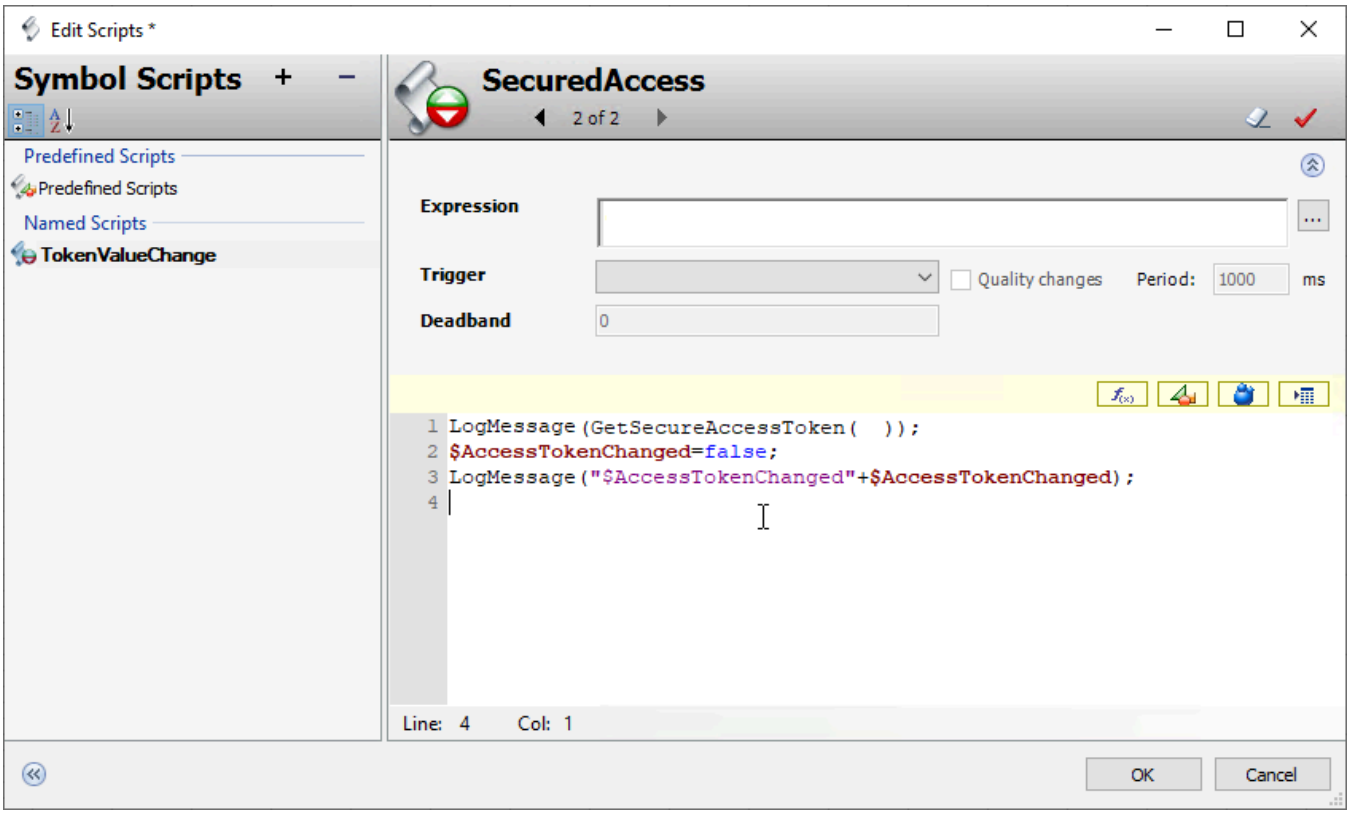
Resultcode indicates the latest access token.

Example:

To get the secured token value renewed every time it expires, you can use the script as follows:

1. In the Symbol Script window, select **Add Script**.
2. Provide a name to the script.
3. Select the **Display Script Function Browser** icon.
4. In the **Script Browser** screen, under **InTouch,** select **GetSecureAccessToken** to insert the script function or you can manually type.
5. Select **OK**.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.



Return Value

The GetSecureAccessToken script function returns the access token with the a token value, if user is authenticated using CONNECT.

The GetSecureAccessToken script function returns an empty string, if user is not authenticated using CONNECT.

GetTokenConnectionStatus() function

Retrieves status of the connection to the AVEVA Identity Manager and CONNECT in AVEVA Operations Control connected experience.

Syntax

The syntax of the script function is as follows:

```
resultcode = GetTokenConnectionStatus();
```

Resultcode indicates the token for connection status.

Return value

- Resultcode is 0 when the connection status is Fully Connected. That is the application is connected to both AVEVA Identity Manager and CONNECT.
- Resultcode is 1 when the connection status is Partially Connected. That is the application is connected to AVEVA Identity Manager and disconnected from CONNECT.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

- Resultcode is 2 when the connection status is Fully Disconnected. That is the application is disconnected from both AVEVA Identity Manager and CONNECT.

Example:

int AccessTokenStatus=GetTokenConnectionStatus()

InfoAppTitle() function

Returns the application title or Windows task list name of a specified application that is running.

Syntax

```
result = InfoAppTitle (appname)
```

Parameters

appname

Name of the application without the .exe extension. A literal string value, message tagname, or string expression.

Example(s)

This script returns "Calculator"

```
InfoAppTitle("calc")
```

This script returns "Microsoft Excel"

```
InfoAppTitle("excel")
```

InfoDisk() function

Returns either the total or free space on a local or network disk drive.

Syntax

```
result = InfoDisk (drive, infotype, trigger);
```

Parameters

drive

The drive letter for which you want to retrieve information. Only the first character of a string is used. A literal string value, message tagname, string expression.

infotype

Specifies the information type. A literal integer value, integer tagname, or integer expression with following possible values:

1 - function returns total size of disk drive (in bytes)

2 - function returns free space of disk drive (in bytes)

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

3 - function returns total size of disk drive (in kilobytes)

4 - function returns free space of disk drive (in kilobytes)

trigger

A tagname (or expression) that acts as a trigger to recalculate the disk information. If the trigger value changes the disk information is recalculated. A discrete or analog taname, or a discrete or analog expression.

Remarks

The trigger tag only has meaning when the InfoDisk() function is used in an animation display link. If this function is used in a script, you can specify any literal numeric value, analog tagname, or numeric expression.

Example(s)

Use this script in an animation display link to show the free space of disk drive C and update the information every minute.

```
InfoDisk("C", 4, $Minute)
```

InfoFile() function

Returns various information on a file or directory.

Syntax

```
result = InfoFile (filename, infotype, trigger)
```

Parameters

filename

The full file name or directory name you want to retrieve information about. A literal string value, message tagname, or string expression. Can also include wildcard characters, such as "*" and "?".

infotype

The type of information you want to retrieve about the specified file or directory. A literal integer value, integer tagname, or integer expression with following values and meaning:

1 - Existence. The InfoFile() function returns 1 if the file exists, 2 if the file is a directory and 0 if the file or directory does not exist.

2 - Size. The InfoFile() function returns the file size in bytes.

3 - Creation timestamp. The InfoFile() function returns the time stamp as seconds that have passed since midnight January 1, 1970. Use the StringFromTimeLocal() function to convert this value to a message timestamp.

4 - Wildcard Search Match. The InfoFile() function returns the number of files that match a specified wildcard search.

trigger

A tagname (or expression) that acts as a trigger to recalculate the file information. If the trigger value changes, the file information is recalculated. A discrete or analog taname, or a discrete or analog expression.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Remarks

The trigger tag only has meaning when the InfoFile() function is used in an animation display link. If this function is used in a script, you can specify any literal numeric value, analog tagname, or numeric expression.

Example(s)

This script returns 1 if the file c:\data\log.txt exists.

```
InfoFile("c:\data\log.txt",1,$minute)
```

This script returns 14223 if the file c:\data\log.txt has a file size of 14223 bytes.

```
InfoFile("c:\data\log.txt",2,$minute)
```

This script returns 1138245266 if the file c:\data\log.txt was created on January 26, 2006 at 11:14:26 AM.

```
InfoFile("c:\data\log.txt",3,$minute)
```

This script returns 14 if there are 14 files in the directory c:\data\ that have a txt ending.

```
InfoFile("c:\data\*.txt",4,$minute)
```

InfoInTouchAppDir() function

Returns the current InTouch application directory.

Syntax

```
result = InfoInTouchAppDir();
```

Return value

A message tagname to contain the directory of the currently running InTouch application.

Remarks

The application directory name may be truncated when passed to a message tagname or shown in an animation link due to the 131 characters limitation.

Example(s)

This script may return c:\documents and settings\user1\my documents\my intouch applications\packaging.

```
InfoInTouchAppDir()
```

InTouchVersion() function

Returns the complete InTouch version number or just parts of it.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Syntax

```
result = InTouchVersion (infotype);
```

Parameters

infotype

Specifies how the version information is returned. A literal integer value, integer tagname, or integer expression with the following meaning:

0- function returns the whole version number

1- function returns just the major version number

2- function returns just the minor version number

3- function returns just the patch level

4- function returns just the build level

Example(s)

Function	Possible result
InTouchVersion(0)	10.5.1626.0521.0045.0012
InTouchVersion(1)	10
InTouchVersion(2)	5
InTouchVersion(3)	0
InTouchVersion(4)	1626

InvisibleVerifyCredentials() function

The InvisibleVerifyCredentials() function can be used in a synchronous QuickScript to verify the credentials of the given user without logging the user on to the InTouch HMI.

Category

security

Syntax

```
AnalogTag=InvisibleVerifyCredentials( "UserId", "Password", "Domain" );
```

Arguments

UserId

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Windows operating system user account name that is part of local computer, workgroup, or domain.

Password

Password for the account.

Domain

The Windows domain for the account.

Remarks

If the supplied combination of user, password, and domain are valid then the corresponding access level associated with the user is returned as an integer. Otherwise, -1 is returned.

Note: The InvisibleVerifyCredentials() function must be run from a synchronous QuickScript. The function always returns -1 if run from an asynchronous QuickScript.

This function does not change the currently logged on user. The Domain argument is only valid for operating system-based security. If ArchestrA security is in use and if ArchestrA security is in turn using operating system-based security, the UserId argument should contain the fully qualified user name with domain name or computer name.

Example

```
AnalogTag=InvisibleVerifyCredentials( "john", "Password", "corporate_hq" );
```

See also

AddPermission() function
AttemptInvisibleLogon() function
IsAssignedRole() function
PostLogonDialog() function
QueryGroupMembership() function

IsAssignedRole() function

Determines whether the currently logged on user is a member of the specified user role. Only applies to ArchestrA security.

Category

security

Syntax

```
DiscreteTag=IsAssignedRole( "RoleName" );
```

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Arguments

RoleName

The role associated with an Application Server user.

Remarks

Valid for ArchestrA security mode only and applies to the currently logged on user. If a user is currently logged on and has the *RoleName* role assigned in the Galaxy IDE, then TRUE is returned. Otherwise, FALSE is returned.

Example

```
DiscreteTag=IsAssignedRole( "Administrators" );
```

See Also

AttemptInvisibleLogon(), PostLogonDialog(), InvisibleVerifyCredentials(), QueryGroupMembership(), AddPermission()

LaunchTagViewer() function

You can start Tag Viewer only when WindowViewer is running, and only after Tag Viewer has been enabled in WindowMaker.

For information about enabling Tag Viewer, see *Configuring General WindowViewer Properties* in *AVEVA™ InTouch HMI Creating Standards for InTouch HMI Components*.

Syntax

```
LaunchTagViewer()
```

Remarks

The LaunchTagViewer() function can be executed from any script type except the application scripts OnStartup and OnShutdown.

If Tag Viewer has not been enabled in WindowMaker, calling the function will not start Tag Viewer and a warning message will appear in the logger.

You must have adequate security privileges to start Tag Viewer.

LogonCurrentUser() function

Logs on to InTouch with a user account that is currently logged on to the Windows operating system.

- InTouch configured with OS security: the user is logged on to WindowViewer.
- InTouch configured with ArchestrA security: the user must be a member of ArchestrA OS user-based or OS group-based security.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

- InTouch configured with ArchestrA OS user-based or OS group-based security and the user account is configured with smart card credentials: user is logged on using the smart card credentials. The user is logged off if the smart card is removed from the reader.

Category

security

Syntax

```
IntegerResult = LogonCurrentUser();
```

Return value

Returns -1 and no change to the values assigned to $Operator, $OperatorName, $OperatorDomain, and $AccessLevel if the logon fails.

Remarks

This function is available only in InTouch scripting, not in ArchestrA client scripting.

Example

```
IntegerResult = LogonCurrentUser();
```

See also

AddPermission() function
AttemptInvisibleLogon() function
InvisibleVerifyCredentials() function
IsAssignedRole() function
PostLogonDialog() function
QueryGroupMembership() function

PlaySound() Function

Plays a sound from a wave file or a Windows default sound.

Syntax

```
Playsound (soundname, flag)
```

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Parameters

soundname

The name of the sound or wave file. A literal string value, message tagname, or string expression. If the sound is defined as a name, it must be defined in the Win.ini file under the [Sounds] section, for example MC="c:\test.wav"

flag

Specifies how the sound is played. A literal integer value, integer tagname, or integer expression with the following meanings:
0 - Play sound one time synchronously (script execution waits until sound has finished playing).
1 - Play sound one time asynchronously (script execution does not wait until sound has finished playing).
9 - Play sound continuously (until the PlaySound() function is called again).

Example(s)

This script plays the sound of the file c:\welcome.wav one time and holds script execution until it has finished playing.

```
PlaySound("c:\welcome.wav",0);
```

This script plays the sound Alert continuously. In the win.ini file [Sounds] section you need to associate the sound name Alert with a sound file, such as:

```
Alert=c:\alert.wav.
PlaySound("Alert",9);
```

PostLogonDialog() function

Shows the InTouch **Logon** dialog box and returns TRUE.

Category

security

Syntax

```
DiscreteTag=PostLogonDialog();
```

Examples

```
DiscreteTag=PostLogonDialog();
```

See Also

InvisibleVerifyCredentials(), AttemptInvisibleLogon(), IsAssignedRole(), QueryGroupMembership(), AddPermission()

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

PrintScreen() Function

You can write a script to print the entire WindowViewer screen with the PrintScreen() function.

Syntax

```
PrintScreen (ScreenOption, PrintOption)
```

Parameters

ScreenOption

Determines how much of the WindowViewer screen is to be printed. A literal integer value, integer tagname, or integer expression.

1 - Print the client area, no menus (default)

2 - Print the entire window area, including menus

PrintOption

Determines how the printed image is to be stretched to fit on the printout.

- 1 - Best Fit:
  image is stretched so that it fits either horizontally or vertically on the printout without changing the aspect ratio. (default)
- 2 - Vertical Fit:
  image is stretched so that it fits vertically on the printout without changing the aspect ratio. The image may be cut off horizontally.
- 3 - Horizontal Fit:
  image is stretched so that it fits horizontally on the printout without changing the aspect ratio. The image may be cut off vertically.
- 4 - Stretch to Page:
  image is stretched so that it fits horizontally and vertically on the printout. The aspect ratio may change but the image is not truncated.
- Invalid options, including 0, default to Best Fit.

Note: Popup windows that extend beyond the WindowViewer screen area are cut off.

Example(s)

This script sends a printout of the current entire WindowViewer screen area without menus to the printer queue. The printout contains the screen area stretched so that it fills the printout dimensions.

```
PrintScreen(1,4);
```

QueryGroupMembership() function

Determines whether the currently logged on user is a member of the specified user group. Only applies to operating system security.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Category

security

Syntax

```
DiscreteTag=QueryGroupMembership( "Domain", "Group" );
```

Arguments

Domain

Name of the domain or local computer in which the group is located

Group

Name of the group.

Remarks

Valid for operating system security mode only and applies to the currently logged on user. If a user is currently logged on and if he or she is part of the group located on the domain, then TRUE is returned. Otherwise, FALSE is returned.

The QueryGroupMembership() function works with operating system-based security and with ArchestrA security only when the ArchestrA security is set to operating system-based security.

Examples

```
DiscreteTag=QueryGroupMembership( "corporate_hq", "InTouchAdmins" );
DiscreteTag=QueryGroupMembership( "JohnS01", "InTouchUsers" );
```

See Also

PostLogonDialog(), InvisibleVerifyCredentials(), IsAssignedRole(), AttemptInvisibleLogon(), AddPermission()

ShowHome() function

Opens the InTouch window(s) you specified in the **Home Windows** tab in the WindowViewer **Properties** dialog box and closes any other windows.

Syntax

```
ShowHome();
```

Start a Windows application

In a script, you can start a Windows application using the StartApp command.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Syntax

```
StartApp appname;
```

Parameters

appname

Path and file name of the application you want to start. A literal string value, message tagname, or string expression.

Note: You need to know the path and file name of the application. If the application is in a directory that is part of the Windows PATH environment variable, you only need to pass the file name (without path).

Example(s)

This script starts Microsoft Calculator.

```
StartApp "calc"
```

SwitchDisplayLanguage() function

Switches the display of visible, static texts and alarm fields in a desired language for which translated strings are provided.

Category

misc

Syntax

```
SwitchDisplayLanguage(LocaleID);
```

Parameter

LocaleID

The language in which static text strings and alarm fields are to be shown at run time.

Example(s)

In this example, German is the language to be shown at run time.

```
SwitchDisplayLanguage(1031);
```

See also

$Language system tag

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

TseGetClientId() function

Returns a string version of the client ID (the TCP/IP address of the client) if the View application is running on a Terminal Server client. This client ID is used internally to generate SuiteLink server names and logger file names. Otherwise, the TseGetClientId() function returns an empty string.

Syntax

```
MessageResult=TseGetClientId();
```

Example

The client IP address 10.103.202.1 is saved to the MsgTag tag.

```
MsgTag=TseGetClientID();
```

TseGetClientNodeName() function

Returns the client node name if the View application is running on a Terminal Server client assigned a name that can be identified by Windows. Otherwise, the TseGetClientNodeName() function returns an empty string.

Syntax

```
MessageResult=TseGetClientNodeName();
```

Example

The client node name is returned as the value assigned to the MsgTag tag.

```
MsgTag=TseGetClientNodeName();
```

TseQueryRunningOnClient() function

Returns a non-zero integer value if the View application is running on a Terminal Services client. Otherwise, it returns a zero.

Syntax

```
Result=TseQueryRunningOnClient();
```

Return value

Returns 0 if View is not running on a Terminal Services client.

Example

IntTag is set to 1 if WindowViewer is running on a Terminal Services client.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

```
IntTag=TseQueryRunningOnClient;
```

TseQueryRunningOnConsole() function

The TseQueryRunningOnConsole() function can be run from a script to indicate whether the View application is running on a Terminal Services console.

Syntax

```
Result=TseQueryRunningOnConsole();
```

Return value

Returns a non-zero integer value if the View application is running on a Terminal Services console. Otherwise, the TseQueryRunningOnConsole() function returns a zero.

Example

IntTag is set to 1 if WindowViewer is running on a Terminal Services console.

```
IntTag=TseQueryRunningOnConsole();
```

Math functions

Use math functions to return the answer to the specified mathematical expression.

In QuickScript, all mathematical operations are calculated internally as double, regardless of the operand data type. Following standard mathematical rules, the result is always rounded in division operations to maintain accuracy. Rounding only occurs on the end result, not intermediate values, and the quotient will match the target data type. This is the standard methodology for SCADA and DCS systems, and provides the data integrity, precision retention, time stamps, and overall data quality propagation and aggregation needed for these systems.

If you want to round at each step instead of only at the final result, you can leverage the support built into QuickScript for .NET libraries and utilize the System.Math.Floor and System.Math.Round methods to explicitly round the intermediate steps. As an example, consider the following script:

```
dim dividend as integer;
dim divisor as integer;
dim quotient as integer;
dim remainder as integer;
dividend = 8;
divisor = 3;
LogMessage("Value of dividend = " + dividend);
LogMessage("Value of divisor = " + divisor);
quotient = dividend/divisor;
LogMessage("Value of quotient = " + quotient);
remainder = dividend mod divisor;
LogMessage ("Value of remainder = " + remainder);
dividend = divisor*quotient +remainder;
LogMessage ("Value of dividend = " + dividend);
```

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

The result is: 8 / 3 = 3

If, instead, you want to drop the remainder (not rounding the final result to the nearest integer), you could add a call to the Math.Floor method and use the following:

```
dim dividend as integer;
dim divisor as integer;
dim quotient as integer;
dim remainder as integer;
dividend = 8;
divisor = 3;
LogMessage("Value of dividend = " + dividend);
LogMessage("Value of divisor = " + divisor);
// *** Add call to Math.Floor. This drops the remainder rather than rounding the internal
Double result to integer
quotient = System.Math.Floor(dividend/divisor);

LogMessage("Value of quotient = " + quotient);
remainder = dividend mod divisor;
LogMessage ("Value of remainder = " + remainder);
dividend = divisor*quotient +remainder;
LogMessage ("Value of dividend = " + dividend);
```

The result is: 8 / 3 = 2 (remainder 2)

Abs()

Returns the absolute value (unsigned equivalent) of a specified number.

Category

Math

Syntax

```
Result = Abs( Number );
```

Parameter

Number

Any number or numeric attribute.

Examples

```
Abs(14); ' returns 14
Abs(-7.5); ' returns 7.5
```

ArcCos()

Returns an angle between 0 and 180 degrees whose cosine is equal to the number specified.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Category

Math

Syntax

```
Result = ArcCos( Number );
```

Parameter

Number

Any number or numeric attribute with a value between -1 and 1 (inclusive).

Examples

```
ArcCos(1); ' returns 0
ArcCos(-1); ' returns 180
```

See also

ArcSin()

ArcTan()

Cos()

Sin()

Tan()

ArcSin()

Returns an angle between -90 and 90 degrees whose sine is equal to the number specified.

Category

Math

Syntax

```
Result = ArcSin( Number );
```

Parameter

Number

Any number or numeric attribute with a value between -1 and 1 (inclusive).

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Examples

```
ArcSin(1); ' returns 90
ArcSin(-1); ' returns -90
```

See also

ArcCos()

ArcTan()

Cos()

Sin()

Tan()

ArcTan()

Returns an angle between -90 and 90 degrees whose tangent is equal to the number specified.

Category

Math

Syntax

```
Result = ArcTan( Number );
```

Parameter

Number

Any number or numeric attribute.

Examples

```
ArcTan(1); ' returns 45
ArcTan(0); ' returns 0
```

See also

ArcCos()

ArcSin()

Cos()

Sin()

Tan()

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Cos()

Returns the cosine of an angle in degrees.

Category

Math

Syntax

```
Result = Cos( Number );
```

Parameter

Number

Any number or numeric attribute.

Examples

```
Cos(90); ' returns 0
Cos(0); ' returns 1
```

This example shows how to use the function in a math equation:

```
Wave = 50 * Cos(6 * Now().Second);
```

See Also

ArcCos()

ArcSin()

ArcTan()

Sin()

Tan()

Exp()

Returns the result of the exponent *e* raised to a power.

Category

Math

Syntax

```
Result = Exp( Number );
```

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Parameter

Number

Any number or numeric attribute.

Example

```
Exp(1); ' returns 2.718...
```

Int()

Returns the next integer less than or equal to a specified number.

Category

Math

Syntax

```
IntegerResult = Int( Number );
```

Parameter

Number

Any number or numeric attribute.

Remarks

When handling negative real (float) numbers, this function returns the integer farthest from zero.

Examples

```
Int(4.7); ' returns 4
Int(-4.7); ' returns -5
```

Log()

Returns the natural log (base e) of a number.

Category

Math

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Syntax

```
RealResult = Log( Number );
```

Parameter

Number

Any number or numeric attribute.

Remarks

Natural log of 0 is undefined.

Examples

```
Log(100); ' returns 4.605...
Log(1); ' returns 0
```

See also

LogN()

Log10()

Log10()

Returns the base 10 log of a number.

Category

Math

Syntax

```
Result = Log10( Number );
```

Parameter

Number

Any number or numeric attribute.

Example

```
Log10(100); ' returns 2
```

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

See also

Log()
LogN()

LogN()

Returns the values of the logarithm of x to base n.

Category

Math

Syntax

```
Result = LogN( Number, Base );
```

Parameters

Number

Any number or numeric attribute.

Base

Integer to set log base. You could also specify an integer attribute.

Remarks

Base 1 is undefined.

Examples

```
LogN(8, 3); ' returns 1.89279
LogN(3, 7); ' returns 0.564
```

See also

Log()
Log10()

Pi()

Returns the value of Pi.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Category

Math

Syntax

```
RealResult = Pi();
```

Example

```
Pi(); ' returns 3.1415926
```

Round()

Rounds a real number to a specified precision and returns the result.

Category

Math

Syntax

```
RealResult = Round( Number, Precision );
```

Parameters

Number

Any number or numeric attribute.

Precision

Sets the precision to which the number is rounded. This value can be any number or a numeric attribute.

Examples

```
Round(4.3, 1); ' returns 4
Round(4.3, .01); ' returns 4.30
Round(4.5, 1); ' returns 5
Round(-4.5, 1); ' returns -4
Round(106, 5); ' returns 105
Round(43.7, .5); ' returns 43.5
```

See also

Trunc()

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Sgn()

Determines the sign of a value (whether it is positive, zero, or negative) and returns the result.

Category

Math

Syntax

```
IntegerResult = Sgn( Number );
```

Parameter

Number

Any number or numeric attribute.

Return value

If the input number is positive, the result is 1. Negative numbers return a -1, and 0 returns a 0.

Examples

```
Sgn(425); ' returns 1;
Sgn(0); ' returns 0;
Sgn(-37.3); ' returns -1;
```

Sin()

Returns the sine of an angle in degrees.

Category

Math

Syntax

```
Result = Sin( Number );
```

Parameter

Number

Angle in degrees. Any number or numeric attribute.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Examples

```
Sin(90); ' returns 1;
Sin(0); ' returns 0;
```

This example shows how to use the function in a math expression:

```
wave = 100 * Sin (6 * Now().Second);
```

See also

ArcCos()

ArcSin()

ArcTan()

Cos()

Tan()

Sqrt()

Returns the square root of a number.

Category

Math

Syntax

```
RealResult = Sqrt( Number );
```

Parameter

Number

Any number or numeric attribute.

Example

This example takes the value of me.PV and returns the square root as the value of x:

```
x=Sqrt(me.PV);
```

Tan()

Returns the tangent of an angle given in degrees.

Category

Math

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Syntax

```
Result = Tan( Number );
```

Parameter

Number

The angle in degrees. Any number or numeric attribute.

Examples

```
Tan(45); ' returns 1;
Tan(0); ' returns 0;
```

This example shows how to use the function in a math expression:

```
Wave = 10 + 50 * Tan(6 * Now().Second);
```

See also

ArcCos()

ArcSin()

ArcTan()

Cos()

Sin()

Trunc()

Truncates a real (floating point) number by simply eliminating the portion to the right of the decimal point, including the decimal point, and returns the result.

Category

Math

Syntax

```
NumericResult = Trunc( Number );
```

Parameter

Number

Any number or numeric attribute.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Remarks

This function accomplishes the same result as placing the contents of a float type attribute into an integer type attribute.

Examples

```
Trunc(4.3); ' returns 4;
Trunc(-4.3); ' returns -4;
```

See also

Round()

Miscellaneous functions

Functions in the miscellaneous group perform a variety of purposes, such as logging data or querying attributes.

ActivateApp()

Restores, minimizes, maximizes, or closes another currently running Windows application.

Category

Miscellaneous

Syntax

```
ActivateApp( TaskName );
```

Parameter

TaskName

The task this function activates.

Remarks

TaskName is the exact text string, including spaces, that appears on the Task Bar or in Windows Task Manager. You can see the task name by opening Task Manager.

Example

```
ActivateApp("Calculator");
```

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Filtering events

To get only specific events, filters can be introduced before getting events from the event service. The filtering should be done before the StartRequestingEvent() method is called.

The following datatypes are supported when filtering the events.

- Integer
- Float
- String
- Bool
- DateTime
- Double
- Short
- Array

The following table shows the comparison types that are supported for filtering events.

Comparison keyword	Description
eq	Means EqualTo. Returns all the events matching the filtered criteria.
beginswith	Means StartsWith. Returns all the events matching the filtered criteria. Applies only to string data type filtering
lt	Means Lesser Than. Applies to all supported data types excluding string. It does not support arrays.
le	Means Lesser or Equal. Applies to all supported data types excluding string. It does not support arrays.
gt	Means Greater Than. Applies to all supported data types excluding string. It does not support arrays.
ge	Means Greater or Equal. Applies to all supported data types excluding string. It does not support arrays.
between	Checks will be made only to paired supplied values. Returns all the events matching the filtered criteria. It supports numeric and date data types.
neg, nbegins, nlt, nle, ngt, nge, nbetween	A keyword 'n' before the comparison keyword Means NOT of.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

DateTimeGMT()

Returns a number representing the number of days and fractions of days since January 1, 1970, in Coordinated Universal Time (UTC), regardless of the local time zone.

Category

Miscellaneous

Syntax

```
Result=DateTimeGMT();
```

Parameters

```
None
```

Example

```
MessageTag = StringFromTime(DateTimeGMT() * 86400.0, 3);
```

IsBad()

Returns a Boolean value indicating if the quality of the specified attribute is Bad.

Category

Miscellaneous

Syntax

```
BooleanResult = IsBad( Attribute1, Attribute2, … );
```

Parameter(s)

Attribute1, Attribute2, ...AttributeN

Names of one or more attributes for which you want to determine Bad quality. You can include a variable-length list of attributes.

Return value

If any of the specified attributes has Bad quality, then true is returned. Otherwise, false is returned.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Examples

```
IsBad(TIC101.PV);
IsBad(TIC101.PV, PIC102.PV);
```

See also

IsGood()

IsInitializing()

IsUncertain()

IsUsable()

IsGood()

Returns a Boolean value indicating if the quality of the specified attribute is Good.

Category

Miscellaneous

Syntax

```
BooleanResult = IsGood( Attribute1, Attribute2, … );
```

Parameter(s)

Attribute1, Attribute2, and so on

Name of the attribute(s) for which you want to determine Good quality. You can include a variable-length list of attributes.

Return Value

If all of the specified attributes have Good quality, then true is returned. Otherwise, false is returned.

Examples

```
IsGood(TIC101.PV);
IsGood(TIC101.PV, PIC102.PV);
```

See Also

IsBad(), IsInitializing(), IsUncertain(), IsUsable()

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

IsInitializing()

Returns a Boolean value indicating if the quality of the specified attribute is Initializing.

Category

Miscellaneous

Syntax

```
BooleanResult = IsInitializing( Attribute1, Attribute2, … );
```

Parameter(s)

Attribute1, Attribute2, and so on

Name of the attribute(s) for which to determine Initializing quality. You can include a variable-length list of attributes.

Return Value

If any of the specified attributes has Initializing quality, then true is returned. Otherwise, false is returned.

Examples

```
IsInitializing(TIC101.PV);
IsInitializing(TIC101.PV, PIC102.PV);
```

See Also

IsBad(), IsGood(), IsUncertain(), IsUsable()

IsUncertain()

Returns a Boolean value indicating if the quality of the specified attribute is Uncertain.

Category

Miscellaneous

Syntax

```
BooleanResult = IsUncertain( Attribute1, Attribute2, … );
```

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Parameter(s)

Attribute1, Attribute2, and so on

Name of the attribute(s) to determine Uncertain quality. You can include a variable-length list of attributes.

Return Value

If all of the specified attributes have Uncertain quality, then true is returned. Otherwise, false is returned.

Examples

```
IsUncertain(TIC101.PV);
IsUncertain(TIC101.PV, PIC102.PV);
```

See also

IsBad()

IsGood()

IsInitializing()

IsUsable()

IsUsable()

Returns a Boolean value indicating if the specified attribute is usable for calculations.

Category

Miscellaneous

Syntax

```
BooleanResult = IsUsable( Attribute1, Attribute2, … );
```

Parameter(s)

Attribute1, Attribute2, ...AttributeN

Name of one or more attributes for which you want to determine unusable quality. You can include a variable-length list of attributes.

Return value

If all of the specified attributes have either Good or Uncertain quality, then true is returned. Otherwise, false is returned.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Remarks

The attributes having Good or Uncertain quality qualifies as usable. In addition, each float or double attribute cannot be a NaN (not a number).

Examples

```
IsUsable(TIC101.PV);
IsUsable(TIC101.PV, PIC102.PV);
```

See also

IsBad()

IsGood()

IsInitializing()

IsUncertain()

LogCustom()

Writes a user-defined custom flag message in the Log Viewer.

Category

Miscellaneous

Syntax

```
LogCustom( CustomFlag, msg );
```

Parameter

CustomFlag

Creates a new log flag based on the first parameter string. The first call creates the custom flag.

msg

The message to write to the Log Viewer. Actual string or a string attribute.

Remarks

The log flag is disabled by default.

The message is always logged under the component "ObjectName.ScriptName". For example, "WinPlatform_001.script1: msg", which identifies what object and what script within the object logged the error.

LogCustom() is similar to LogMessage(), but displays the message in the custom log flag when Log Custom is enabled.

The parameter help tooltip and Function Browser sample parameter list will show "LogCustom(CustomFlag, msg

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

)" rather than "LogCustom(CustomFlag, Message)". "Message" is a reserved keyword.

Example

```
LogCustom(EditBox1.text, "User-defined message.";
```

This statement writes to the Log Viewer as follows:

```
10/24/2005 12:49:14 PM ScriptRuntime
<ObjectName.ScriptName>: <LogFlag EditBox1> User-defined message.
```

LogDataChangeEvent()

Logs an application change event to the application Historian.

The LogDataChangeEvent() function works only in object scripts, not in graphic scripts.

Category

Miscellaneous

Syntax

```
LogDataChangeEvent(AttributeName, Description, OldValue, NewValue, TimeStamp);
```

Parameters

AttributeName

Attribute name as a tag name.

Description

Description of the object.

OldValue

Old value of the attribute.

NewValue

New value of the attribute.

TimeStamp

The time stamp associated with the logged event. The timestamp can be UTC or local time. The TimeStamp parameter is optional. The timestamp of the logged event defaults to Now() if a TimeStamp parameter is not included.

Remarks

A graphic script still compiles if the LogDataChangeEvent() function is included. However, a warning message is written to the log at run time that the function is inoperable.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Example

This example logs an event when a pump starts or stops with a timestamp of the current time when the event occurred.

```
LogDataChangeEvent(TC104.pumpstate, "Pump04", OldState, NewState);
```

LogError()

Writes a user-defined error message in the Log Viewer with a red error log flag.

Category

Miscellaneous

Syntax

```
LogError( msg );
```

Parameter

msg

The message to write to the Log Viewer. Actual string or a string attribute.

Remarks

The log flag is enabled by default.

The message is always logged under the component "ObjectName.ScriptName". For example, "WinPlatform_001.script1: msg", which identifies what object and what script within the object logged the error.

LogError() is similar to LogMessage(), but displays the message in red.

The parameter help tooltip and Function Browser sample parameter list will show "LogError(msg)" rather than "LogError(Message)". "Message" is a reserved keyword.

Example

```
LogError("User-defined error message.");
```

This statement writes to the Log Viewer as follows:

```
10/24/2005 12:49:14 PM ScriptRuntime
<ObjectName.ScriptName>: User-defined error message.
```

LogMessage()

Writes a user-defined message to the Log Viewer.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Category

Miscellaneous

Syntax

```
LogMessage( msg );
```

Parameter

msg

The message to write to the Log Viewer. Actual string or a string attribute.

Remarks

This is a very powerful function for troubleshooting scripting. By strategically placing LogMessage() functions in your scripts, you can determine the order of script execution, performance of scripts, and identify the value of attributes both before they are changed and after they are affected by the script.

Each message posted to the Log Viewer is stamped with the exact date and time. The message always begins with the component "Tagname.ScriptName" so you can tell what object and what script within the object posted the message to the log.

Examples

```
LogMessage("Report Script is Running");
```

The above statement writes the following to the Log Viewer:

```
10/24/2005 12:49:14 PM ScriptRuntime <Tagname.ScriptName>:Report Script is Running.
MyTag=MyTag + 10;
LogMessage("The Value of MyTag is " + Text(MyTag, "#"));
```

LogTrace()

Writes a user-defined trace message in the Log Viewer.

Category

Miscellaneous

Syntax

```
LogTrace( msg );
```

Parameter

msg

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

The message to write to the Log Viewer. Actual string or a string attribute.

Remarks

The log flag is disabled by default.

The message is always logged under the component "ObjectName.ScriptName". For example, "WinPlatform_001.script1: msg", which identifies what object and what script within the object logged the error.

LogTrace() is similar to LogMessage(), but displays the message as Trace when Log Trace is enabled.

The parameter help tooltip and Function Browser sample parameter list will show "LogTrace(msg)" rather than "LogTrace(Message)". "Message" is a reserved keyword.

Example

```
LogTrace("User-defined trace message.");
```

This statement writes to the Log Viewer as follows:

```
10/24/2005 12:49:14 PM ScriptRuntime
<ObjectName.ScriptName>: User-defined trace message.
```

LogWarning()

Writes a user-defined error message in the Log Viewer with a yellow warning log flag.

Category

Miscellaneous

Syntax

```
LogWarning( msg );
```

Parameter

msg

The message to write to the Log Viewer. Actual string or a string attribute.

Remarks

The log flag is disabled by default.

The message is always logged under the component "ObjectName.ScriptName". For example, "WinPlatform_001.script1: msg", which identifies what object and what script within the object logged the error.

LogWarning() is similar to LogMessage(), but displays the message as a yellow warning message.

The parameter help tooltip and Function Browser sample parameter list will show "LogWarning(msg)" rather than "LogWarning(Message)". "Message" is a reserved keyword.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Example

```
LogWarning("User-defined warning message.")
```

This statement writes to the Log Viewer as follows:

```
10/24/2005 12:49:14 PM ScriptRuntime
<ObjectName.ScriptName>: User-defined warning message.
```

SendKeys()

Sends keystrokes to an application. To the receiving application, the keys appear to be entered from the keyboard. You can use SendKeys() within a script to enter data or send commands to an application. Most keyboard keys can be used in a `SendKeys()` statement. Each key is represented by one or more characters, such as A for the letter A or {ENTER} for the Enter key.

Category

Miscellaneous

Syntax

```
SendKeys( KeySequence );
```

Parameter

KeySequence

Any key sequence or a string attribute.

Remarks

To specify more than one key, concatenate the codes for each character. For example, to specify the dollar sign ($) key followed by a (b), enter $b.

The following lists the valid send key codes for unique keyboard keys:

Key	Code
BACKSPACE	{BACKSPACE}or {BS}
BREAK	{BREAK}
CAPSLOCK	{CAPSLOCK}
DELETE	{DELETE} or {DEL}
DOWN	{DOWN}
END	{END}

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Key	Code
ENTER	{ENTER} or tilde (~)
ESCAPE	{ESCAPE} or {ESC}
F1...F12	{F1}...{F12}
HOME	{HOME}
INSERT	{INSERT}
LEFT	{LEFT}
NUMLOCK	{NUMLOCK}
PAGE DOWN	{PGDN}
PAGE UP	{PGUP}
PRTSC	{PRTSC}
RIGHT	{RIGHT}
TAB	{TAB}
UP	{UP}
HOME	{HOME}

Special keys (SHIFT, CTRL, and ALT) have their own key codes:

Key	Code
SHIFT	+ (plus)
CTRL	^ (caret)
ALT	% (percent)

Enhancements to the Microsoft Hardware Abstraction Layer in Windows prevents the SendKeys() function from operating on some computers.

Examples

To use two special keys together, use a second set of parentheses. The following statement holds down the CTRL key while pressing the ALT key, followed by p:

```
SendKeys ("^(%(p))");
```

Commands can be preceded by the `ActivateApp()` command to direct the keystrokes to the proper application.

The following statement gives the computer focus to Calculator and sends the key combination 1234:

```
ActivateApp("Calculator");
```

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

```
SendKeys("^(1234)");
```

SetAttributeVT()

Sets the value and timestamp of an object attribute. For buffered values, only the last calculated value is captured for historization.

Category

Miscellaneous

Syntax

```
SetAttributeVT( Attribute, Value, TimeStamp);
```

Parameter

Attribute

Name of the object attribute whose value and timestamp are modified. The specified attribute must belong to the object to which the script is attached.

Value

Value of the attribute, which can be a reference. The quality is always set to Good.

TimeStamp

Timestamp that can be a reference, a variable, or a string interpreted as the computer's local time or UTC. The timestamp is converted internally to UTC format before the attribute's value is sent to the run-time component.

Remarks

Interim calculated buffered values are NOT historized. Use SetAttributeVT2() if historization of interim values is needed.

Timestamp can be set only for object attributes that support a timestamp. At compile time, the script cannot detect whether the attribute specified with the SetAttributeVT() function supports a timestamp or not. No warning is issued if the attribute does not support a timestamp.

Example

This example sets an integer value and timestamp for an attribute that indicates pump RPM.

```
SetAttributeVT(me.PV, TC104.PumpRPM, LCLTIME);
```

SetAttributeVT2()

Sets the value and timestamp of an object attribute. This function is identical to SetAttributeVT(), but SetAttributeVT2() allows interim calculated data for buffered values to be historized once per scan cycle.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Category

Miscellaneous

Syntax

```
SetAttributeVT2( Attribute, Value, TimeStamp);
```

Parameter

Attribute

Name of the object attribute whose value and timestamp are modified. The specified attribute must belong to the object to which the script is attached.

Value

Value of the attribute, which can be a reference. The quality is always set to Good.

TimeStamp

Timestamp that can be a reference, a variable, or a string interpreted as the computer's local time or UTC. The timestamp is converted internally to UTC format before the attribute's value is sent to the run-time component.

Remarks

In contrast to SetAttributeVT(), SetAttributeVT2() allows historization of interim calculated buffered values.

Timestamp can be set only for object attributes that support a timestamp. At compile time, the script cannot detect whether the attribute specified with the SetAttributeVT2() function supports a timestamp or not. No warning is issued if the attribute does not support a timestamp.

Example

This example sets an integer value and timestamp for an attribute that indicates pump RPM (interim calculated values for buffered data are historized).

```
SetAttributeVT2(me.PV, TC104.PumpRPM, LCLTIME);
```

SetBad()

Sets the quality of an attribute to Bad.

Category

Miscellaneous

Syntax

```
SetBad( Attribute );
```

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Parameter

Attribute
The attribute for which you want to set the quality to Bad.

Remarks

The specified attribute needs be within the object to which the script is attached.

Example

```
SetBad(me.PV);
```

See also

SetGood()
SetInitializing()
SetUncertain()

SetGood()

Sets the quality of an attribute to Good.

Category

Miscellaneous

Syntax

```
SetGood( Attribute );
```

Parameter

Attribute
The attribute for which you want to set the quality to Good.

Remarks

The specified attribute needs to be within the object to which the script is attached.

Example

```
SetGood(me.PV);
```

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

See also

SetBad()
SetInitializing()
SetUncertain()

SetInitializing()

Sets the quality of an attribute to Initializing.

Category

Miscellaneous

Syntax

```
SetInitializing( Attribute );
```

Parameter

Attribute
The attribute for which you want to set the quality to Initializing.

Remarks

The specified attribute needs to be within the object to which the script is attached.

Example

```
SetInitializing(me.PV);
```

See also

SetBad()
SetGood()
SetUncertain()

SetUncertain()

Sets the quality of an attribute to Uncertain.

Category

Miscellaneous

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Syntax

```
SetUncertain( Attribute );
```

Parameter

Attribute

The attribute for which you want to set the quality to Uncertain.

Remarks

The specified attribute needs to be within the object to which the script is attached.

Example

```
SetUncertain(me.PV);
```

See also

SetBad()

SetGood()

SetInitializing()

SignedAckAll()

SignedAckAll is a script function for ArchestrA graphics to perform an acknowledgment of all the alarms on ArchestrA attributes or IADAS references within a graphic - optionally requiring a signature depending on whether any of the indicated alarms is currently waiting for an ACK and falls within a designated priority range. If so, a user must perform a log-in operation to acknowledge the alarms. This function returns an integer status indicating success or failure:

- zero if the function succeeds
- non-zero if the function fails or operation is canceled by the user

SignedAckAll has the similar design-time and runtime behaviors as SignedAlarmAck script function except the following:

- Automatically detects all the ArchestrA alarms and IADAS alarms within a graphic that are waiting for an ACK
- ACK both ArchestrA alarms and IADAS alarms

Note: AVEVA OMI ViewApps do not support the SignedAckAll method. Web Client does not support the SignedAckAll method.

Category

Miscellaneous

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Syntax

```
int SignedAckAll(String Graphic_Instance_Name,
Boolean Include_Parent_Symbol,
Boolean Signature_Reqd_for_Range,
Integer Min_Priority,
Integer Max_Priority,
String Default_Ack_Comment,
Boolean Ack_Comment_Is_Editable,
String TitleBar_Caption,
String Message_Caption
);
```

Parameters

Graphic_Instance_Name

- The Graphic_Instance_Name is the name of the embedded Visualization folder graphic or object graphic. For example, Graphic_Instance_Name = cp1, cp1 is a string type of custom property, cp1 = "symbol_0011" or cp1 = "Pump001_s11".
- The Graphic_Instance_Name also can be the name of a specific graphic element inside a graphic. For example, "symbol_0011.Text1".
- If Graphic_Instance_Name is empty "", this script function will ack all the alarms of the graphic where this script function resides.

  Data Type

  String

  Valid Range

  Limit 1024 characters

  Additional Information

  Can be a constant string, a reference, or an expression.

Include_Parent_Symbol

Indicates whether all the applicable alarms on all level parent graphics will be acknowledged when Graphic_Instance_Name is empty. If Graphic_Instance_Name is empty and Include_Parent_Symbol is true, all the applicable alarms on all level parent graphics of the graphic owning graphic will be acknowledged. If Graphic_Instance_Name is empty and Include_Parent_Symbol is false, only the applicable alarms on the graphic owning graphic will be acknowledged. If Graphic_Instance_Name is not empty, Include_Parent_Symbol option will be ignored.

Data Type

Bool

Additional Information

Can be a constant, a reference, or an expression.

Signature_Reqd_for_Range

Indicates whether a signature is required for acknowledging alarms.

Data Type

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Bool

Additional Information

Can be a constant, a reference, or an expression.

Min_Priority

Represents the minimum priority value of the range for which the signature is required.

Data Type

Integer

Valid Range

1-999; must be less than or equal to the Max_Priority value.

Additional Information

Can be a constant, a reference, or an expression.

Max_Priority

Represents the maximum priority value of the range for which the signature is required.

Data Type

Integer

Valid Range

1-999; must be greater than or equal to the Min_Priority value.

Additional Information

Can be a constant, a reference, or an expression.

Default_Ack_Comment

Comment to be shown in the Acknowledge Alarms dialog box.

Data Type

String

Valid Range

Limit 200 characters

Additional Information

Can be a constant, a reference or an expression. If the parameter is empty, then no default comment is shown in the Acknowledge Alarms dialog box.

Ack_Comment_Is_ Editable

Indicates whether the run-time user can modify the acknowledgement comment.

Data Type

Bool

Additional Information

Can be a constant, a reference, or an expression. If set to False, the Comment box in the Acknowledge Alarms dialog box is unavailable.

TitleBar_Caption

Shows a title in the title bar of the Acknowledge Alarms dialog box.

Data Type

String

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Valid Range

Limit 1024 characters

Additional Information

Can be a constant, a reference, or an expression. If the TitleBar_Caption is empty, the default title, Acknowledge Alarms, is shown.

Message_Caption

Shows a customizable message to the run-time user in the Acknowledge Alarms dialog box.

Data Type

String

Valid Range

Limit 250 characters

Additional Information

Can be a constant, a reference, or an expression. Use the parameter to provide more information on the alarm to the run-time user. This message is not propagated to the event record.

Return values

Return Value	Data type	Description
status	Integer	The status values are as follows: **0 :** The function returns a value of 0 (meaning success) if the following are all true: • The function parameters were valid • At least one alarm was waiting for an ACK • The user credentials were valid or no credentials were needed **Note:** User permissions are not validated. Thus, a user may have valid credentials, but not have the necessary permissions to acknowledge alarms. • The user did not cancel the operation • The function wrote to the .AckMsg Attributes or the .UNACK IADAS references of the indicated alarms that currently were waiting for an ACK **Note:** A return value of 0 does

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

	not indicate that the alarms are acknowledged, only that the function wrote to the AckMsg attributes. The alarms may not be acknowledged if the validated user had insufficient permissions, or if the alarms were already acknowledged.
	-2 ,-1, 1 : See the description of the return status in SignedAckAll script function. Return status -1 is applicable only if user canceled the login dialog or ack comment edit dialog when Signature_Reqd_for_Range is true or Ack_Comment_Is_editable is true.
	2 : If the Graphic_Instance_Name was not valid at runtime
	3: If the Min_Priority or Max_Priority is out of the range 1..999, the function writes a message to the Logger identifying which parameter(s) was out of range and displaying the actual value(s).
	4: If the Min_Priority > Max_Priority, the function writes a message to the Logger identifying the problem and displaying the actual values.

Example

```
n = SignedAckAll ("",false,false,1,999,"Ack all by script",false,"","")
```

SignedAlarmAck()

Acknowledges one or more alarms on tags or attributes, optionally requiring a signature if any of the indicated alarms falls within a designated priority range.

This function is supported only for client scripting and not object scripting.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Category

Miscellaneous

Syntax

```
int SignedAlarmAck(String Alarm_List,
Boolean Signature_Reqd_for_Range,
Integer Min_Priority,
Integer Max_Priority,
String Default_Ack_Comment,
Boolean Ack_Comment_Is_Editable,
String TitleBar_Caption,
String Message_Caption
);
```

Parameters

Alarm_List

The list of alarms to be acknowledged. The list must be a single text string with each alarm name separated by a space or a comma.

Data Type

String

Valid Range

Limit 1024 characters

Additional Information

Can be a constant string, a reference, or an expression.
Only alarms on tags or attributes are supported.
If there is any invalid alarm in the list, then none of the alarms are acknowledged.

Examples

Example 1:

```
"UD1.analog_001.HiHi"
```

The collection is represented as a text string, with alarms separated by blanks and/or commas.

Example 2:

```
"UD1.analog_001.HiHi UD9.x14.dev.major"
```

Example 3:

```
"UD1.analog_001.HiHi, UD9.x14.dev.major"
```

Example 4, an array of strings such as:

```
Pump1.AlarmArray[1] = "Pump1.Level.HiHi"
Pump1.AlarmArray[2] = "Pump1.Level.LoLo"
```

uses the function as follows:

```
SignedAlarmAck(Pump1.AlarmArray[ ], ...)
```

The script passes to the function the following single string:

```
"Pump1.Level.HiHi, Pump1.Level.LoLo"
```

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Signature_Reqd_for_Range

Indicates whether a signature is required for acknowledging alarms.

Data Type

Bool

Additional Information

Can be a constant, a reference, or an expression.

Min_Priority

Represents the minimum priority value of the range for which the signature is required.

Data Type

Integer

Valid Range

1-999; must be less than or equal to the Max_Priority value.

Additional Information

Can be a constant, a reference, or an expression.

Max_Priority

Represents the maximum priority value of the range for which the signature is required.

Data Type

Integer

Valid Range

1-999; must be greater than or equal to the Min_Priority value.

Additional Information

Can be a constant, a reference, or an expression.

Default_Ack_Comment

Comment to be shown in the **Acknowledge Alarms** dialog box.

Data Type

String

Valid Range

Limit 200 characters

Additional Information

Can be a constant, a reference or an expression.
If the parameter is empty, then no default comment is shown in the **Acknowledge Alarms** dialog box.

Ack_Comment_Is_ Editable

Indicates whether the run-time user can modify the acknowledgement comment.

Data Type

Bool

Additional Information

Can be a constant, a reference, or an expression.
If set to False, the **Comment** box in the **Acknowledge Alarms** dialog box is unavailable.

TitleBar_Caption

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Shows a title in the title bar of the **Acknowledge Alarms** dialog box.

Data Type

String

Valid Range

Limi 1024 characters

Additional Information

Can be a constant, a reference, or an expression.

If the TitleBar_Caption is empty, the default title, Acknowledge Alarms, is shown.

Message_Caption

Shows a customizable message to the run-time user in the **Acknowledge Alarms** dialog box.

Data Type

String

Valid Range

Limit 250 characters

Additional Information

Can be a constant, a reference, or an expression.
Use the parameter to provide more information on the alarm to the run-time user.
This message is not propagated to the event record.

Return values

Return values indicate success or failure status. A non-zero value indicates type of failure.

-1	The user canceled the operation. The function writes a message to the Logger indicating user cancellation.
-2	No alarms are waiting for acknowledgement.
0	The function is successful and the following are all true: • The function parameters are valid. • The user credentials are valid (or no credentials are needed). • The user did not cancel the operation. • Function wrote to the .AckMsg attributes of the indicated alarms.
1	The function failed due to any error that is not covered by the other specified return values.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

2	One or more parameters were not coerced to the appropriate data type at run time. Example: Parameter is a reference with Boolean as the expected data type. At run time, reference is to a String data type that cannot be coerced to True or False. The function writes a message to the Logger.
3	The Alarm_List parameter was not valid at run time. • String was null, empty or contained no attribute references. • Contained one or more items that were not valid attribute references. • Contained one or more attribute references that did not exist or did not identify valid alarm primitives. If Alarm_List contains a mixture of valid and invalid references, the function does nothing. The function does not attempt to operate on the valid references, and returns this error status.
4	The Min_Priority or MaxPriority values do not fall within the range of 1 to 999. The function writes a message to the Logger indicating which parameter was out of range and showing the actual value.
5	The Min_Priority value is greater than the Max_Priority value. The function writes a message to the Logger identifying the problem and showing the actual values.

Note: A return value of zero does not indicate if the alarms are acknowledged, only that the function wrote to the AckMsg attributes. The alarms may not be acknowledged due to insufficient permission or if the alarms have already been acknowledged.

Remarks

For more information about using the SignedAlarmAck() function, see the topic *Signature Security for Acknowledging Alarms,* under "Adding and Maintaining Graphic Scripts" in the *Creating and Managing Industrial Graphics User Guide*.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Examples

```
Dim n as Integer;
n = SignedAlarmAck("UD1.analog_001.HiHi UD9.x14.dev.major", true, 1, 250, "Acknowledged
by script", true, "Acking Tank Alarms", "Acknowledge the tank alarms");
```

Using an array of strings:

```
dim arr[2] as String;
arr[1] = "UD1.analog_001.HiHi";
arr[2] = "UD9.x14.dev.major";
n = SignedAlarmAck(arr[], true, 200, 500, "Acked by script", true, "Acking Tank Alarms",
"Please acknowledge the tank alarms.");
```

SignedWrite()

Performs a write to an AutomationObject attribute that has a Secured Write or Verified Write security classification.

Category

Miscellaneous

Syntax

```
int SignedWrite(string Attribute,
object Value,
string ReasonDescription,
Bool Comment_Is_Editable,
Enum Comment_Enforcement,
string[] Predefined_Comment_List
);
```

Brackets [ ] indicate an array.

Parameters

Attribute	*This is the name of the attribute to be updated and is a string data type.* *Can be a constant string, a reference, or an expression.* *Supports bound and nested bound references.* *For detailed examples of Attribute parameter uses, see the topic Examples of Using the Attribute Parameter in the SignedWrite() Function under "Managing Graphics" in the Creating and Managing Industrial Graphics User Guide.* *Example 1:* `"UserDefined_001.temp"` *Example 2:*

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

	`"Pump15" + ".valve4"`
	Example 3:
	With UDO_7 containing two string attributes, namestrA and namestrB set to the values "Tank1" and "Tank5" respectively, the following script writes to Tank1.Level or Tank5.Level according to whether strselect is "A" or "B":
	`Dim strselect As String;`
	`Dim x As Indirect;`
	`{ logic to set strselect to "A" or "B" }`
	`x.BindTo ("UDO_7.namestr" + strselect);`
	`SignedWrite(x + ".Level", 243, "Set " + x + " Level", true, 0, null);`

Value	*This is the value to be written. Data type is object. The valid range for value must match the data type of the attribute being updated.* *Value can be a constant value, a reference, an expression, or NULL if nothing is to be entered.*
ReasonDescription	*This is text that explains the purpose of the target attribute and the impact of changing it.* *Data type is string with a maximum length is 256 characters. It can be a constant string, a reference, or an expression.* *ReasonDescription is passed to the indicated Attribute as part of the write operation. The object also includes the user's write comment, if any. A Field Attribute description is used for the ReasonDescription parameter only if the attribute is a Field Attribute and it has a description (is not null). Otherwise, the Short Description for the corresponding ApplicationObject is used for the ReasonDescription parameter.*
Comment_is _Editable	*This parameter indicates whether a user can edit the write comment.* *Data type is Boolean.* *If set to **True**: The comment text box is enabled with exceptions.* • *If Comment_Is_Editable is true and if the Comment_ Enforcement parameter is PredefinedOnly, the comment text box is disabled. At run time, the user can only select a comment from the predefined comment list.* • *If the Comment_ Enforcement parameter is not PredefinedOnly, the comment list and box are*

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

	enabled. You can select a comment from the comment list and modify it in the comment box. • *If the predefined list is empty, the comment list is not shown in the dialog box.* *If set to **False**: The predefined comment list does not appear in the Secured Write or Verified Write dialog boxes. The editable comment text box is disabled.*
Comment_Enforcement	*This parameter is of data type Enum, with the following value options:* *0 = Optional* *The run-time user can enter a comment or leave it blank.* *1 = Mandatory* *The run-time user must add a comment, either by selecting from the comment list or by entering a comment in the comment box.* *2 = PredefinedOnly* *The run-time user can select a comment from the comment list only. The comment text box is disabled.*
Predefined_Comment_List	*An array of strings that can be used as predefined comments.* *Data type is string. A maximum of 20 comments are allowed, each with a maximum of 200 characters.* • *The array can be empty (number of elements is 0).* • *Can be a constant, a reference, an expression, or NULL if empty. Can reference an attribute that contains an array of strings.* • *If no predefined comment is entered, the predefined comment list is disabled at run time.* • *If Comment_Is_Editable is False, the predefined comment is still placed in the editable comment text box, but the user cannot modify it at run time.*

Return values

Return values indicate success or failure status. A non-zero value indicates type of failure.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

0	The function returns a value of 0 (meaning success) if the following are all true: • The function parameters were valid. • The write operation was successfully placed on the queue for Secured and Verified Writes. • If the user cancels the operation, a message is written to the Logger indicating user cancellation.
1	The function failed due to any error that is not covered by the other specified return values. This includes any error that is not covered by the other specified return values. If there is a failure, a specific message is logged in the Logger.
2	One or more parameters were not coerced to the appropriate data type at run time. Example: Parameter is a reference with Boolean as the expected data type. At run time, reference is to a String data type that cannot be coerced to True or False. The function returns this value and writes a message to the Logger.
3	The attribute parameter was not valid at run time. • Attribute string was null, empty, or contained no attribute reference. • Attribute string contained an item that was not a valid attribute reference. • Attribute string contained an attribute reference that did not exist. • Attribute string contained an attribute reference that was not of the Secured Write or Verified Write security classification. The function writes a message to the Logger identifying the error and the invalid attribute string.
4	The Comment_Enforcement parameter value was out of the range of valid enumerators.

Remarks

The SignedWrite() function is supported only for client scripting and not for object scripting.

A return value of 0 does not indicate whether the attribute was updated, only that the function placed an entry

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

on the queue to write to the attribute. The operator may decide to cancel the operation after the Secured Write or Verified write dialog box is presented. In this case the attribute is not updated and a message is placed in the Logger indicating that the user canceled the operation. Even if the user enters valid credentials and clicks **OK**, the attribute still might not have been updated because of inadequate permission or data coercion problems.

The SignedWrite() function supports the custom property passed as the first parameter with opened and closed quotation marks, "".

If you configure the custom property CP as shown in the following script, the function attempts to resolve CP and determine if it has a reference. If it has a reference, then the reference is retrieved and the write is performed on the reference.

```
SignedWrite("CP", value, reason, editable, enforcement, null);
```

For more information about using the SignedWrite() function, see the topic *Working with the SignedWrite() Function for Secured and Verified Writes* under "Managing Graphics" in the *Creating and Managing Industrial Graphics User Guide.*

Examples

```
SignedWrite ("UserDefined_001.temp", 185, "This will change the oven temperature", true, 1,
null);
```

The following example shows the user an array of predefined comments:

```
Dim n as Integer;
n = SignedWrite("UserDefined_001.temp", 185, "This will change the oven temperature", true,
1, UserDefined_001.OvenCommentArray[ ]);
```

where UserDefined_001.OvenCommentArray is an attribute containing an array of strings.

WriteStatus()

Returns the enumerated write status of the last write to the specified attribute.

Category

Miscellaneous

Syntax

```
Result = WriteStatus( Attribute );
```

Parameter

Attribute

The attribute for which you want to return write status.

Return value

The return statuses are:

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

- MxStatusOk
- MxStatusPending
- MxStatusWarning
- MxStatusCommunicationError
- MxStatusConfigurationError
- MxStatusOperationalError
- MxStatusSecurityError
- MxStatusSoftwareError
- MxStatusOtherError

Remarks

If the attribute has never been written to, this function returns MxStatusOk. This function always returns MxStatusOk for attributes that do not support a calculated (non-Good) quality.

Example

```
WriteStatus(TIC101.SP);
```

WWControl()

Restores, minimizes, maximizes, or closes an application.

Category

Miscellaneous

Syntax

```
WWControl( AppTitle, ControlType );
```

Parameters

AppTitle

The name of the application title to be controlled. Actual string or a string attribute.

ControlType

Determines how the application is controlled. Possible values are shown below. These actions are identical to clicking on their corresponding selections in the application's Control Menu. Actual string or a string attribute.

"Restore" = Activates and shows the application's window.

"Minimize" = Activates a window and shows it as an icon.

"Maximize" = Activates and shows the application's window.

"Close" = Closes an application.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Example

```
WWControl("Calculator","Restore");
```

See also

ActivateApp()

OPC UA Methods

You can use Application Server object scripts and/or Industrial Graphics scripts to call OPC UA servers through aaMethods.

Note: InTouch does not support using OPC UA script methods. InTouch ViewApps will not recognize any of the aaMethods documented in this section.

aaMethod calls are executed in the OPC UA server through the OPC UA client, and are supported through the OPC client in the Gateway Communication Driver. aaMethods calls are divided into the following classes:

- Client class
- GenericStruct class
- GenericField class
- Value class
- MethodArgument class
- MethodCallStatus class

An Application Server script library allows System Platform script developers to call methods in OPC UA servers, via the OPC UA client in the Gateway Communication Driver. Internally, the script library leverages PCS infrastructure to facilitate secure off-node method calls.

To use OPC UA scripting, the Communication Driver Gateway must be configured with an OPC UA client, with the correct configuration settings (URL, security settings, etc.). This ensures that you can connect to the OPC UA server, thus exposing the method(s) of interest.

See Sample Application Server script for details.

Introduction to OPC UA scripts

The Application Server script library lets you invoke methods in OPC UA servers, via the OPC UA client in the Gateway Communication Driver.

The script library provides a high-level abstraction to enable calling OPC UA methods in scripts, without the need to know the details of the underlying OPC UA method call mechanism.

In general, an OPC UA client calls a method on an OPC UA server by specifying the method name (including the node id of the OPC UA object containing the method) and an array of input arguments. Once the method call completes, the OPC UA server returns an array of output arguments to the client. The input and output arguments may be either primitive data types, or complex user-defined types exposed by the OPC UA server, or a mixture of both.

Since the Gateway Communication Driver does not currently support browsing methods or user-defined types in

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

the OPC UA server, you must know the name of each method to be called, the OPC UA node id of the object hosting the method, its input arguments and their types to be supplied in the method call, as well as the output arguments and their types returned by the OPC UA server.

Third-party tools, such the UaExpert from Unified Automation, will let you browse the OPC UA server and thus can help you with scripting.

Function

Create a method client

Syntax

```
public Client CreateMethodClient(string ScopeName)
```

Description

Creates a `Method client` to the OI Server with the PCS Scopename in the ASB Solution and connects to it.

Create a method client when there are multiple servers or nodes

Syntax

```
public Client CreateMethodClient(string ScopeName, string ServerMachineName)
```

Description

Creates a Method client to the OI Server with the PCS Scopename in the ASB Solution on the server node, as the PCS ScopeName could be assigned to multiple servers within the node/across multiple nodes, in an ASB solution.

Client class

Create a new client instance

Syntax

```
public Client()
```

Description

Creates a new `Client` instance.

Initialize the client

Syntax

```
public bool Initialize(string ScopeName, string serverMachineName)
```

Description

Initializes the Client to connect to the specified Gateway Communication Driver instance on the specified computer. See the *Gateway Communication Driver User Guide* for details about how to specify the instance to which you want to connect.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Close the client

Syntax

```
public void Close()
```

Description

Closes the Client and disconnects from the server.

Call a method

Syntax

```
public MethodCallStatus CallMethod(string sourceId, string nodeId, string methodName,
string userName, MethodArgument[] inputArgs)
```

Description

Calls the specified method.

See the *Gateway Communication Driver User Guide* for details about specifying `sourceId` and `nodeId`. See the MethodArgument class for details on how to construct/deconstruct method arguments.

GenericStruct class

Create a new GenericStruct

Syntax

```
public GenericStruct(string structTypename)
```

Description

Creates a new, empty `GenericStruct` with the specified name.

Add a field to the GenericStruct

Syntax

```
public void AddField(string fieldName, Value fieldValue)
public void AddField(string fieldName, GenericStruct fieldValue)
public void AddField(string fieldName, params Value[] fieldValue)
public void AddField(string fieldName, params GenericStruct[] fieldValue)
public void Add(GenericField field)
```

Description

Adds a field to the `GenericStruct`. See Value class for additional information.

Access the GenericField

Syntax

```
public GenericField this[int index]
```

Description

Accesses the GenericField at the specified index (zero-based) in the GenericStruct.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Example

```
GenericField myField = myStruct[2]
```

GenericField class

Create a new GenericField

Syntax

```
public GenericField(string name, Value value)
public GenericField(string name, GenericStruct value)
```

Description

Creates a new `GenericField` with the specified name and value. The value can be either a `Value` or a `GenericStruct`, or an array of `Value` or `GenericStruct`. This allows nested structures to be created.

Properties

```
public string Name
public bool IsArray
public bool IsStructure
```

These properties determine:

- The name of the `GenericField`
- Whether its value is an array or of type `GenericStruct`, respectively. Otherwise, the value is of type `Value`.

```
public Values[] SimpleValue
```

This property determines the value of the `GenericField` if it is of type `Value` (null otherwise).

```
public Struct[] StructValue
```

This property contains the value of the `GenericField` if it is of type `GenericStruct` (null otherwise).

Access the value of the GenericField

Syntax

```
public object this[int key]
```

Description

Accesses the value of the `GenericField` at the specified index (zero-based).

Example

```
Value myVal = myField[0]
```

Value class

Create a new value

Syntax

```
public static Value Create(ValueType valtype, object value)
```

Description

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Creates a new `Value` of the specified `ValueType`. See ValueType enumerations for `Value` types.

Convert value to a different type

Syntax

```
public byte GetAsByte()
public sbyte GetAsSByte()
public char GetAsChar()
public bool GetAsBool()
public short GetAsInt16()
public ushort GetAsUInt16()
public int GetAsInt32()
public uint GetAsUInt32()
public long GetAsInt64()
public ulong GetAsUInt64()
public float GetAsSingle()
public double GetAsDouble()
public string GetAsString()
public decimal GetAsDecimal()
public DateTime GetAsDateTime()
public TimeSpan GetAsTimeSpan()
public Guid GetAsGuid()
public byte[] GetAsByteArray()
```

Description

Attempts to convert the `Value` to the specified type. See ValueType enumerations for `Value` types.

ValueType enumerations

The following list of `ValueType` enumerations are for use with `Value` class methods.

```
Empty
Char
Guid
```

Note: Guid enum is not supported in System Platform 2023 R2

```
Boolean
Byte
SByte
String
Int16
UInt16
Int32
UInt32
Int64
UInt64
Single
Double
Decimal
DateTime
TimeSpan
```

Note: TimeSpan enum is not supported in System Platform 2023 R2

```
Custom
```

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Note: Custom enum is not supported in System Platform 2023 R2

MethodArgument class

Create a new MethodArgument

Syntax

```
public MethodArgument(string name, Value value)
public MethodArgument(string name, Value[] value)
public MethodArgument(string name, GenericStruct value)
public MethodArgument(string name, GenericStruct[] value)
```

Description

Creates a new `MethodArgument` with the specified name and value. The value can be either a `Value` or a `GenericStruct` (or an array of `Value` or `GenericStruct`). This allows nested structures to be created.

Examples

```
public string Name
public bool IsArray
public bool IsStructure
```

Properties containing the name of the `MethodArgument` and whether its value is an array and of type `GenericStruct`, respectively (otherwise, the value is of type `Value`).

```
public Value[] SimpleValue
```

The value of the `MethodArgument` if it is of type `Value`, null otherwise.

```
public Struct[] StructValue
```

The value of the `MethodArgument` if it is of type `GenericStruct`, null otherwise.

```
public object this[int key]
```

Accesses the value of the `MethodArgument` at the specified index (zero-based).

For example:

```
Value myVal = myArg[0]
```

MethodCallStatus class

Get a unique RunId

Syntax

```
public Guid RunId { get; private set; }
```

Description

Gets a unique `RunId` for the method call.

Get the name of the method

Syntax

```
public string MethodName { get; private set; }
```

Description

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Gets the name of the method on the Method Server being called.

Get error code

Syntax

```
public int ErrorCode { get; private set; }
```

Description

Gets the error code on a call execution failure.

Get run status

Syntax

```
public MethodCallState RunStatus { get; private set; }
```

Description

`RunStatus` is an enumeration that notes the state of `MethodCallState`. See MethodCallState enumeration for details.

Get output values of method execution

Syntax

```
public MethodArgument[] OutputArguments { get; private set; }
```

Description

Returns output values on method execution. Values will be 'null' if method execution fails.

Get call identifier

Syntax

```
public string UserCallId { get; set; }
```

Description

Returns the unique call identifier that is supplied in the CallMethod API.

MethodCallState enumeration

MethodCallState enumerations are as follows:

```
public enum MethodCallState
{
NotStarted = 0,
Running,
Completed,
Failed,
TimedOut
}
```

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Sample Application Server script

The following sample ApplicationObject script utilizing the script library to call two methods on a OPC UA server is shown for illustration.

The first method (`Multiply`) accepts two arguments of type double and returns their product as a single argument, also of type double. In pseudo-syntax:

```
double Multiply(double x, double y)
```

Declarations

Make the following variable declarations to start building the script:

```
dim methodHelper as aaMethods.Client;
dim methodCallStatus as aaMethods. MethodCallStatus;
dim inputArgs[2] as aaMethods.MethodArgument;
dim outputArgs[1] as aaMethods.MethodArgument;
dim multiplyResult as aaMethods.Value;
```

Initialize the script library

Initialize the script library with connection information prior to calling methods:

```
methodHelper = CreateMethod.Client(PCSScopeName);
```

or

```
methodHelper = CreateMethodClient("PCSScopeName", "OIGatewayServerNodeName")
```

The `CreateMethodClient()` method requires the mandatory argument `PCSScopeName.` The argument `OIGatewayServerNodeName` is optional; this is the name of the machine that is hosting the instance of the Gateway Communication Driver that is connected to the remote OPC UA server. Refer to the *Gateway Communication Driver User Guide* for details on configuring a Gateway instance.

The `CreateMethodClient()` method returns an instance of `aaMethods.Client`, that is the handler for all Method calls to the Gateway Communication Driver. It returns a null value in case of failure to establish a connection.

Call a method with simple arguments

Next, the script calls a simple method (`Multiply`) on the OPC UA server, with two input arguments of type `double` and a single output argument, also of type `double`.

aaMethods supports two types of method calls:

1. CallMethod: This method is recommended when the script is executing in Asynchronous mode.
2. CallMethodAsync: This method is recommended when the script is executing in Synchronous mode. Note that CallMethodAsync is not supported in Visual Elements, i.e., in graphics and layouts.

CallMethod example:

```
inputArgs[1] = new aaMethods.MethodArgument("x",
       aaMethods.Value.Create(aaMethods.ValueType.Double, Me.X));
inputArgs[2] = new aaMethods.MethodArgument("y",
       aaMethods.Value.Create(aaMethods.ValueType.Double, Me.Y));
```

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

```
outputArgs = methodHelper.CallMethod("OPCUA_DeviceGroup",
      "OPCUA_DeviceGroup./DemoServer/s=Demo.Method", "Multiply",
      "Joe Operator", inputArgs);
```

The two input arguments are created and stored in the `inputArgs` array (note that in Application Server scripts, arrays are indexed starting at 1), using a static method in the script library (`aaMethods.Value.Create`) to generate a value of the specified type, in this case `double`. The references `Me.X` and `Me.Y` assume that the Application Object hosting the script has two attributes X and Y, both of type `double`.

The script library method `CallMethod()` instructs the Gateway Communication Driver to issue the specified method call to the OPC UA server. The `CallMethod()` arguments are:

- `OPCUA_DeviceGroup` and `OPCUA_DeviceGroup./DemoServer/s=Demo.Method`
  These uniquely identify the object in the OPC UA server hosting the method to be called.
- `Multiply` is the name of the method exposed by the OPC UA server object.
- `JoeOperator` is the name of the user calling the method.
- `inputArgs` is the array containing the method input arguments.
- `outputArgs` is the array containing the arguments returned from the method call.

The script stores the result of the method call in an object attribute `Result` of type `double`:

```
   multiplyResult = outputArgs[1][0]
   Me.Result = multiplyResult.GetAsDouble();
```

CallMethodAsync example

```
inputArgs[1] = new aaMethods.MethodArgument("x",
      aaMethods.Value.Create(aaMethods.ValueType.Double, Me.X));
inputArgs[2] = new aaMethods.MethodArgument("y",
      aaMethods.Value.Create(aaMethods.ValueType.Double, Me.Y));
if(methodCallStatus == null ) or (methodCallStatus.RunStatus ==
aMethods.MethodCallState.NotStarted) then
      methodCallStatus = client.CallMethodAsync("ScriptName",” OPCUA_DeviceGroup”,
      "/DemoServer/s=Demo.Method", "Multiply", "Joe Operator", inputArgs);
endif;
outputArgs = methodCallStatus. OutputArguments;
```

The script stores the result of the method call in the object attribute `Result` of type `double`:

```
   multiplyResult = outputArgs[1][0]
   Me.Result = multiplyResult.GetAsDouble();
```

The `CallMethodAsync()` arguments are:

- `ScriptName` is the name of the userCallerID. Value of the CallerID, which can be any identifier string/ scriptname. It can also be empty.
- `OPCUA_DeviceGroup` and `OPCUA_DeviceGroup./DemoServer/s=Demo.Method` uniquely identify the object in the OPC UA server hosting the method to be called. Refer to the *Gateway Communication Driver User Guide* for details.
- `Multiply` is the name of the method exposed by the OPC UA server object.
- `JoeOperator` is the name of the user calling the method.
- `inputArgs` is the array containing the method input arguments.
- `outputArgs` is the array containing the arguments returned from the method call.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

How the script library handles MethodArgument values

The `MethodArgument` class in the script library stores argument values in one of two arrays called `SimpleValue` and `StructValue`.

- If the actual argument value is a single value instead of an array, it is always contained in `SimpleValue[0]` or `StructValue[0]`.
- If the argument value is a simple (non-structured) type, that is, of type `Value`, it is stored in the `SimpleValue` array.
- If the argument is a structured type (`GenericStruct`), it is stored in the `StructValue` array.
- This also applies to the `GenericField` class. See Use complex (structured) data for more information.

Generally speaking, it is expected that you know the types of method arguments you are invoking. As an aid, however, the `MethodArgument` class also has two public properties (`IsArray` and `IsStructure`) that indicate whether the method argument is an array or a single value, and whether it is a simple type (`Value`) or a structured type (`GenericStruct`).

To illustrate dealing with `MethodArgument` data, consider the two lines of script code below:

```
multiplyResult = outputArgs[1][0]
Me.Result = multiplyResult.GetAsDouble();
```

This is equivalent to the following pseudo-code:

```
1. MethodArgument argument = outputArgs[1]
2. Value multiplyResult = argument[0]
3. double actualValue = multiplyResult.GetAsDouble();
4. Me.Result = actualValue;
```

Line 1 stores the (one and only) `MethodArgument` returned by the method call in argument (of type MethodArgument).

Line 2 calls an indexer in the `MethodArgument` class to return the first (and only, in this case) method argument value (of type `Value`) and stores it in `multiplyResult`. If the method argument value was a structured type, `argument[0]` would return a `GenericStruct`.

In Line 3, the actual value of the method call result of type `double` is obtained by converting the `Value` into a `double` by calling the `GetAsDouble()` method in the `Value` class.

Note: The first time the `CallMethodAsync` API is called, the script may log a timeout warning when run in synchronous mode, as the API sets up the connection with the MethodProvider in the Gateway Server. This warning can be ignored as the method call runs in a background worker and returns the Method OutputArguments, on completion.

Use complex (structured) data

The second method (`SubmitWorkOrder`) is more advanced, accepting a single argument of type `WorkOrder` and returning a single argument, also of type `WorkOrder`. In pseudo-syntax:

```
WorkOrder SubmitWorkOrder(WorkOrder wo)
```

The `WorkOrder` type is a fictitious composite structure type defined by the OPC UA server with the following definition:

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.



`WorkOrderType` contains three primitive type fields (`ID`, `AssetID` and `StartTime`), and a field (`StatusComments`) that contains an array of `WorkOrderStatusType` structures.

Declarations

Add the following variable declarations to the script:

```
dim wOrder as aaMethods.GenericStruct;
dim wOrderIn as aaMethods.GenericStruct;
dim statusComment1 as aaMethods.GenericStruct;
dim statusComment2 as aaMethods.GenericStruct;
```

Call a method with complex (structured) arguments

In this example, the script calls a method (`SubmitWorkOrder`) on the OPC UA server, with a single input argument of type `WorkOrderType` and a single output argument, also of type `WorkOrderType`. The OPC UA server method implementation adds an additional `StatusCommentType` to the passed-in `WorkOrderType` and returns the result to the caller.

The first step is to create the method input argument by creating an empty structure named `WorkOrderType`:

```
wOrderIn = methodHelper.CreateStruct("WorkOrderType");
```

This simply creates a `GenericStruct` with no fields. As mentioned previously, the script library has no knowledge of the actual `WorkOrderType` defined in the OPC UA server. As the script developer, it is up to you to ensure that the `WorkOrderType` structure created in the script library mirrors the actual type defined in the OPC UA server.

Next, start adding the appropriate fields to the `WorkOrderType GenericStruct`:

```
wOrderIn.AddField("ID", aaMethods.Value.Create(aaMethods.ValueType.String,
   Me.WorkOrderId));
wOrderIn.AddField("AssetID", aaMethods.Value.Create(aaMethods.ValueType.String,
   Me.WorkOrderAssetId));
wOrderIn.AddField("StartTime", aaMethods.Value.Create(aaMethods.ValueType.String,
   Me.WorkOrderStartTime));
```

For the sake of simplicity, assume that the Application Object has attributes `WorkOrderId`, `WorkOrderAssetId` and `StartTime`, all of type `string`. Note that all Application Server attributes types except `CustomStruct` are supported.

The `WorkOrderType` contains an array of `WorkOrderStatusType` structures in a field called StatusComments. Create the WorkOrderStatusType structures as shown below:

```
statusComment1 = methodHelper.CreateStruct("WorkOrderStatusType");
statusComment1.AddField("Actor", aaMethods.Value.Create(aaMethods.ValueType.String,
   Me.WorkOrderHistoryActor[1]));
statusComment1.AddField("TimeStamp",
   aaMethods.Value.Create(aaMethods.ValueType.String,
   Me.WorkOrderHistoryTime[1]));
statusComment1.AddField("Comment",
   aaMethods.Value.Create(aaMethods.ValueType.String,
```

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

```
   Me.WorkOrderHistoryComment[1]));
statusComment2 = methodHelper.CreateStruct("WorkOrderStatusType");
statusComment2.AddField("Actor", aaMethods.Value.Create(aaMethods.ValueType.String,
   Me.WorkOrderHistoryActor[2]));
statusComment2.AddField("TimeStamp",
   aaMethods.Value.Create(aaMethods.ValueType.String,
   Me.WorkOrderHistoryTime[2]));
statusComment2.AddField("Comment",
   aaMethods.Value.Create(aaMethods.ValueType.String,
   Me.WorkOrderHistoryComment[2]));
```

Add the `StatusComments` field to the `WorkOrderType`:

```
wOrderIn.AddField("StatusComments", statusComment1, statusComment2);
```

The `WorkOrderType GenericStruct` now resembles the `WorkOrderType` defined earlier.

Finally, create the input argument and call the `SubmitWorkOrder` method:

```
inputArgs[1] = new aaMethods.MethodArgument("WorkOrder", wOrderIn);
outputArgs = methodHelper.CallMethod(("OPCUA_DeviceGroup",
   "OPCUA_DeviceGroup./DemoServer/s=Demo.Method", "SubmitWorkOrder", "Joe Operator",
   inputArgs);
```

Process the result

As before, we get the result from the output argument returned by the method call:

```
wOrder = outputArgs[1][0];
```

In this case, we know that `outputArgs[1]` contains a single `GenericStruct` representing the `WorkOrderType` structure returned by the OPC UA server.

The individual `WorkOrderType` fields are accessed as before:

```
Me.WorkOrderId = wOrder.GetAsString("ID");
Me.WorkOrderAssetId = wOrder.GetAsString("AssetID");
Me.WorkOrderStartTime = wOrder.GetAsString("StartTime");
Me.WorkOrderHistoryActor[1] = wOrder.GetAsString("StatusComments[0].Actor");
Me.WorkOrderHistoryComment[1] = wOrder.GetAsString("StatusComments[0].Comment");
Me.WorkOrderHistoryTime[1] = wOrder.GetAsString("StatusComments[0].TimeStamp");
Me.WorkOrderHistoryActor[2] = wOrder.GetAsString("StatusComments[1].Actor");
Me.WorkOrderHistoryComment[2] = wOrder.GetAsString("StatusComments[1].Comment");
Me.WorkOrderHistoryTime[2] = wOrder.GetAsString("StatusComments[1].TimeStamp");
Me.WorkOrderHistoryActor[3] = wOrder.GetAsString("StatusComments[2].Actor");
Me.WorkOrderHistoryComment[3] = wOrder.GetAsString("StatusComments[2].Comment");
Me.WorkOrderHistoryTime[3] = wOrder.GetAsString("StatusComments[2].TimeStamp");
```

Note that the fields of the contained structure `WorkOrderStatusType` are accessed by specifying the index into the `StatusComment` field of the `WorkOrderType` structure, for example:

```
Me.WorkOrderHistoryActor[2] = wOrder.GetAsString("StatusComments[1].Actor");
```

Limitations

The PCS Scope Name, defined in the OI Server configuration, is used in the PCS discovery during the client connection establishment stage by the aaMethods Client. The OI Server, if started before the platform is deployed to the node, may not have the correct ASB Solution in one of its scopes and thus the method client connection establishment could fail. This can be overcome by restarting the OI Server after deploying a platform

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

to the OI Server node.

String functions

Use string functions to work with character strings and string values.

DText()

Returns one of two possible strings, depending on the value of the *Discrete* parameter.

Category

String

Syntax

```
StringResult = DText( Discrete, OnMsg, OffMsg );
```

Parameters

Discrete

A Boolean value or Boolean attribute.

OnMsg

The message that is shown when the value of *Discrete* equals true.

OffMsg

The message shown when *Discrete* equals false.

Example

```
StringResult = DText(me.temp > 150, "Too hot", "Just right");
```

StringASCII()

Returns the ASCII value of the first character in a specified string.

Category

String

Syntax

```
IntegerResult = StringASCII( Char );
```

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Parameter

Char

Alphanumeric character or string or string attribute.

Remarks

When this function is processed, only the single character is tested or affected. If the string provided to StringASCII contains more than one character, only the first character of the string is tested.

Examples

```
StringASCII("A"); ' returns 65;
StringASCII("A Mixer is Running"); ' returns 65;
StringASCII("a mixer is running"); ' returns 97;
```

See also

StringChar()
StringFromIntg()
StringFromReal()
StringFromTime()
StringInString()
StringLeft()
StringLen()
StringLower()
StringMid()
StringReplace()
StringRight()
StringSpace()
StringTest()
StringToIntg()
StringToReal()
StringTrim()
StringUpper()
Text()

StringChar()

Returns the character corresponding to a specified ASCII code.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Category

String

Syntax

```
StringResult = StringChar( ASCII );
```

Parameter

ASCII

ASCII code or an integer attribute.

Remarks

Use the StringChar function to add ASCII characters not normally represented on the keyboard to a string attribute.

This function is also useful for SQL commands. The where expression sometimes requires double quotation marks around string values, so use StringChar(34).

Example

In this example, a [Carriage Return (13)] and [Line Feed (10)] are added to the end of StringAttribute and passed to ControlString. Inserting characters out of the normal 32-126 range of displayable ASCII characters can be very useful for creating control codes for external devices such as printers or modems.

```
ControlString = StringAttribute+StringChar(13)+StringChar(10);
```

StringCompare()

Compares a string value with another string.

Category

String

Syntax

```
StringCompare( Text1, Text2 );
```

Parameters

Text1

First string in the comparison.

Text2

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Second string in the comparison.

Return value

The return value is zero if the strings are identical, -1 if Text1's value is less than Text2, or 1 if Text1's value is greater than Text2.

Example

```
Result = StringCompare ("Text1","Text2"); (or)
Result = StringCompare (MText1,MText2);
Where Result is an Integer or Real tag and MText1 and MText2 are Memory Message tags.
```

See also

StringASCII()
StringChar()
StringFromIntg()
StringFromReal()
StringFromTime()
StringFromTimeLocal()
StringInString()
StringLeft()
StringLen()
StringLower()
StringMid()
StringReplace()
StringRight()
StringSpace()
StringTest()
StringToIntg()
StringToReal()
StringTrim()
StringUpper()
Text()

StringCompareNoCase()

Compares a string value with another string and ignores the case.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Category

String

Syntax

```
SStringCompareNoCase( Text1, Text2 );
```

Parameters

Text1

First string in the comparison.

Text2

Second string in the comparison.

Return Value

The return value is zero if the strings are identical (ignoring case), -1 if Text1's value is less than Text2 (ignoring case), or 1 if Text1's value is greater than Text2 (ignoring case).

Example

```
Result = StringCompareNoCase ("Text1","TEXT1"); (or)
Result = StringCompareNoCase (MText1,MText2);
Where Result is an Integer or Real tag and MText1 and MText2 are Memory Message tags.
```

See Also

StringASCII()
StringChar()
StringFromIntg()
StringFromReal()
StringFromTime()
StringFromTimeLocal()
StringInString()
StringLeft()
StringLen()
StringLower()
StringMid()
StringReplace()
StringRight()
StringSpace()

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

StringTest()
StringToIntg()
StringToReal()
StringTrim()
StringUpper()
Text()

StringFromGMTTimeToLocal()

Converts a time value (in seconds since Jan-01-1970) to a particular string representation. This is the same as StringFromTime().

Category

String

Syntax

```
MessageResult=StringFromGMTTimeToLocal(SecsSince1-1-70,StringType);
```

Parameters

SecsSince1-1-70

Is converted to the StringType specified and the result is stored in MessageResult.

StringType

Determines the display method:

1 = Displays the date in the same format set from the windows control Panel. (Similar to that displayed for $DateString.)

2 = Displays the time in the same format set from the Windows control Panel. (Similar to that displayed for $TimeString.)

3 = Displays a 24-character string indicating both the date and time: "Wed Jan 02 02:03:55 1993"

4 = Displays the short form for the day of the week: "Wed"

5 = Displays the long form for the day of the week: "Wednesday"

Remarks

Any adjustments necessary due to Daylight Savings Time are automatically applied to the return result. Therefore, it is not necessary to make any manual adjustments to the input value to convert to DST.

Example

This example assumes that the time zone on the local node is Pacific Standard Time (UTC-0800). The UTC time passed to the function is 12:00:00 AM on Friday, 1/2/1970. Since PST is 8 hours behind UTC, the function will

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

return the following results:

```
StringFromGMTTimeToLocal(86400, 1); ' returns "1/1/1970"
StringFromGMTTimeToLocal(86400, 2); ' returns "04:00:00 PM"
StringFromGMTTimeToLocal(86400, 3); ' returns "Thu Jan 01 16:00:00 1970"
StringFromGMTTimeToLocal(86400, 4); ' returns "Thu"
StringFromGMTTimeToLocal(86400, 5); ' returns "Thursday"
```

See also

StringASCII()
StringChar()
StringFromIntg()
StringFromReal()
StringFromTime()
StringFromTimeLocal()
StringInString()
StringLeft()
StringLen()
StringLower()
StringMid()
StringReplace()
StringRight()
StringSpace()
StringTest()
StringToIntg()
StringToReal()
StringTrim()
StringUpper()
Text()

StringFromIntg()

Converts an integer value into its string representation in another base and returns the result.

Category

String

Syntax

```
SringResult = StringFromIntg( Number, numberBase );
```

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Parameters

Number

Number to convert. Any number or an integer attribute.

numberBase

Base to use in conversion. Any number or an integer attribute.

Examples

```
StringFromIntg(26, 2); ' returns "11010"
StringFromIntg(26, 8); ' returns "32"
StringFromIntg(26, 16); ' returns "1A"
```

See also

StringASCII()
StringChar()
StringFromReal()
StringFromTime()
StringInString()
StringLeft()
StringLen()
StringLower()
StringMid()
StringReplace()
StringRight()
StringSpace()
StringTest()
StringToIntg()
StringToReal()
StringTrim()
StringUpper()
Text()

StringFromReal()

Converts a real value into its string representation, either as a floating-point number or in exponential notation, and returns the result.

Category

String

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Syntax

```
StringResult = StringFromReal( Number, Precision, Type );
```

Parameters

Number

Converted to the *Precision* and *Type* specified. Any number or a float attribute.

Precision

Specifies how many decimal places is shown. Any number or an integer attribute.

Type

A string value that determines the display method. Possible values are:

f = Display in floating-point notation.

e = Display in exponential notation with a lowercase "e."

E = Display in exponential notation with an uppercase "E" followed by a plus sign and at least three exponential digits.

Examples

```
StringFromReal(263.355, 2,"f"); ' returns "263.36";
StringFromReal(263.355, 2,"e"); ' returns "2.63e2";
StringFromReal(263.355, 2,"E"); ' returns "2.63 E+002";
```

See also

StringASCII()
StringChar()
StringFromIntg()
StringFromTime()
StringInString()
StringLeft()
StringLen()
StringLower()
StringMid()
StringReplace()
StringRight()
StringSpace()
StringTest()
StringToIntg()
StringToReal()
StringTrim()

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

StringUpper()

Text()

StringFromTime()

Converts a time value (in seconds since January 1, 1970) into a particular string representation and returns the result.

Category

String

Syntax

```
StringResult = StringFromTime( SecsSince1-1-70, StringType );
```

Parameters

SecsSince1-1-70

Converted to the *StringType* specified.

StringType

Determines the display method. Possible values are:

1 = Shows the date in the same format set from the Windows Control Panel.

2 = Shows the time in the same format set from the Windows Control Panel.

3 = Shows a 24-character string indicating both the date and time: "Wed Jan 02 02:03:55 1993"

4 = Shows the short form for a day of the week: "Wed"

5 = Shows the long form for a day of the week: "Wednesday"

Remarks

The time value is UTC equivalent: number of elapsed seconds since January 1, 1970 GMT. The value returned reflects the local time.

Examples

```
StringFromTime(86400, 1); ' returns "1/2/1970"
StringFromTime(86400, 2); ' returns "12:00:00 AM"
StringFromTime(86400, 3); ' returns "Fri Jan 02 00:00:00 1970"
StringFromTime(86400, 4); ' returns "Fri"
StringFromTime(86400, 5); ' returns "Friday"
```

See also

StringASCII()

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

StringChar()
StringFromIntg()
StringFromReal()
StringFromTime()
StringInString()
StringLeft()
StringLen()
StringLower()
StringMid()
StringReplace()
StringRight()
StringSpace()
StringTest()
StringToIntg()
StringToReal()
StringTrim()
StringUpper()
Text()

StringFromTimeLocal()

Converts a time value (in seconds since Jan-01-1970) into a particular string representation. The value returned also represents local time.

Category

String

Syntax

```
MessageResult=StringFromTimeLocal(SecsSince1-1-70,
StringType);
```

Parameters

SecsSince1-1-70

Is converted to the StringType specified and the result is stored in MessageResult.

StringType

Determines the display method:

1 = Displays the date in the same format set from the windows control Panel. (Similar to that displayed for $DateString.)

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

2 = Displays the time in the same format set from the Windows control Panel. (Similar to that displayed for $TimeString.)

3 = Displays a 24-character string indicating both the date and time: "Wed Jan 02 02:03:55 1993"

4 = Displays the short form for the day of the week: "Wed"

5 = Displays the long form for the day of the week: "Wednesday"

Remarks

Any adjustments necessary due to Daylight Savings Time will automatically be applied to the return result. Therefore, it is not necessary to make any manual adjustments for DST to the input value.

Example

```
StringFromTimeLocal (86400, 1); ' returns "1/2/1970"
StringFromTimeLocal (86400, 2); ' returns "12:00:00 AM"
StringFromTimeLocal (86400, 3); ' returns "Fri Jan 02 00:00:00 1970"
StringFromTimeLocal (86400, 4); ' returns "Fri"
StringFromTimeLocal (86400, 5); ' returns "Friday"
```

See also

StringASCII()
StringChar()
StringFromIntg()
StringFromReal()
StringFromTime()
StringInString()
StringLeft()
StringLen()
StringLower()
StringMid()
StringReplace()
StringRight()
StringSpace()
StringTest()
StringToIntg()
StringToReal()
StringTrim()
StringUpper()
Text()

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

StringInString()

Returns the position in a string of text where a specified string first occurs.

Category

String

Syntax

```
IntegerResult = StringInString( Text, SearchFor, StartPos, CaseSens );
```

Parameters

Text

The string that is searched. Actual string or a string attribute.

SearchFor

The string to be searched for. Actual string or a string attribute.

StartPos

Determines the position in the text where the search begins. Any number or an integer attribute.

CaseSens

Determines whether the search is case-sensitive.

```
   0 = Not case-sensitive
   1 = Case-sensitive
   Any number or an integer attribute.
```

Remarks

If multiple occurrences of *SearchFor* are found, the location of the first is returned.

Examples

```
StringInString("The mixer is running", "mix", 1, 0) ' returns 5;
StringInString("Today is Thursday", "day", 1, 0) ' returns 3;
StringInString("Today is Thursday", "day", 10, 0) ' returns 15;
StringInString("Today is Veteran's Day", "Day", 1, 1) ' returns 20;
StringInString("Today is Veteran's Day", "Night", 1, 1) ' returns 0;
```

See also

StringASCII()

StringChar()

StringFromIntg()

StringFromReal()

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

StringFromTime()
StringLeft()
StringLen()
StringLower()
StringMid()
StringReplace()
StringRight()
StringSpace()
StringTest()
StringToIntg()
StringToReal()
StringTrim()
StringUpper()
Text()

StringLeft()

Returns a specified number of characters in a string value, starting with the leftmost string character.

Category

String

Syntax

```
StringResult = StringLeft( Text, Chars );
```

Parameters

Text

Actual string or a string attribute.

Chars

Number of characters to return or an integer attribute.

Remarks

If *Chars* is set to 0, the entire string is returned.

Examples

```
StringLeft("The Control Pump is On", 3) ' returns "The";
StringLeft("Pump 01 is On", 4) ' returns "Pump";
StringLeft("Pump 01 is On", 96) ' returns "Pump 01 is On";
```

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

```
StringLeft("The Control Pump is On", 0) ' returns "The Control Pump is On";
```

See also

StringASCII()
StringChar()
StringFromIntg()
StringFromReal()
StringFromTime()
StringInString()
StringLen()
StringLower()
StringMid()
StringReplace()
StringRight()
StringSpace()
StringTest()
StringToIntg()
StringToReal()
StringTrim()
StringUpper()
Text()

StringLen()

Returns the number of characters in a string.

Category

String

Syntax

```
IntegerResult = StringLen( Text );
```

Parameter

Text
Actual string or a string attribute.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Remarks

All the characters in the string attribute are counted, including blank spaces and those not normally shown on the screen.

Examples

```
StringLen("Twelve percent") ' returns 14;
StringLen("12%") ' returns 3;
StringLen("The end." + StringChar(13)) ' returns 9;
```

The carriage return character is ASCII 13.

See also

StringASCII()
StringChar()
StringFromIntg()
StringFromReal()
StringFromTime()
StringInString()
StringLeft()
StringLower()
StringMid()
StringReplace()
StringRight()
StringSpace()
StringTest()
StringToIntg()
StringToReal()
StringTrim()
StringUpper()
Text()

StringLower()

Converts all uppercase characters in text string to lowercase and returns the result.

Category

String

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Syntax

```
StringResult = StringLower( Text );
```

Parameter

Text

String to be converted to lowercase. Actual string or a string attribute.

Remarks

Lowercase characters, symbols, numbers, and other special characters are not affected.

Examples

```
StringLower("TURBINE") ' returns "turbine";
StringLower("22.2 Is The Value") ' returns "22.2 is the value";
```

See also

StringASCII()
StringChar()
StringFromIntg()
StringFromReal()
StringFromTime()
StringInString()
StringLeft()
StringLen()
StringMid()
StringReplace()
StringRight()
StringSpace()
StringTest()
StringToIntg()
StringToReal()
StringTrim()
StringUpper()
Text()

StringMid()

Extracts a specific number of characters from a starting point within a string and returns the extracted character

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

string as the result.

Category

String

Syntax

```
StringResult = StringMid( Text, StartChar, Chars );
```

Parameters

Text

Actual string or a string attribute to extract a range of characters.

StartChar

The position of the first character within the string to extract. Any number or an integer attribute.

Chars

The number of characters within the string to return. Any number or an integer attribute.

Remarks

This function is slightly different than the StringLeft() function and StringRight() function in that it allows you to specify both the start and end of the string that is to be extracted.

Examples

```
StringMid("The Furnace is Overheating",5,7); ' returns "Furnace";
StringMid("The Furnace is Overheating",13,3); ' returns "is ";
StringMid("The Furnace is Overheating",16,50); ' returns "Overheating"
```

See also

StringASCII()
StringChar()
StringFromIntg()
StringFromReal()
StringFromTime()
StringInString()
StringLeft()
StringLen()
StringLower()
StringReplace()
StringRight()

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

StringSpace()
StringTest()
StringToIntg()
StringToReal()
StringTrim()
StringUpper()
Text()

StringReplace()

Replaces or changes specific parts of a provided string and returns the result.

Category

String

Syntax

```
StringResult = StringReplace( Text, SearchFor, ReplaceWith, CaseSens, NumToReplace,
MatchWholeWords );
```

Parameters

Text

The string in which characters, words, or phrases will be replaced. Actual string or a string attribute.

SearchFor

The string to search for and replace. Actual string or a string attribute.

ReplaceWith

The replacement string. Actual string or a string attribute.

CaseSens

Determines whether the search is case-sensitive. (0=no and 1=yes) Any number or an integer attribute.

NumToReplace

Determines the number of occurrences to replace. Any number or an integer attribute. To indicate all occurrences, set this value to -1.

MatchWholeWords

Determines whether the function limits its replacement to whole words. (0=no and 1=yes) Any number or an integer attribute. If *MatchWholeWords* is turned on (set to 1) and the *SearchFor* is set to "and", the "and" in "handle" are not replaced. If the *MatchWholeWords* is turned off (set to 0), it is replaced.

Remarks

Use this function to replace characters, words, or phrases within a string.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

The StringReplace() function does not recognize special characters, such as @ # $ % & * ( ). It reads them as delimiters. For example, if the function StringReplace() (abc#,abc#,1234,0,1,1) is processed, there is no replacement. The # sign is read as a delimiter instead of a character.

Examples

```
StringReplace("In From Within","In","Out",0,1,0) ' returns "Out From Within" (replaces only
the first one);
StringReplace("In From Within","In","Out",0,-1,0) ' returns "Out From without" (replaces
all occurrences);
StringReplace("In From Within","In","Out",1,-1,0) ' returns "Out From Within" (replaces all
that match case);
StringReplace("In From Within","In","Out",0,-1,1) ' returns "Out From Within" (replaces all
that are whole words);
```

See also

StringASCII()
StringChar()
StringFromIntg()
StringFromReal()
StringFromTime()
StringInString()
StringLeft()
StringLen()
StringLower()
StringMid()
StringRight()
StringSpace()
StringTest()
StringToIntg()
StringToReal()
StringTrim()
StringUpper()
Text()

StringRight()

Returns the specified number of characters starting at the right-most character of text.

Category

String

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Syntax

```
StringResult = StringRight( Text, Chars );
```

Parameters

Text

Actual string or a string attribute.

Chars

The number of characters to return or an integer attribute.

Remarks

If *Chars* is set to 0, the entire string is returned.

Examples

```
StringRight("The Pump is On", 2) ' returns "On";
StringRight("The Pump is On", 5) ' returns "is On";
StringRight("The Pump is On", 87) ' returns "The Pump is On";
StringRight("The Pump is On", 0) ' returns "The Pump is On";
```

See also

StringASCII()
StringChar()
StringFromIntg()
StringFromReal()
StringFromTime()
StringInString()
StringLeft()
StringLen()
StringLower()
StringMid()
StringReplace()
StringSpace()
StringTest()
StringToIntg()
StringToReal()
StringTrim()
StringUpper()
Text()

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

StringSpace()

Generates a string of spaces either within a string attribute or within an expression and returns the result.

Category

String

Syntax

```
StringResult = StringSpace( NumSpaces );
```

Parameter

NumSpaces

Number of spaces to return. Any number or an integer attribute.

Examples

All spaces are represented by the "×" character:

```
StringSpace(4) ' returns "××××";
"Pump" + StringSpace(1) + "Station" ' returns "Pump×Station";
```

See also

StringASCII()
StringChar()
StringFromIntg()
StringFromReal()
StringFromTime()
StringInString()
StringLeft()
StringLen()
StringLower()
StringMid()
StringReplace()
StringRight()
StringTest()
StringToIntg()
StringToReal()
StringTrim()
StringUpper()

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Text()

StringTest()

Tests the first character of text to determine whether it is of a certain type and returns the result.

Category

String

Syntax

```
DiscreteResult = StringTest( Text, TestType );
```

Parameters

Text

String that function acts on. Actual string or a string attribute.

TestType

Determines the type of test. Possible values are:

1 = Alphanumeric character ('A-Z', 'a-z' and '0-9')

2 = Numeric character ('0- 9')

3 = Alphabetic character ('A-Z' and 'a-z')

4 = Uppercase character ('A-Z')

5 = Lowercase character ('a'-'z')

6 = Punctuation character (0x21-0x2F)

7 = ASCII characters (0x00 - 0x7F)

8 = Hexadecimal characters ('A-F' or 'a-f' or '0-9')

9 = Printable character (0x20-0x7E)

10 = Control character (0x00-0x1F or 0x7F)

11 = White Space characters (0x09-0x0D or 0x20)

Remarks

StringTest() function returns true to *DiscreteResult* if the first character in *Text* is of the type specified by *TestType*. Otherwise, false is returned. If the StringTest() function contains more than one character, only the first character of the attribute is tested.

Examples

```
StringTest("ACB123",1) ' returns 1;
StringTest("ABC123",5) ' returns 0;
```

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

See also

StringASCII()
StringChar()
StringFromIntg()
StringFromReal()
StringFromTime()
StringInString()
StringLeft()
StringLen()
StringLower()
StringMid()
StringReplace()
StringRight()
StringSpace()
StringToIntg()
StringToReal()
StringTrim()
StringUpper()
Text()

StringToIntg()

Converts the numeric value of a string to an integer value and returns the result.

Category

String

Syntax

```
IntegerResult = StringToIntg( Text );
```

Parameter

Text
String that function acts on. Actual string or a string attribute.

Remarks

When this statement is evaluated, the system reads the first character of the string for a numeric value. If the first character is other than a number, the string's value is equated to zero (0). Blank spaces are ignored. If the

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

first character is a number, the system continues to read the subsequent characters until a non-numeric value is detected.

Examples

```
StringToIntg("ABCD"); ' returns 0;
StringToIntg("22.2 is the Value"); ' returns 22 (since integers are whole numbers);
StringToIntg("The Value is 22"); ' returns 0;
```

See also

StringASCII()
StringChar()
StringFromIntg()
StringFromReal()
StringFromTime()
StringInString()
StringLeft()
StringLen()
StringLower()
StringMid()
StringReplace()
StringRight()
StringSpace()
StringTest()
StringToReal()
StringTrim()
StringUpper()
Text()

StringToReal()

Converts the numeric value of a string to a real (floating point) value and returns the result.

Category

String

Syntax

```
RealResult = StringToReal( Text );
```

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Parameter

Text

String that function acts on. Actual string or a string attribute.

Remarks

When this statement is evaluated, the system reads the first character of the string for a numeric value. If the first character is other than a number (blank spaces are ignored), the string's value is equated to zero (0). If the first character is found to be a number, the system continues to read the subsequent characters until a non-numeric value is encountered.

Examples

```
StringToReal("ABCD"); ' returns 0;
StringToReal("22.261 is the value"); ' returns 22.261;
StringToReal("The Value is 2"); ' returns 0;
```

See also

StringASCII()
StringChar()
StringFromIntg()
StringFromReal()
StringFromTime()
StringInString()
StringLeft()
StringLen()
StringLower()
StringMid()
StringReplace()
StringRight()
StringSpace()
StringTest()
StringToIntg()
StringTrim()
StringUpper()
Text()

StringTrim()

Removes unwanted spaces from text and returns the result.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Category

String

Syntax

```
StringResult = StringTrim( Text, TrimType );
```

Parameter

Text

String that is trimmed of spaces. Actual string or a string attribute.

TrimType

Determines how the string is trimmed. Possible values are:

1 = Remove leading spaces to the left of the first non-space character

2 = Remove trailing spaces to the right of the last non-space character

3 = Remove all spaces except for single spaces between words

Remarks

The text is searched for white-spaces (ASCII 0x09-0x0D or 0x20) that are to be removed. *TrimType* determines the method used by the function:

Examples

All spaces are represented by the "×" character.

```
StringTrim("×××××This×is×a××test×××××", 1) ' returns "This×is×a××test×××××";
StringTrim("×××××This×is×a××test×××××", 2) ' returns "×××××This×is×a××test";
StringTrim("×××××This×is×a××test×××××", 3) ' returns "This×is×a×test";
```

The StringReplace() function can remove ALL spaces from a specified a string attribute. Simply replace all the space characters with a "null."

See also

StringASCII()

StringChar()

StringFromIntg()

StringFromReal()

StringFromTime()

StringInString()

StringLeft()

StringLen()

StringLower()

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

StringMid()
StringReplace()
StringRight()
StringSpace()
StringTest()
StringToIntg()
StringToReal()
StringUpper()
Text()

StringUpper()

Converts all lowercase text characters to uppercase and returns the result.

Category

String

Syntax

```
StringResult = StringUpper( Text );
```

Parameter

Text
String to be converted to uppercase. Actual string or a string attribute.

Remarks

Uppercase characters, symbols, numbers, and other special characters are not affected.

Examples

```
StringUpper("abcd"); ' returns "ABCD";
StringUpper("22.2 is the value"); ' returns "22.2 IS THE VALUE";
```

See also

StringASCII()
StringChar()
StringFromIntg()
StringFromReal()
StringFromTime()

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

StringInString()
StringLeft()
StringLen()
StringLower()
StringMid()
StringReplace()
StringRight()
StringSpace()
StringTest()
StringToIntg()
StringToReal()
StringTrim()
Text()

Text()

Converts a number to text based on a specified format.

Category

String

Syntax

```
StringResult = Text( Number, Format );
```

Parameters

Number

Any number or numeric attribute.

Format

Format to use in conversion. Actual string or a string attribute.

Examples

```
Text(66,"#.00"); ' returns 66.00;
Text(22.269,"#.00"); ' returns 22.27;
Text(9.999,"#.00"); ' returns 10.00;
```

The following example shows how to use this function within another function:

```
LogMessage("The current value of FreezerRoomTemp is:" + Text (FreezerRoomTemp, "#.#"));
```

In the following example, MessageTag is set to "One=1 Two=2".

```
MessageTag = "One + " + Text(1,"#") + StringChar(32) + "Two +" + Text(2,"#");
```

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

See also

StringFromIntg()

StringToIntg()

StringFromReal()

StringToReal()

WWStringFromTime()

Converts a time value given in local time into UTC time (Coordinated Universal Time), and displays the result as a string.

Category

String

Syntax

```
MessageResult = wwStringFromTime(SecsSince1-1-70,StringType);
```

Parameters

SecsSince1-1-70

Integer Type. Number of Seconds elapsed since Jan 01 00:00:00 1970.

StringType

Determines the display method:

1 = Displays the date in the same format set from the windows control Panel. (Similar to that displayed for $DateString.)

2 = Displays the time in the same format set from the Windows control Panel. (Similar to that displayed for $TimeString.)

3 = Displays a 24-character string indicating both the date and time: "Wed Jan 02 02:03:55 1993"

4 = Displays the short form for the day of the week: "Wed"

5 = Displays the long form for the day of the week: "Wednesday"

Remarks

Any adjustments necessary due to Daylight Savings Time will automatically be applied to the return result. Therefore, it is not necessary to make any manual adjustments for DST to the input value.

Example

This example assumes that the time zone on the local node is Pacific Standard Time (UTC-0800). The local time passed to the function is 04:00:00 PM on Thursday, 1/1/1970. Since PST is 8 hours behind UTC, the function will

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

return the following results:

```
wwStringFromTime(57600, 1) will return "1/2/70"
wwStringFromTime(57600, 2) will return "12:00:00 AM"
wwStringFromTime(57600, 3) will return "Fri Jan 02 00:00:00 1970"
wwStringFromTime(57600, 4) will return "Fri"
wwStringFromTime(57600, 5) will return "Friday"
```

Security

Security functions use the security namespace. There is currently one security function, GetNamedCredentials().

GetNamedCredentials()

Returns the username and password for the given named credential.

Category

Security

Syntax

```
Security.GetNamedCredentials("CredentialName");
```

where *CredentialName* is the name of the credential whose value you are retreiving.

Return value

The GetNamedCredentials() script function returns the the username and password for a named credential. You can specify if the username and password are returned as a string or as a SecureString.

Note: GetNamedCredentials() does not let you retrieve the domain name, if the credential you are retrieving includes domain.

Example

```
dim stCredentialName as string;
dim stUsername as string;
dim stPassword as System.Security.SecureString;
dim credentials as aaSecurity.Credentials;
stCredentialName = "cred1";
credentials = GetNamedCredentials(stCredentialName);
if(credentials.Values.Count > 0) then
if(credentials.ContainsKey("Username")) then
       stUsername = credentials.Get("Username");
   endif;
if(credentials.ContainsKey("Password")) then
       dim networkcred as System.Net.NetworkCredential;
       networkcred = new System.Net.NetworkCredential(stUsername,
       credentials.Get("Password"));
       stPassword = networkcred.SecurePassword;
```

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

```
   endif;
endif;
```

Note: Use “stpassword” SecureString as required.

System functions

Use system functions to interact with the operating system or other core system functions, such as ActiveX objects.

CreateObject()

Creates an ActiveX (COM) object.

Category

System

Syntax

```
ObjectResult = CreateObject( ProgID );
```

Parameter

ProgID
The program ID (as a string) of the object to be created.

Example

```
CreateObject("ADODB.Connection");
```

Now()

Returns the current time.

Category

System

Syntax

```
TimeValue = Now();
```

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Remarks

The return value can be formatted using .NET functions.

WWDDE functions

Use WWDDE functions when working with the DDE protocol.

WWExecute()

Using the DDE protocol, executes a command to a specified application and topic and returns the status.

Category

WWDDE

Syntax

```
Status = WWExecute( Application, Topic, Command );
```

Parameters

Application

The application to which you want to send an execute command. Actual string or a string attribute.

Topic

The topic within the application. Actual string or a string attribute.

Command

The command to send. Actual string or a string attribute.

Return value

Status is an Integer attribute to which 1, -1, or 0 is written. The WWExecute() function returns 1 if the application is running, the topic exists, and the command was sent successfully. It returns 0 when the application is busy, and -1 when there is an error.

Remarks

Note: The three WWDDE functions Execute(), Poke() and Request() exist for legacy purposes.

The *Command* string is sent to a specified application and topic.

Important: The following applies to using WWExecute() in synchronous scripts:
1. Never loop them (call them over and over).
2. Never call several of them in a row and in the same script.
3. Never use them to call a lengthy task in another DDE application.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

All three actions, though, are appropriate in asynchronous scripts.

Examples

The following statement executes a macro in Excel:

```
Macro="Macro1!TestMacro";
Command="[Run(" + StringChar(34) + Macro + StringChar(34)
+ ",0)]";
WWExecute("excel","system",Command);
```

When `WWExecute("excel","system",Command);` is processed, the following is sent to Excel (and *TestMacro* runs):

```
[Run("Macro1!TestMacro")];
```

The following script executes a macro in Microsoft Access:

```
WWExecute("MSAccess","system","MyMacro");
```

WWPoke()

Using the DDE protocol, pokes a value to a specified application, topic, and item and returns the status.

Category

WWDDE

Syntax

```
Status = WWPoke( Application, Topic, Item, TextValue );
```

Parameters

Application

The application to which you want to send the Poke command. Actual string or a string attribute.

Topic

The topic within the application. Actual string or a string attribute.

Item

The item to poke within the topic. Actual string or a string attribute.

TextValue

The value to poke. If the value you want to send is a number, you can convert it using the Text(), StringFromIntg(), or StringFromReal() functions. Actual string or a string attribute.

Return value

Status is an Integer attribute to which 1, -1, or 0 is written. The WWPoke() function returns 1 if the application is running, the topic and item exist, and the value was sent successfully. It returns 0 if the application is busy, and -1 if there is an error.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Remarks

Note: The three WWDDE functions Execute(), Poke() and Request() exist for legacy purposes.

The value *TextValue* is sent to the particular application, topic, and item specified.

Important: The following applies to using WWRequest() in synchronous scripts:
1. Never loop them (call them over and over).
2. Never call several of them in a row and in the same script.
3. Never use them to call a lengthy task in another DDE application. All three actions, though, are appropriate in asynchronous scripts.

Example

The following statement converts a value to text and pokes the result to an Excel spreadsheet cell:

```
String=Text(Value,"0");
WWPoke("excel","[Book1.xls]sheet1","r1c1",String);
```

The behavior for WWPoke() from within the application "View" to "View" is undefined and is not supported. The WWPoke() command is not guaranteed to succeed in this instance, and the command will probably time-out without the desired results.

See also

StringFromIntg()

StringFromReal()

Text()

WWRequest()

Using the DDE protocol, makes a one-time request for a value from a particular application, topic, and item and returns the status.

Category

WWDDE

Syntax

```
Status = WWRequest( Application, Topic, Item, Attribute );
```

Parameters

Application

The application from which you want to request data. Actual string or a string attribute.

Topic

The topic within the application. Actual string or a string attribute.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Item

The item within the topic. Actual string or a string attribute.

Attribute

A string attribute, enclosed in quotation marks, that contains the requested value from the application, topic, and item. Actual string or a string attribute.

Return value

Status is an integer attribute to which 1, -1, or 0 is written. The WWRequest() function returns 1 if the application is running, the topic and item exist, and the value was returned successfully. It returns 0 if the application is busy, and -1 if there is an error.

Remarks

Note: The three WWDDE functions Execute(), Poke() and Request() exist for legacy purposes.

The DDE value in the particular application, topic, and item is returned into *Attribute*.

The value is returned as a string into a string attribute. If the value is a number, you can then convert it using the StringToIntg() or StringToReal() functions.

Important: Never do the following when using WWRequest() in synchronous scripts:
1. Loop scripts (call them over and over).
2. Call several of scripts in a row and in the same script.
3. Use scripts to call a lengthy task in another DDE application.
All three actions can be done in asynchronous scripts.

Example

The following statement requests a value from an Excel spreadsheet cell and converts the resulting string into a value:

```
WWRequest("excel","[Book1.xls]sheet1","r1c1",Result);
Value=StringToReal(Result);
```

See also

StringToIntg()

StringToReal()

QuickScript .NET operators

The following QuickScript .NET operators require a single operand:

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Operator	Short description
~	Complement
-	Negation
NOT	Logical NOT

The following QuickScript .NET operators require two operands:

Operator	Short description
+	Addition and concatenation
-	Subtraction
&	Bitwise AND
*	Multiplication
**	Power
/	Division
^	Exclusive OR
|	Inclusive OR
<	Less than
<=	Less than or equal to
<>	Not equal to
=	Assignment
==	Equivalency (is equivalent to); not supported for entire array compares. Compare the arrays one element at a time using ==.
>	Greater than
>=	Greater than or equal to
AND	Logical AND
MOD	Modulo
OR	Logical OR
SHL	Left shift
SHR	Right shift

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

The following table shows the precedence of QuickScript .NET operators:

Precedence	Operator
1 (highest)	**( )**
2	- (negation), NOT, ~
3	**
4	*, /, MOD
5	+, - (subtraction)
6	SHL, SHR
7	<, >, <=, >=
8	==, <>
9	&
10	^
11	|
12	=
13	AND
14 (lowest)	OR

The arguments of the listed operators can be numbers or attribute values. Putting parentheses around an argument is optional. Operator names are not case-sensitive.

Parentheses ( )

Parentheses specify the correct order of evaluation for the operator(s). They can also make a complex expression easier to read. Operator(s) in parentheses are evaluated first, preempting the other rules of precedence that apply in the absence of parentheses. If the precedence is in question or needs to be overridden, use parentheses.

In the example below, parentheses add B and C together before multiplying by D:

```
( B + C ) * D;
```

Negation (-)

Negation is an operator that acts on a single component. It converts a positive integer or real number into a negative number.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Complement (~)

This operator yields the one's complement of a 32-bit integer. It converts each zero-bit to a one-bit and each one-bit to a zero-bit. The one's complement operator is an operator that acts on a single component, and it accepts an integer operand.

Power (**)

The Power operator returns the result of a number (the base) raised to the power of a second number (the power). The base and the power can be any real or integer numbers, subject to the following restrictions:

- A zero base and a negative power are invalid.

  Example: "0 ** - 2" and "0 ** -2.5"
- A negative base and a fractional power are invalid.

  Example: "-2 ** 2.5" and "-2 ** -2.5"
- Invalid operands yield a zero result.

The result of the operation cannot be so large or so small that it cannot be represented as a real number. Example:

```
1 ** 1 = 1.0
3 ** 2 = 9.0
10 ** 5 = 100,000.0
```

Multiplication (*), division (/), addition (+),subtraction (-)

These binary operators perform basic mathematical operations. The plus (+) can also concatenate String datatypes.

For example, in the data change script below, each time the value of "Number" changes, "Setpoint" changes as well:

```
Number=1;
Setpoint.Name = "Setpoint" + Text(Number, "#" );
```

Where: The result is "`Setpoint1`."

Modulo (MOD)

MOD is a binary operator that divides an integer quantity to its left by an integer quantity to its right. The remainder of the quotient is the result of the MOD operation. Example:

```
97 MOD 8 yields 1
63 MOD 5 yields 3
```

Shift left (SHL), shift right (SHR)

SHL and SHR are binary operators that use only integer operands. The binary content of the 32-bit word referenced by the quantity to the left of the operator is shifted (right or left) by the number of bit positions

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

specified in the quantity to the right of the operator.

Bits shifted out of the word are lost. Bit positions vacated by the shift are zero-filled. The shift is an unsigned shift.

Example 1

If `Attribute2 = 00000111` (decimal 7)

Then the operation:

```
Attribute1 = Attribute2 SHL 3;
```

Results in:

`Attribute1 = 00111000` (decimal 56)

Example 2

If `Attribute2 = 00001011` (decimal 11)

Then the operation:

```
Attribute1 = Attribute2 SHR 2;
```

Results in:

`Attribute1 = 00000010` (decimal 2)

Bitwise AND (&)

A bitwise binary operator compares 32-bit integer words with each other, bit for bit. Typically, this operator masks a set of bits. The operation in this example "masks out" (sets to zero) the upper 24 bits of the 32-bit word. For example:

```
result = name & 0xff;
```

Exclusive OR (^) and inclusive OR (|)

The ORs are bitwise logical operators compare 32-bit integer words to each other, bit for bit. The Exclusive OR compare the status of bits in corresponding locations. If the corresponding bits are the same, a zero is the result. If the corresponding bits differ, a one is the result. Example:

```
0 ^ 0 yields 0
0 ^ 1 yields 1
1 ^ 0 yields 1
1 ^ 1 yields 0
```

The Inclusive OR examines the corresponding bits for a one condition. If either bit is a one, the result is a one. Only when both corresponding bits are zeros is the result a zero. For example:

```
0 | 0 yields 0
0 | 1 yields 1
1 | 0 yields 1
1 | 1 yields 1
```

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Assignment (=)

Assignment is a binary operator which accepts integer, real, or any type of operand. Each statement can contain only one assignment operator. Only one name can be on the left side of the assignment operator.

Read the equal sign (=) of the assignment operator as "is assigned to" or "is set to."

Don't confuse the equal sign with the equivalency sign (==) used in comparisons.

Comparisons (<, >, <=, >=, ==, <>)

Comparisons in IF-THEN-ELSE statements execute various instructions based on the state of an expression.

AND, OR, and NOT

These operators work only on discrete attributes. If these operators are used on integers or real numbers, they are converted as follows:

- Real to Discrete: If real is 0.0, discrete is 0, otherwise discrete is 1.
- Integer to Discrete: If integer is 0, discrete is 0, otherwise discrete is 1.

If the statement is: "`Disc1 = Real1 AND Real2;`" and `Real1` is 23.7 and `Real2` is 0.0, `Disc1` has 0 assigned to it, since `Real1` is converted to 1 and `Real2` is converted to 0.

When assigning the floating-point result of a mathematical operation to an integer, the value is rounded to the nearest integer instead of truncating it. This means that an operation like `IntAttr = 32/60` results in `IntAttr` having a value of 1, not 0. If truncation is needed, use the `Trunc()` function.

QuickScript .NET variables

Declare the QuickScript .NET variables before they can be used in QuickScript .NET scripts. Variables can be used on both the left and right side of statements and expressions.

Local variables or attributes can be used together in the same script. Variables declared within the script body lose their value after the script is executed. Those declared in the script body cannot be accessed by other scripts.

Variables declared in the **Declarations** area maintain their values throughout the lifetime of the object that the script is associated with.

Declare each variable in the script by a separate DIM statement followed by a semicolon. Enter DIM statements in the **Declarations** area of the **Script** tab page. The DIM statement syntax is as follows:

```
DIM <variable_name> [ ( <upper_bound>
[, <upper_bound >[, < upper_bound >]] ) ]
[ AS <data_type> ];
```

where:

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

DIM	Required keyword.
<variable_name>	Name that begins with a letter (A-Z or a-z) and whose remaining characters can be any combination of letters (A-Z or a-z), digits (0-9) and underscores (_). The variable name is limited to 255 Unicode characters.
<upper_bound>	Reference to the upper bound (a number between 1 and 2,147,483,647, inclusive) of an array dimension. Three dimensions are supported in a DIM statement, each being nested in the syntax structure. After the upper bound is specified, it is fixed after the declaration. A statement similar to Visual Basic's ReDim is not supported. The lower bound of each array dimension is always 1.
AS	Optional keyword for declaring the variable's datatype.

<data_type>	Any one of the following 11 datatypes: Boolean, Discrete, Integer, ElapsedTime, Float, Real, Double, String, Message, Time or Object.
	
	Data_type can also be a .Net data_type like System.Xml.XmlDocument or a type defined in an imported script library
	
	If you omit the AS clause from the DIM statement, the variable, by default, is declared as an Integer datatype. For example:
	
	`DIM LocVar1;`
	
	is equivalent to:
	
	`DIM LocVar1 AS Integer;`

In contrast to attribute names, variable names must not contain dots. Variable names and the data type identifiers are not case sensitive. If there is a naming conflict between a declared variable and another named entity in the script (for example, attribute name, alias or name of an object leveraged by the script), the variable name takes precedence over the other named entities. If the variable name is the same as an alias name, a warning message appears when the script is validated to indicate that the alias is ignored.

The syntax for specifying the entire array is "[ ]" for both local array variables and for attribute references. For example, to assign an attribute array to a local array, the syntax is:

```
locarr[] = tag.attr[];
```

DIM statements can be located anywhere in the script body, but they have to precede the first referencing script statement or expression. If a local variable is referenced before the DIM statement, script validation done when

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

you save the object containing the script prompts you to define it.

The validation mentioned above occurs only when you save the object containing the script. This is not the script syntax validation done when you click the **Validate Script** button.

Don't cascade DIM statements. For example, the following examples are invalid:

```
DIM LocVar1 AS Integer, LocVar2 AS Real;
DIM LocVar3, LocVar4, LocVar5, AS Message;
```

To declare multiple variables, enter separate `DIM` statements for each variable.

When used on the right side of an equation, declared local variables always cause expressions on the left side to have Good quality. For example :

```
dim x as integer;
dim y as integer;
x = 5;
y = 5;
me.attr = 5;
me.attr = x;
me.attr = x+y;
```

In each case of `me.attr`, quality is Good.

When you use a variable in an expression to the right of the operator, its Quality is treated as Good for the purpose of data quality propagation.

You can use null to indicate that there is no object currently assigned to a variable. Using null has the same meaning as the keyword "null" in C# or "nothing" in Visual Basic. Assigning null to a variable makes the variable eligible for garbage collection. You may not use a variable whose value is null. If you do, the script terminates and an error message appears in the logger. You may, however, test a variable for null. For example:

```
IF myvar == null THEN ...
```

It is not possible to pass attributes as parameters for system objects. To work around this issue, use a local variable as an intermediary or explicitly convert the attribute to a string using an appropriate function call when calling the system object.

Numbers and strings

Allowed format for integer constants in decimal format is as follows:

```
IntegerConst = 0 or [sign] <non-zero_digit> <digit>*;
```

where:

```
sign :: = + | -
non-zero_digit ::= 1-9
digit ::= 0-9
```

For example, an integer constant is a zero or consists of an optional sign followed by one or more digits. Leading zeros are not allowed. Integer constants outside the range –2147483648 to 2147483647 cause an overflow error.

Prepending either 0x or 0X causes a literal integer constant to be interpreted as hexadecimal notation. The +/- sign is supported.

The acceptable float for integers in hexadecimal is as follows:

```
IntegerHexConst = [<sign>] <0><x (or X)> <hexdigit>*
```

where:

`sign` ::= + or -

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

hexdigit ::= 0-9, A-F, a-f (only eight hexdigits [32-bits] are allowed)

Allowed format for floats is as follows:

```
FloatConst ::= [<sign>] <digit>* .<digit>+ [<exponent>;]
```

or

```
[<sign>] <digit>+ [.<digit>* [<exponent>]];
```

where:

```
sign ::= + or -
digit ::= 0-9 (can be one or more decimal digits)
exponent = e (or E) followed by a sign and then digit(s)
```

Float constants are applicable as values for variables of type float, real, or double. For example, float constants don't take the number of bytes into account. Script validation detects an overflow when a float, real, or double variable has been assigned a float constant that exceeds the maximum value.

If no digits appear before the period (.), at least one has to appear after it. If neither an exponent part nor the period appears, a period is assumed to follow the last digit in the string.

If an attribute reference exists that has a format similar to a float constant with an exponent (such as "5E3"), then use the Attribute qualifier, as follows:

```
Attribute("5E3")
```

Strings have to be surrounded by double quotation marks. They are referred to as quoted strings. The double-double quote indicates a single double-quote in the string. For example, the string:

```
Joe said, "Look at that."
```

can be represented in QuickScript .NET as:

```
"Joe said, ""Look at that."""
```

QuickScript .NET control structures

QuickScript .NET provides five primary control structures in the scripting environment:

- IF ... THEN ... ELSEIF ... ELSE ... ENDIF
- FOR ... TO ... STEP ... NEXT loop
- FOR EACH ... IN ... NEXT
- TRY ... CATCH
- WHILE loop

IF ... THEN ... ELSEIF ... ELSE ... ENDIF

IF-THEN-ELSE-ENDIF conditionally executes various instructions based on the state of an expression. The syntax is as follows:

```
IF <Boolean_expression> THEN
[statements];
[ { ELSEIF
[statements] } ];
[ ELSE
[statements] ];
ENDIF;
```

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Where Boolean_expression is an expression that can be evaluated as a Boolean.

Depending on the data type returned by the expression, the expression is evaluated to constitute a True or False state according to the following table:

Data Type	Mapping
Boolean, Discrete	Directly used (no mapping needed).
Integer	Value = 0 evaluated as False. Value != 0 evaluated as True.
Float, Real	Value = 0 evaluated as False. Value != 0 evaluated as True.
Double	Value = 0 evaluated as False. Value != 0 evaluated as True.
String, Message	Cannot be mapped. Using an expression that results in a string type as the Boolean_expression results in a script validation error.
Time	Cannot be mapped. Using an expression that results in a time type as the Boolean_expression results in a script validation error.
ElapsedTime	Cannot be mapped. Using an expression that results in an elapsed time type as the Boolean_expression results in a script validation error.
Object	Using an expression that results in an object type. Validates, but at run time, the object is converted to a Boolean. If the type cannot be converted to a Boolean, a run-time exception is raised.

The first block of statements is executed if Boolean_expression evaluates to True. Optionally, a second block of statements can be defined after the keyword ELSE. This block is executed if the Boolean_expression evaluates to False.

To help decide between multiple alternatives, an optional ELSEIF clause can be used as often as needed. The ELSEIF clause mimics switch statements offered by other programming languages. For example:

```
IF value == 0 Then
Message = "Value is zero";
ELSEIF value > 0 Then
Message = "Value is positive";
ELSEIF value < 0 Then
Message = "Value is negative";
ELSE
{Default. Should never occur in this example};
ENDIF;
```

The following approach nests a second IF compound statement within a previous one and requires an additional ENDIF:

```
IF (X1 == 1) THEN
```

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

```
X1 = 5;
{ ELSEIF <X1 == 2> THEN
X1 = 10;
ELSEIF X1 == 3 THEN
X1 = 20 ;
ELSEIF X1 == 4 THEN
X1 = 30 };
IF X1 == 99 THEN
X1 = 0;
ENDIF;
ENDIF;
```

See Sample Scripts for more ideas about using this type of control structure.

IF … THEN … ELSEIF … ELSE … ENDIF and attribute quality

When an attribute value is copied to another attribute of the same type, the attribute's quality is also copied. This can be especially relevant when working with I/O attributes. For example, the following two statements copy both value and quality:

```
me.Attr2 = me.Attr1;
me.Attr2.value = me.Attr1.value;
```

If only the value needs to be copied and the attribute has the quality BAD, you can use a temporary variable to hold the value. For example:

```
Dim temp as Integer;
temp = me.Attr1;
me.Attr2 = temp;
```

If there is a comparison such as Attr1 <> Attr2 and one of the attributes has the quality BAD, then the statements within the IF control block are not executed. For example, assuming Attr1 has the quality BAD:

```
if me.Attr1<> me.Attr2 then
   me.Attr2 = me.Attr1;
endif;
```

In this script, the statement `me.Attr2 = me.Attr1` is not executed because Attr1 has the quality BAD and comparing a BAD quality value with a good quality value is not defined/not possible.

The recommended approach is to first verify the quality of Attr1, as shown in the following example:

```
if(IsBad(me.Attr1)) then
LogMessage("Attr1 quality is bad, its value is not copied to Attr2");
else
if me.Attr1<> me.Attr2 then
me.AttrA2 = me.Attr1;
endif;
endif;
```

An alternative method of verifying quality is to use the "==" operator:

```
if Me.Attr1 == TRUE then
```

Or, you can add the "value" property to the simplified IF THEN statement:

```
if Me.Attr1.value then
```

Your scripts will execute correctly if you verify the data quality using any of the above methods.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

FOR … TO … STEP … NEXT loop

FOR-NEXT performs a function (or set of functions) within a script several times during a single execution of a script. The general format of the FOR-NEXT loop is as follows:

```
FOR <analog_var> = <start_expression> TO <end_expression> [STEP <change_expression>];
[statements];
[EXIT FOR;];
[statements];
NEXT;
```

Where:

- analog_var is a variable of type Integer, Float, Real, or Double.
- start_expression is a valid expression to initialize analog_var to a value for execution of the loop.
- end_expression is a valid expression. If analog_var is greater than end_expression, execution of the script jumps to the statement immediately following the NEXT statement.

  This holds true if loop is incrementing up, otherwise, if loop is decrementing, loop termination occurs if analog_var is less than end_expression.
- change_expression is an expression that defines the increment or decrement value of analog_var after execution of the NEXT statement. The change_expression can be either positive or negative.
  - If change_expression is positive, start_expression has to be less than or equal to end_expression or the statements in the loop don't execute.
  - If change_expression is negative, start_expression has to be greater than or equal to end_expression for the body of the loop to be executed.
- If STEP is not set, then change_expression defaults to 1 for increasing increments, and defaults to -1 for decreasing increments.

Exit the loop from within the body of the loop with the EXIT FOR statement.

The FOR loop is executed as follows:

1. analog_var is set equal to start_expression.
2. If change_expression is positive, the system tests to see if analog_var is greater than end_expression. If so, the loop exits. If change_expression is negative, the system tests to see if analog_var is less than end_expression. If so, program execution exits the loop.
3. The statements in the body of the loop are executed. The loop can potentially be exited via the EXIT FOR statement.
4. analog_var is incremented by 1,-1, or by change_expression if it is specified.
5. Steps 2 through 4 are repeated.

FOR-NEXT loops can be nested. The number of levels of nesting possible depends on memory and resource availability.

FOR EACH … IN … NEXT

FOR EACH loops can be used only with collections exposed by OLE Automation servers. A FOR-EACH loop performs a function (or set of functions) within a script several times during a single execution of a script. The

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

general format of the FOR-EACH loop is as follows:

```
FOR EACH <object_variable> IN <collection_object >
   [statements];
   [EXIT FOR;];
   [statements];
NEXT;
```

Where:

- object_variable is a dimmed variable.
- collection_object is a variable holding a collection object.

As in the case of the FOR … TO loop, it is possible to exit the execution of the loop through the statement EXIT FOR from within the loop.

TRY … CATCH

TRY ... CATCH provides a way to handle some or all possible errors that may occur in a given block of code, while still running rather than terminating the program. The TRY part of the code is known as the try block. Deal with any exceptions in the CATCH part of the code, known as the catch block.

The general format for TRY ... CATCH is as follows:

```
TRY
   [try statements] ’guarded section
CATCH
   [catch statements]
ENDTRY
```

Where:

tryStatements

Statement(s) where an error can occur. Can be a compound statement. The tryStatement is a guarded section.

catchStatements

Statement(s) to handle errors occurring in the associated Try block. Can be a compound statement.

Statements inside the Catch block may reference the reserved ERROR variable, which is a .NET System.Exception thrown from the Try block. The statements in the Catch block run only if an exception is thrown from the Try block.

TRY ... CATCH is executed as follows:

1. Run-time error handling starts with TRY. Put code that might result in an error in the try block.
2. If no run-time error occurs, the script will run as usual. Catch block statements will be ignored.
3. If a run-time error occurs, the rest of the try block does not execute.
4. When a run-time error occurs, the program immediately jumps to the CATCH statement and executes the catch block.

   The simplest kind of exception handling is to stop the program, write out the exception message, and continue the program.

   The error variable is not a string, but a .NET object of System.Exception. This means you can determine the type of exception, even with a simple CATCH statement. Call the GetType() method to determine the exception type, and then perform the operation you want, similar to executing multiple catch blocks.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Example:

```
dim command = new System.Data.SqlClient.SqlCommand;
dim reader as System.Data.SqlClient.SqlDataReader;
command.Connection = new System.Data.SqlClient.SqlConnection;
try
   command.Connection.ConnectionString = "Integrated Security=SSPI";
   command.CommandText="select * from sys.databases";
   command.Connection.Open();
   reader = command.ExecuteReader();
   while reader.Read()
      me.name = reader.GetString(0);
      LogMessage(me.name);
   endWhile;
catch
   LogMessage(error);
endtry;
if reader <> null and not reader.IsClosed then
   reader.Close();
endif;
if command.Connection.State == System.Data.ConnectionState.Open then
   command.Connection.Close();
endif;
```

WHILE loop

WHILE loop performs a function or set of functions within a script several times during a single execution of a script while a condition is true. The general format of the WHILE loop is as follows:

```
WHILE <Boolean_expression>
[statements]
[EXIT WHILE;]
[statements]
ENDWHILE;
```

Where: Boolean_expression is an expression that can be evaluated as a Boolean as defined in the description of IF…THEN statements.

It is possible to exit the loop from the body of the loop through the EXIT WHILE statement.

The WHILE loop is executed as follows:

1. The script evaluates whether the Boolean_expression is true or not. If not, program execution exits the loop and continues after the ENDWHILE statement.
2. The statements in the body of the loop are executed. The loop can be exited through the EXIT WHILE statement.
3. Steps 1 through 2 are repeated.

WHILE loops can be nested. The number of levels of nesting possible depends on memory and resource availability.

Sample QuickScript .NET scripts

This section includes sample scripts to help you to understand the QuickScript .NET scripting language.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Important Notes: The sample scripts provided with a number of the Application Server scripting functions should work as written in most Windows operating system and installed software environments, but might not work with all possible hardware, operating system, and software combinations. We recommend that you modify the example scripts as necessary to fit your specific environment.

Some sample scripts include references to public websites as examples. You may need to replace those URLs with a current and verified URLs.

Use an imported type library to access an Excel spreadsheet

The purpose of this script is to write data to an open Excel spreadsheet.

Before using this script, you must first:

1. Import the Microsoft Office Excel dll (**Microsoft.Office.Interop.Excel.dll**) to create the required namespace. From the **Galaxy** menu, select **Import**, then **Script Function Library**.

   Note: The Microsoft Office Excel dll file name and location may vary, depending on which version of Excel is installed.

2. Open the Excel spreadsheet you want to access. In the following sample script, the Excel file name and path are: C:\documents\sample.xlsx

```
dim app as object;
dim wb as object;
dim ws as object;
app = CreateObject("Excel.Application");
app.Visible=true;
wb = app.Workbooks.Open("C:\documents\sample.xlsx");
ws = wb.ActiveSheet;
ws.Range("A1").Value = 100;
ws.Range("A2").Value = 200;
ws.Range("A3").Value = "=A1*A2";
LogMessage(ws.Range("A3").Value);
wb.Visible=true;
wb.Close(false);
```

Use CreateObject to access an Excel spreadsheet

```
dim app as object;
dim wb as object;
dim ws as object;
app = CreateObject("Excel.Application");
wb = app.Workbooks.Add();
ws = wb.ActiveSheet;
ws.Range.("A1").value = 20;
ws.Range.("A2").value = 30;
ws.Range.("A3").value = "=A1*A2";
LogMessage(ws.Range("A3").Value);
wb.Close(false);
```

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Call a web service to get the temperature in a Zip code

Note: This sample script includes a reference to a public website as an example. You may need to replace that URL with a current and verified URL.

```
' Requires input string uda me.zipcode and output float uda me.temperature.
' First, generate a wrapper for the web service (.Net SDK must be installed).
' To generate wrapper, run the following commands from the DOS prompt:
' set path=%path%;C:\Program Files\Microsoft Visual Studio .NET\FrameworkSDK\Bin
' wsdl http://www.vbws.com/services/weatherretriever.asmx
' csc /target:library WeatherRetriever.cs
' Next import the generated WeatherRetriever.dll library into your galaxy.
' Now write your script:
dim wr as WeatherRetriever;
wr = new WeatherRetriever;
me.temperature = wr.GetTemperature(me.zipcode);
```

Call a web service to send an e-mail

Note: This sample script includes a reference to a public website as an example. You may need to replace that URL with a current and verified URL.

```
' First, generate a wrapper for the web service (.Net SDK must be installed).
' To generate wrapper, run the following commands from the DOS prompt:
' set path=%path%;C:\Program Files\Microsoft Visual Studio .NET\FrameworkSDK\Bin
' wsdl /namespace:SendMail http://www.xml-webservices.net/services/messaging/smtp_mail/
mailsender.asmx
' csc /target:library Message.cs
' Next import the generated Message.dll library into your galaxy.
' Now write your script:
dim m as SendMail.Message;
m = new SendMail.Message;
m.SendSimpleMail
(
{to: } "<type valid email address here>",
{from: } "<type valid email address here>",
{subject: } "Reminder to self",
{body: } "Pick up eggs and milk on your way home."
);
```

Create a look-up table and do a look-up on it

```
dim zipcodes as System.Collections.Hashtable;
zipcodes = new System.Collections.Hashtable;
zipcodes["Irvine"] = 92618;
zipcodes["Mission Viejo"] = 92692;
LogMessage(zipcodes["Irvine"]);
```

Create an XML document and save it to disk

```
dim doc as System.Xml.XmlDocument;
```

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

```
dim catalog as System.Xml.XmlElement;
dim book as System.Xml.XmlElement;
dim title as System.Xml.XmlElement;
dim author as System.Xml.XmlElement;
dim lastName as System.Xml.XmlElement;
dim firstName as System.Xml.XmlElement;
' create new XML document rooted in catalog
doc = new System.Xml.XmlDocument;
catalog = doc.CreateElement("catalog");
doc.AppendChild(catalog);
' add a book to the catalog
book = doc.CreateElement("book");
title = doc.CreateElement("title");
author = doc.CreateElement("author");
lastName = doc.CreateElement("lastName");
firstName = doc.CreateElement("firstName");
author.AppendChild(lastName);
author.AppendChild(firstName);
book.AppendChild(title);
book.AppendChild(author);
catalog.AppendChild(book);
book.SetAttribute("isbn", "0385503822");
title.InnerText = "The Summons";
lastName.InnerText = "Grisham";
firstName.InnerText = "John";
' add another book
book = doc.CreateElement("book");
title = doc.CreateElement("title");
author = doc.CreateElement("author");
lastName = doc.CreateElement("lastName");
firstName = doc.CreateElement("firstName");
author.AppendChild(lastName);
author.AppendChild(firstName);
book.AppendChild(title);
book.AppendChild(author);
catalog.AppendChild(book);
book.SetAttribute("isbn", "044023722X");
title.InnerText = "A Painted House";
lastName.InnerText = "Grisham";
firstName.InnerText = "John";
' save the XML document to disk
doc.Save("c:\catalog.xml");
```

Execute a SQL parameterized INSERT command

```
dim connection as System.Data.SqlClient.SqlConnection;
dim command as System.Data.SqlClient.SqlCommand;
dim regionId as System.Data.SqlClient.SqlParameter;
dim regionDesc as System.Data.SqlClient.SqlParameter;
dim commandText as string;
connection = new
System.Data.SqlClient.SqlConnection("server=(local);uid=sa;database=northwind");
connection.Open();
commandText = "INSERT INTO Region (RegionID, RegionDescription) VALUES (@id, @desc)";
command = new System.Data.SqlClient.SqlCommand(commandText, connection);
```

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

```
regionId = command.Parameters.Add("@id", System.Data.SqlDbType.Int, 4);
regionDesc = command.Parameters.Add("@desc", System.Data.SqlDbType.NChar, 50);
command.Prepare();
regionId.Value = 5;
regionDesc.Value = "Europe";
command.ExecuteNonQuery();
regionId.Value = 6;
regionDesc.Value = "South America";
command.ExecuteNonQuery();
connection.Close();
```

Fill and use a string array

```
dim numbers[3] as string;
dim s as string;
numbers[1] = "one";
numbers[2] = "two";
numbers[3] = "three";
LogMessage(numbers[3]);
for each s in numbers[]
LogMessage(s);
next;
```

Fill and use a two-dimensional integer array

```
dim x[2,3] as integer;
dim i as integer;
x[1, 1] = 1;
x[1, 2] = 2;
x[1, 3] = 3;
x[2, 1] = 4;
x[2, 2] = 5;
x[2, 3] = 6;
LogMessage(x[2, 3]);
for each i in x[]
LogMessage(i);
next;
```

Format a Number with a .NET format 'Picture'

```
dim i as integer;
i = 1234;
LogMessage("Total cost: " + i.ToString("$#,###,###.00"));
```

Format a time with a .NET format 'Picture'

```
dim t as time;
t = Now();
LogMessage("The current time is: " + t.ToString("hh:mm:ss") + ".");
```

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Get the directories under the C drive

```
dim dir as System.IO.DirectoryInfo;
for each dir in System.IO.DirectoryInfo("c:\").GetDirectories()
LogMessage(dir.FullName);
next;
```

Load an XML Document from disk and do look-ups on it

```
dim doc as System.Xml.XmlDocument;
dim node as System.Xml.XmlNode;
doc = new System.Xml.XmlDocument;
doc.Load("c:\catalog.xml");
' find the title of the book whose isbn is 044023722X
node = doc.SelectSingleNode("/catalog/book[@isbn='044023722X']/title");
LogMessage(node.InnerText);
' find all titles written by Grisham
for each node in doc.SelectNodes("/catalog/book[author/lastName='Grisham']/title")
   LogMessage(node.InnerText);
next;
```

Query a SQL Server database

```
dim connection as System.Data.SqlClient.SqlConnection;
dim command as System.Data.SqlClient.SqlCommand;
dim reader as System.Data.SqlClient.SqlDataReader;
connection = new
System.Data.SqlClient.SqlConnection("server=(local);uid=sa;database=northwind");
connection.Open();
command = new System.Data.SqlClient.SqlCommand("select * from customers", connection);
reader = command.ExecuteReader();
while reader.Read()
   LogMessage(reader("CompanyName"));
endwhile;
reader.Close();
connection.Close();
```

Read a performance counter

```
' Requires output float UDA me.PercentProcessorTime.
' Declarations
dim counter as System.Diagnostics.PerformanceCounter;
' Startup
counter = new System.Diagnostics.PerformanceCounter;
counter.CategoryName = "Processor";
counter.CounterName = "% Processor Time";
counter.InstanceName = "0";
' Execute
me.PercentProcessorTime = counter.NextValue();
```

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Read a text file from disk

```
dim sr as System.IO.StreamReader;
sr = System.IO.File.OpenText("c:\MyFile.txt");
while sr.Peek() > -1
   LogMessage(sr.ReadLine());
endwhile;
sr.Close();
```

Share a SQL connection or other .NET object

In UserDefined_001 do this:

```
dim connection as System.Data.SqlClient.SqlConnection;
’ Startup
connection = new
System.Data.SqlClient.SqlConnection("server=(local);uid=sa;database=northwind");
connection.Open();
System.AppDomain.CurrentDomain.SetData
("NorthwindConnection", connection);
’ Shutdown
System.AppDomain.CurrentDomain.SetData("NorthwindConnection", Null);
connection.Close();
```

Then in UserDefined_002, UserDefined_003, and so on, do this:

```
dim connection as System.Data.SqlClient.SqlConnection;
connection = System.AppDomain.CurrentDomain.GetData
("NorthwindConnection");
if connection <> null then
System.Threading.Monitor.Enter(connection);
' use the connection
System.Threading.Monitor.Exit(connection);
endif;
```

Use DDE to access an Excel spreadsheet

```
WWPoke("excel", "sheet1", "r1c1", "Hello");
WWRequest("excel", "sheet1", "r1c1", me.Greeting);
' Note: use "" to embed double quotation marks in strings
WWExecute("excel", "sheet1", "[SELECT(""R1C1"")][FONT.PROPERTIES(,""Bold"")]");
```

Use Microsoft Exchange to send an e-mail message

The following compatibility notes apply to use of this script:

- The script will not work on Microsoft Office 2010 or later due to changes in MAPI handling.
- The script is supported only for 32-bit versions of Microsoft Office.
- If you are using Microsoft Office 2007, you must download and install Microsoft Collaboration Data Objects (CDO) 1.2.1. Additional information on CDO 1.2.1 is available at: http://www.microsoft.com/en-us/download/details.aspx?id=3671

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

```
dim session as object;
dim msg as object;
dim sProfileInfo as string;
sProfileInfo = "<type valid Microsoft Exchange Server Name here>" + StringChar(10) +
"<type valid Exchange Server user name here>";
session = CreateObject("MAPI.Session");
session.Logon(, , False,False , , True, sProfileInfo);
msg = session.Outbox.Messages.Add();
msg.Recipients.Add("<type valid email address here>");
msg.Recipients.Resolve();
msg.Subject = "Reminder to self";
msg.Text = "Pick up eggs and milk on your way home.";
msg.Send();
session.Logoff();
```

Use screen-scraping to get the temperature for a city

Note: This sample script includes a reference to a public website as an example. You may need to replace that URL with a current and verified URL.

```
' Screen-scraping involves downloading a web page,
' then using a regular expression to retrieve the desired data.
' Requires input string UDA me.CityState, e.g. "Los Angeles,CA"
' and output float UDA me.temperature.
dim request as System.Net.WebRequest;
dim reader as System.IO.StreamReader;
dim regex as System.Text.RegularExpressions.Regex;
dim match as System.Text.RegularExpressions.Match;
request = System.Net.WebRequest.Create
(
"http://www.srh.noaa.gov/data/forecasts/zipcity.php?inputstring=" +
System.Web.HttpUtility.UrlEncode(me.CityState)
);
reader = new System.IO.StreamReader(request.GetResponse().GetResponseStream());
regex = new System.Text.RegularExpressions.Regex("<br><br>(.*)&deg;F<br>");
match = regex.Match(reader.ReadToEnd());
me.temperature = match.Groups(1);
```

Use SMTP to send an e-mail message

```
System.Web.Mail.SmtpMail.Send
(
{from: } "<type valid email address here>",
{to: } "<type valid email address here>",
{subject: } "Reminder to self",
{body: } "Pick up eggs and milk on your way home."
);
```

Write a text file to disk

```
dim sw as System.IO.StreamWriter;
sw = System.IO.File.CreateText("C:\MyFile.txt");
```

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

```
sw.WriteLine("one");
sw.WriteLine("two");
sw.WriteLine("three");
sw.Close();
```

Dynamically bind an indirect variable to a reference

You can dynamically bind a variable of type Indirect to an arbitrary reference string and then use it for get/set purposes. For example:

```
' Assume reference obj1.Attr1 has value of 7
dim x as indirect;
dim s as string;
s = "obj1.Attr1";
x.BindTo(s); ' where s is any expression that returns a string.
   ' The string should be an ArchestrA reference.
obj2.Attr2 = x; ' sets obj2.Attr2 to the reference x is bound to
   ' (obj1.Attr1 in this example, which has value of 7)
x = 1234; ' sets obj1.Attr1 (in this example) to 1234
IF WriteStatus(x) == MxStatusOk THEN
   ' ... do something
endif;
```

Note: You can use .BindTo with an attribute on another engine, but this requires additional scripting of the bind to ensure good quality. For more information, see Bind to off-engine attributes.

An unbound indirect returns no data.

If the Galaxy has Advanced Communication Management enabled, we do not recommend that you use references that are part of an ActiveOnDemand DIObject scan group in a script with an Indirect. The reference activation process is not in sync with the script execution, so using a function such as the IsUseable() function always returns false.

For example, the following scripting is NOT recommended.

In the declarations section:

```
Dim pPump as Indirect;
In Script Body section
pPump.BindTo("Pump_001.State"); 'Pump_001 is part of a DIObject scan group that has
ActiveOnDemand enabled
IF IsUsable(pPump)
THEN Do this….' this will not execute
ELSE
Do that…
ENDIF;
pPump.BindTo("Pump_002.State");'Pump_002 is part of a DIObject scan group that has
ActiveOnDemand enabled
IF IsUsable(pPump)
THEN Do this….' this will not execute
ELSE
Do that…
ENDIF;
```

In the script, only Pump_002 is executing all of the time.

Important: If you have an existing application that uses the same Indirect variable with scripting more than one time for the items extended to device integration (DI) items or for DI items directly, and you enable Advanced Communication Management in the IDE, these scripts behave differently or do not execute as expected.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Bind to off-engine attributes

Reference binding is inherently an asynchronous process. This means that a reference to an attribute hosted on the same engine is immediately available, but off-engine references can require additional scan cycles to bind. In scripting binding to off-engine attributes, we recommend you check indirect variables for quality before using them.

You can use the following guidelines when binding to off-engine attributes:

- Declare the indirect variables in the declarations section of the script. This retains the value across scan cycles.
- Implement a mechanism to define different states of the script to distinguish between the normal execution cycle and waiting for bound references to resolve.
- In the execution or assignment state, use BindTo() to bind the indirect variables. After BindTo, change the state to "waiting for references" to check these in the next scan.
- In the "waiting for reference" state, use IsGood() to check the quality of the indirect variables. When all references show good quality, change to the normal execution state. The variables then are usable.
- You can implement a TimeOut state if required. For example, the remote engine might be Off Scan.

The following script examples illustrate these guidelines.

```
dim bindedRef1 as indirect;
dim bindedRef2 as indirect;
dim scriptState as integer;
' Script States
' 0 = normal execution
' 1 = wait for remote references
if (scriptState==0) then;
   ' normal script logic
   ' perform normal tasks
   ' ...
   ' for this example, change the binding to remote engine
   if (System.DateTime.Now.Second mod 10 == 0) then;
      LogMessage("Change binding (1)");
      bindedRef1.BindTo("ApplicationObject_001.SimValue01");
      bindedRef2.BindTo("ApplicationObject_001.SimValue02");
      scriptState = 1;
   endif;
   if (System DateTime.Now.Second mod 10 == 5) then;
      LogMessage("Change binding (2)");
      bindedRef1.BindTo("ApplicationObject_001.SimValue2");
      bindedRef2.BindTo("ApplicationObject_001.SimValue1");
      scriptState = 1;
   endif;
endif;
if (scriptState==1) then;
   ' wait for remote references
   ' in this example we want two valid references
   if (IsGood(bindedRef1) and IsGood(bindedRef2)) then;
      LogMessage("Binded references are good.");
      scriptState = 0;
   endif;
endif;
```

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

As an alternative, you can use a WHILE-triggered script to allow evaluation at every scan cycle. Generally, WHILE loops are not recommended, but can be used to ensure execution.

```
In Declarations:

dim x as indirect;
dim y as boolean;

In Execute (while true: me.z):

if not y then
   'This could also be done in the startup script
   x.BindTo("Object1.attribute1");
   y = true;
endif;

if IsGood(x) then
   LogMessage(x);
   me.z = false;
endif;
```

SQLData Script Library

The SQLData Script Library provides database integration using ArchestrA scripting. The SQLData Script Library provides the following benefits:

- Your resources are managed more efficiently because the connection manager reduces the number of open connections to the database provider. This activity is known as *connection pooling*.
- You can process scripts asynchronously, which reduces impact on the hosting engine during the following activities:
  - Opening a connection to a data source
  - Running SQL queries
  - Running SQL transactions

  Note: It is recommended, to avoid returning excessive amount of data using script library commands, for example executing 15,000 SQL commands which each select 200 records from the Person.Contact table in the AdventureWorks sample database. This can lead to a crash of the host engine. To avoid this problem, consider using the TOP command to limit the number of records returned.

Import and access the SQLData Script Library

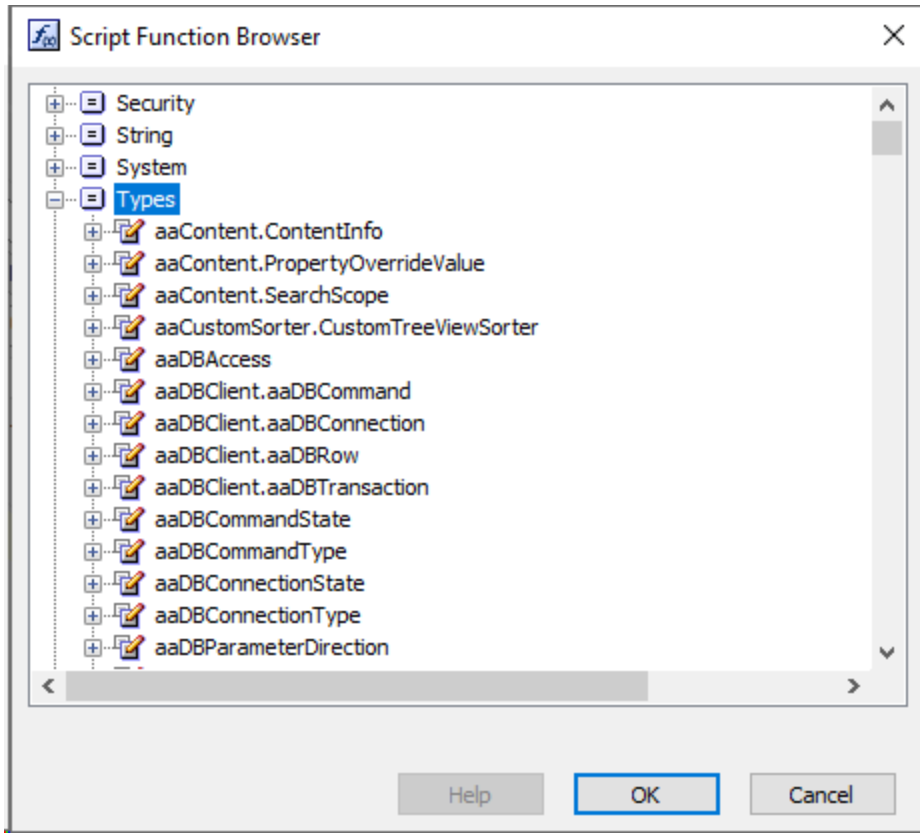
The SQLData Script Library is contained in the file named aaDBIntegration.aaSLIB. Copy the file to your ArchestrA development computer.

To access the SQLData Script Library

1. Open the System Platform IDE.
2. On the main menu, click **Galaxy/Import/Script Library**.

   After you import the SQLData Script Library, you can access the library functions from the **Script Function Browser** in the **Script** tab of any object. The functions are visible in the **Types** section.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.



For more information about working with function libraries, see the *AVEVA Application Server User's Guide*.

SQLData Script Library interface

This section provides an overview of the SQLData Script Library interface.

SQLData Script Library architecture

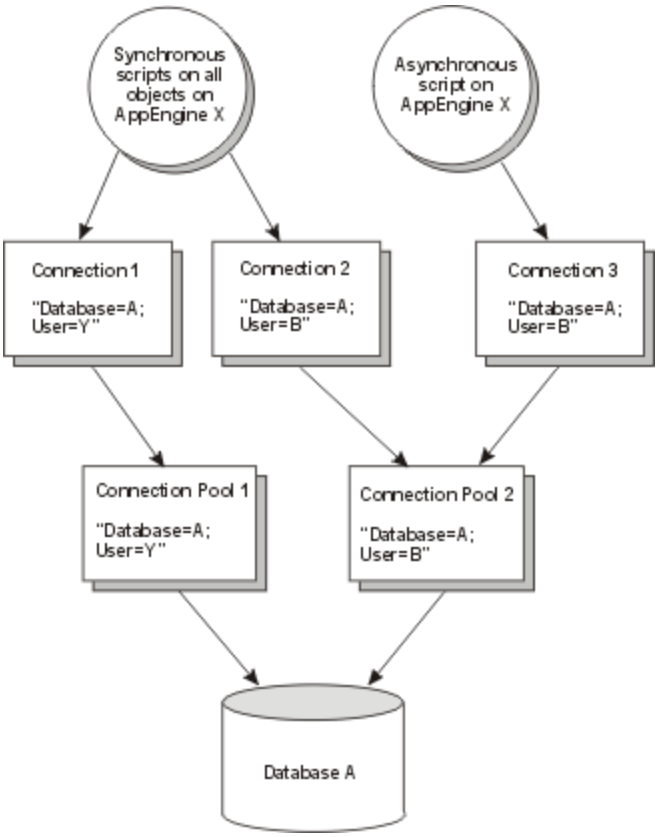
You can integrate the SQLData Script Library into ArchestrA by using synchronous or asynchronous scripting. The SQLData Script Library contains the following public objects:

- aaDBAccess
- aaDBConnection
- aaDBTransaction
- aaDBCommand
- aaDBRow

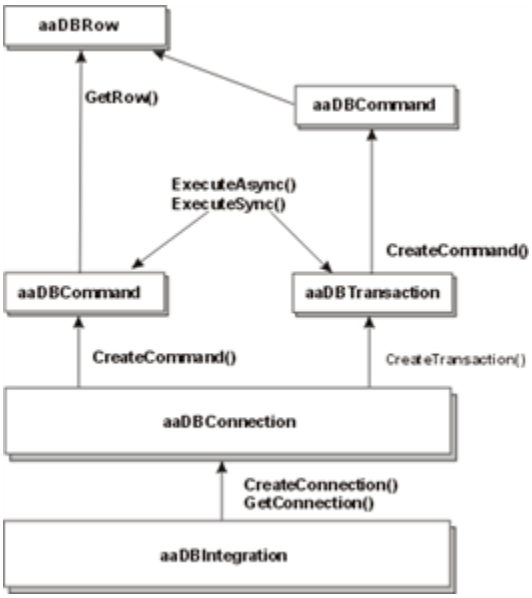
You can find details about these objects as well as their methods, properties, and enumerations in the remaining chapters in this guide.

The following figure shows connection pooling in the SQLData Script Library. Although each script generates its own connection object in the script library, scripts with identical connection strings are allocated to the same connection pool. The result is fewer connections to the database.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.



The following figure shows a conceptual diagram of the relationships among the various components of the SQLData script library.



Note that **aaDBRow** is one mechanism provided by **aaDBCommand** to read and modify data returned from the SQL query. The other mechanisms are implemented as methods of **aaDBCommand**.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

SQLData Script Library work flow

This section shows the work flow to create a connection object with a command and a transaction. This section also provides connection string examples.

Create a connection object with a command

SQLData scripts follow a typical flow when they are written without transactions:

1. Create a connection object.
2. Create one or more command objects using the **CreateCommand()** method of the connection object.
3. For each command object whose SQL statement contains parameters, initialize each parameter by using either the **Set<Type>ParameterByName()** method or the **Set<Type>ParameterByIndex()** method of the command object.
4. Run the command object either synchronously or asynchronously.

When command processing is complete, you can retrieve and modify the returned dataset. If you modify the dataset, you can save it back to the database either synchronously or asynchronously.

You can reuse command objects indefinitely. When finished, clean up the command objects by calling their **Dispose()** methods.

Create a connection object with a transaction

SQLData scripts follow a typical flow written with transactions:

1. Create a connection object.
2. Create a transaction object by using the **CreateTransaction()** method of the connection object.
3. Add one or more command objects to the transaction by using the **CreateCommand()** method of the transaction object.
4. For each command object whose SQL statement contains parameters, initialize each parameter by using either the **Set<Type>ParameterByName()** method or the **Set<Type>ParameterByIndex()** method of the command object.
5. Run the transaction object either synchronously or asynchronously.

If any of the commands fails or returns an error, the transaction and any commands that ran within the transaction are rolled back. The transaction is returned with the appropriate indicator

When transaction processing is complete, you can retrieve and modify the returned dataset for each command in the transaction that returns data. If you modify a dataset, you can save it back to the database either synchronously or asynchronously.

You can reuse transaction objects indefinitely. When finished, clean up the transaction object and command objects by calling their **Dispose()** methods.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Specify connection strings

The SQLData Script Library uses a connection string to specify the parameters for a database connection. The connection string is specified by Microsoft in their ADO.NET implementation. You can find details about the connection string and its parameters in the following locations:

`http://msdn2.microsoft.com/en-us/library/ms254978(VS.85).aspx`

`http://msdn2.microsoft.com/en-us/library/ms254499.aspx`

The following examples show how to write connection strings for the most commonly used data providers.

Connect to a SQL Server data source

The following script shows an example connection for SQL Server:

```
Connection=aaDBAccess.GetConnection(
   me.ConnectionString,
   aaDBConnectionType.Sql);
```

Using **aaDBConnectionType.Sql** is synonymous with the form of **GetConnection()** that takes only a connection string. Use the connection string example shown above in place of me.ConnectionString.

Connect to an Oracle data source

The following script shows an example connection for Oracle:

```
Connection=aaDBAccess.GetConnection("Provider=MSDAORA;
   Data Source=myOracleServer;User ID=<name>;
   Password=<password>.",aaDBConnectionType.Oracle);
```

Connect to Microsoft Access through OLEDB

The following script shows an example connection for Microsoft Access through OLEDB:

```
Connection=aaDBAccess.GetConnection(
   "Provider=Microsoft.Jet.OLEDB 4.0;
   Data Source=C:\myAccessDb.mdb",
   aaDBConnectionType.OleDb);
```

Connect to Microsoft Excel through OLEDB

The following script shows an example connection for Microsoft Excel through OLEDB:

```
Connection=aaDBAccess.GetConnection(
   "Provider=Microsoft.Jet.OLEDB 4.0;
   Data Source=C:\NameAndAddress.xls;
   Extended Properties=Excel 8.0;",
   aaDBConnectionType.OleDb);
```

Note: Microsoft Excel does not support transactions (commit or rollback).

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Example scripts

The following two examples illustrate the use of the SQLData Script Library to access a database.

Note: These scripts are provided only as a reference.

Overview of sample scripts

The sample scripts have the following purpose in a larger, ArchestrA context:

- The script provides an object that other requesting objects can use to read the specified column of a row whose part number is given and then return the column value to the requesting object.
- The requesting object provides the row name, part number, and start signal in UDAs.
- The sample object returns the requested row value and done signal in UDAs.

Detailed description of sample scripts

This section is a detailed step-by-step description of what is happening in the sample scripts that follow. For more information about using application objects, see the AVEVA *Application Server User Guide*.

1. Create a connection object and supply the connection string. The connection string in this example is a literal string, but you can use a connection string supplied by another ArchestrA object through a UDA.
2. If you are using transactions, create a transaction object on the connection object that you created.
3. Create a command object. This example supplies a literal string for the SQL statement, but it could be supplied by another ArchestrA object through a UDA. Note that the SQL statement is a query with a parameter, which allows the requesting object to specify the column to read:
   a. If you are not using transactions, create the command object on the connection object.
   b. If you are using transactions, create the command object on the transaction object.
4. Because the SQL statement of the command object contains a parameter, you must initialize that parameter object (or the transaction that contains it).
5. Run the transaction or command using either the **ExecuteAsync()** or **ExecuteSync()** methods.
6. If the command or transaction is processed synchronously, the SQLData Script Library does not return from the **ExecuteSync()** method until the command is complete. Because processing can take an extended period of time, you should use an asynchronous script (that is, a script that is not synchronized with the scan of the ArchestrA AppEngine).

   If the command or transaction is executed asynchronously, poll for completion as follows:
   - If your script runs synchronously with the scan of the ArchestrA AppEngine, you must signal polling to occur by a script once per scan cycle of the engine. This example script starts a second polling script, but you can write a single script with a state variable that starts processing in one state and polls in the other state.
   - If your script runs asynchronously with the ArchestrA AppEngine, you can poll in the following lines of the same script that starts processing using a while loop, which can take an extended period of time to finish.
7. When the command or transaction completes without an error, the result can be read from the command

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

object using the parameter that was initialized before the command object was run.

8. Check for errors at each of the major script processing steps.

Asynchronous command script

The scripts in this object show how to use asynchronous SQL processing in the SQLData Script Library. This script is written to use a command object directly on the connection, without using a transaction object. For comparison, see Synchronous transaction script where the example uses the synchronous SQL processing for a command object on a transaction object.

This object is written with two scripts. One script starts asynchronous command processing (the **Query** script) and the other script polls for asynchronous command completion and results manipulation (the **Process** script). Both scripts run synchronously with the ArchestrA AppEngine scan. To use the scripts, set the number to search for in the **PartNumber** User Defined Attribute (UDA) and column whose value is to be read in the **ColumnToRead** UDA. Signal the script to run by setting the **ReadCommand** UDA to True. When the script finishes processing, the **ColumnReadDone** UDA becomes True, and the results are in the **ColumnValue** UDA.

Query script configuration

On the **Script** tab of the SQLData Object Editor, configure the **Query** script with the following attributes.

Attribute	Value
Execution Type	Execute
Expression	me.ReadCommand
Trigger Type	OnTrue
Runs Asynchronously	Selected

Process script configuration

On the **Script** tab of the SQLData Object Editor, configure the **Process** script with the following attributes.

Attribute	Value
Execution Type	Execute
Expression	me.ProcessCommand
Trigger Type	WhileTrue
Runs Asynchronously	Selected

Query script code

Use the following sample code for the **Query** script.

```
DIM Connection as aaDBClient.aaDBConnection;
```

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

```
DIM Command as aaDBClient.aaDBCommand;
'Create a connection object with the connection string.
LogMessage("Creating connection");
Connection = aaDBAccess.CreateConnection("Data Source=localhost;Initial
Catalog=AdventureWorks;Integrated Security=true");
'Create a command object, with a SQL statement.
LogMessage("Creating a command object");
Command = Connection.CreateCommand("Select * from Production.Product WHERE ProductNumber =
@ProductNumber", aaDBCommandType.SqlStatement, true);
'We used a parameter to specify the value for the ProductNumber field, so initialize it.
Command.SetCharParameterByName("ProductNumber", me.PartNumber,
aaDBParameterDirection.Input, 50);
'Everything is ready, let's execute the command async.
LogMessage("Executing command async");
DIM ResultCode as integer;
ResultCode = Command.ExecuteAsync();
if ResultCode <> 0 then
'Failed to start async execution, report the reason.
LogMessage("Got error " + ResultCode + " executing command async");
else
'Execution started, identify the command by ID, for use later.
LogMessage("Command async execution started successfully");
me.CommandID = Command.GetID();
'Allow the Process script to run.
me.ProcessCommand = true;
endif;
'Reset for next time
me.ReadCommand = false;
```

Process script code

Use the following sample code for the **Process** script.

```
DIM Command as aaDBClient.aaDBCommand;
'Retrieve the command object using its ID.
Command = aaDBAccess.GetCommand(me.CommandID);
if Command <> null then
'Poll for command complete
if Command.ExecutionState <> aaDBCommandState.Queued then
LogMessage("Command execution state is " + Command.ExecutionState);
if Command.ExecutionState == aaDBCommandState.Completed then
DIM Rows as integer;
Rows = Command.RowCount;
LogMessage("Row count returned from command is " + Rows);
'We expect one row, use the first one, if we have any.
if Rows > 0 then
LogMessage("Getting column '" + me.ColumnToRead + "' from row 0");
Command.SelectRow(0);
'Return the requested column value from this row, signal done.
me.ColumnValue = Command.GetCurrentRowColumnByName(me.ColumnToRead);
me.ColumnReadDone = true;
endif;
endif;
'When done, dispose the command.
Command.Dispose();
'Reset for next time
```

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

```
me.ProcessCommand = false;
endif;
else
LogMessage("Cannot find command " + me.CommandID);
me.ProcessCommand = false;
endif;
```

Synchronous transaction script

The single script for this SQLData object shows how to use the synchronous SQL processing in the SQLData Script Library. This script is uses a transaction object with the command object on the transaction. For comparison, see Asynchronous command script where the example uses the command object directly on the connection, without a transaction.

This transaction object is written with a single script called **QueryandProcess**, which performs the SQL processing synchronously and then manipulates the results. This script runs without synchronizing with the ArchestrA AppEngine scan (asynchronous script). To use the script, set the part number to search for in the **PartNumber** UDA and the column whose value is to be read in the **ColumnToRead** UDA. Signal the script to run by setting the **ReadTransaction** UDA to True. When the script finishes processing, the **ColumnReadDone** UDA becomes True and the results are in the **ColumnValue** UDA.

QueryandProcess Script Configuration

On the **Script** tab of the SQLData Object Editor, configure the **QueryandProcess** script with the following attributes.

Attribute	Value
Execution Type	Execute
Expression	me.ReadTransaction
Trigger Type	OnTrue
Runs Asynchronously	Selected

QueryandProcess script code

Use the following sample code for the **QueryandProcess** script.

```
DIM Connection as aaDBClient.aaDBConnection;
DIM Command1 as aaDBClient.aaDBCommand;
DIM Transaction as aaDBClient.aaDBTransaction;
'Create a connection object with the connection string.
LogMessage("Creating connection");
Connection = aaDBAccess.CreateConnection("Data Source=localhost;Initial
Catalog=AdventureWorks;Integrated Security=true");
'Create a transaction object within the connection object.
LogMessage("Creating transaction object");
Transaction = Connection.CreateTransaction();
'Create a command object for the transaction object, with a SQL statement.
LogMessage("Creating a command object");
Command1 = Transaction.CreateCommand("Select * from Production.Product WHERE ProductNumber
= @ProductNumber", aaDBCommandType.SqlStatement, true);
```

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

```
'We used a parameter to specify the value for the ProductNumber field, so initialize it.
Command1.SetCharParameterByName("ProductNumber", me.PartNumber,
aaDBParameterDirection.Input, 50);
'Everything is ready, let's execute the transaction sync.
LogMessage("Executing transaction sync");
DIM ResultCode as integer;
ResultCode = Transaction.ExecuteSync();
if ResultCode <> 0 then
'Failed to execute transaction sync, report the reason.
LogMessage("Got error " + ResultCode + " executing transaction sync");
else
if Transaction.ExecutionState == aaDBTransactionState.Completed then
DIM Rows as integer;
Rows = Command1.RowCount;
LogMessage("Row count returned from command is " + Rows);
'Use other methods of script library to read data and assign to UDAs, etc.
if Rows > 0 then
LogMessage("Getting column '" + me.ColumnToRead + "' from row 0");
Command1.SelectRow(0);
'Return the requested column value from this row, signal done.
me.ColumnValue = Command1.GetCurrentRowColumnByName(me.ColumnToRead);
me.ColumnReadDone = true;
endif;
'When done, dispose the command.
Command1.Dispose();
endif;
'When done, dispose the transaction.
Transaction.Dispose();
endif;
'Reset for next time
me.ReadTransaction = false;
```

aaDBAccess object

The aaDBAccess object exposes only static methods. Use static methods with the aaDBAccess object to request a connection to the data source by providing a connection string. You can use two categories of methods to create a database connection:

- A reusable connection object: **GetConnection()**
- A unique connection object: **CreateConnection()**

By default, the SQLData Script Library assumes that a connection to a SQL Server database is requested and establishes a physical connection by using the **System.Data.SqlClient** namespace.

For details about methods that you can use with this object, see Methods.

Create a reusable connection object

To create a reusable connection object, use the following method:

```
aaDBAccess.GetConnection (<ConnectionString>, <ProviderType>)
```

For more information, see Connecting to databases other than SQL Server.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Create a unique connection object

To create a unique connection object for a specific purpose, use the following method:

```
aaDBAccess.CreateConnection(<ConnectionString>, <ProviderType>)
```

Differences between CreateConnection() and GetConnection()

The **CreateConnection()** method always creates a new connection object, even for calls with identical connection strings and even if **GetConnection()** has already been called with the same connection string.

Multiple calls to **GetConnection()** with identical—not similar— connection strings return the same connection object, over and over. If the connection string has never been used in a call to **GetConnection()**, a new connection object is created, but subsequent calls with the same connection string return the same connection object.

A call to **CreateConnection()** followed by a call to **GetConnection()** with the same connection string returns two different connection objects. That is, the connection object returned by **GetConnection()** is never the same as the connection object that has been returned by **CreateConnection()**.

Note: Because GetConnection() shares the same connection object across multiple scripts, connection pooling is effectively disabled for a specific connection string. All commands or transactions that run on a shared connection are routed through a single queue. Therefore, multiple physical connections for a single unique connection string cannot occur. If you want to use connection pooling, use the CreateConnection() method.

For more information, see Connecting to databases other than SQL Server.

Work with Connections

Supply all connection settings in a single parameter named ConnectionString that follows the documented Microsoft syntax. For details about Microsoft syntax, follow this link:

http://msdn2.microsoft.com/en-us/library/ms254499.aspx

Note: The parsing of the ConnectionString is not case sensitive.

For authentication, you can use one of the following security modes:

- Windows Integrated Security
- Windows Account
- SQL Server Authentication

Windows integrated security

Internally, **IntegratedSecurity** provides the connection by using the credentials of the currently logged on ArchestrA user.

Specify the following syntax in the connection string:

```
Integrated Security=True
```

The keyword `Integrated Security=True` in the connection string overrides any other authentication control. If the `Integrated Security=True` keyword is present, you get Windows User Authentication for the user who is currently logged on, even if you also include credentials for a different user.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

If you want to impersonate another Windows user's credentials, `Integrated Security=True` must be omitted from the connection string.

Windows account

In the **ConnectionString parameter**, you must provide the following information: domain, user name and password. The domain and user name must be specified using the following syntax:

```
User ID=<Domain>\<UserName>; Password=<pasword>;
```

The SQLData Script Library takes the following actions:

1. Removes domain, user name, and password from <**ConnectionString**>.
2. Sets **Integrated Security**=**True** in <**ConnectionString**>.
3. Configures the connection manager with the specified domain, user name, and password and then requests impersonation of this user.

The SQLData Script Library impersonates the user based on the properties that were provided in the connection string.

SQL Server authentication

Provide the following information in the **ConnectionString**;

```
User ID=<UserName>; Password=<password>;
```

The SQLData Script Library directly passes <**ConnectionString**> through to the **SQL Server database**.

The **aaDCM** object uses <**ConnectionString**> as is. There is no need for impersonation.

Methods

You can use the following methods with the **aaDBAccess** object.

CreateConnection()

Use the **CreateConnection()** method to request a connection to a SQL Server data source.

Syntax

```
aaDBConnection.CreateConnection(
string ConnectionString)
```

Parameters

ConnectionString

A previously formatted connection string or a reference to an attribute in any ArchestrA object.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Remarks

The **CreateConnection()** method returns a connection object to be used for subsequent SQL requests. Each call to **CreateConnection()** returns a unique and different connection object. Each connection object represents a separate connection that uses the same connection string. This method is best used to provide different connections for different purposes. For example, when one connection is used to query and another is used to update the database.

You can check the status of a connection by using the **ConnectionState** read-only property.

Example

The following example shows a connection string for use with SQL Server. The connection string can also be stored in an attribute in an Archestra object:

```
me.ExampleConnectionString
```

Connecting to databases other than SQL Server

Use the overloaded **GetConnection()** method to access a data source other than Microsoft SQL Server.

Syntax

```
aaDBConnection.GetConnection(
string ConnectionString,
aaDBConnectionType ConnectionType)
```

Parameters

Acceptable values for **ConnectionType** are as follows:

- OLEDB
- Oracle

Note: Use OLEDB to connect to the Microsoft Access data source.

GetCommand()

Use the **GetCommand()** method to retrieve a reference to an **aaDBCommand** object created previously with the same or a different script. Do not create a new **aaDBConnection** object to access a previously created command object.

Syntax

```
aaDBCommand.GetCommand(
string CommandId)
```

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Parameters

commandId

The Id is generated internally by the SQLData Script Library. For details, see GetId() on page 42.

Remarks

If **CommandID** does not represent a valid ID, **GetCommand()** returns a null reference.

GetConnection()

Use the **GetConnection()** method to request a connection to a data source.

Syntax

```
aaDBConnection.GetConnection(
string ConnectionString)
```

Parameters

ConnectionString

A previously formatted connection string or valid ArchestrA reference.

Remarks

The **GetConnection()** method returns a connection object to be used for subsequent SQL Server requests. Each time that **GetConnection()** is called with the identical connection string to a previous call, it returns the same connection object for reuse. This method is best used with a script that runs repeatedly, where a new connection object for each iteration would constitute a risk of memory leakage.

You cannot assume that a physical connection occurs after requesting a connection to a data source. Connections are opened only on an as-needed basis to perform an operation. The only time that you can check connectivity status is immediately after an operation completes.

This method immediately provides a connection object to be used for subsequent SQL requests. The first call to **GetConnection()** returns a unique connection object for the specified connection string, similar to **CreateConnection()**.

Subsequent calls to **GetConnection()** return references to the same connection object. Thus, you can make multiple calls to reuse the same object.

The connection returned by **GetConnection()** is never the same object as the one returned by **CreateConnection()**. This difference enables you to place **GetConnection()** calls at the top of a script that run once per scan without constantly creating connection objects.

Note: Be sure to call Dispose() on connection objects that have been created with this method. The aaDBAccess SQLData Script Library contains a reference to the objects. Garbage collection cannot be performed on them until you call Dispose().

You can check the status of the connection by using the **ConnectionState** read-only property.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Authentication

The following three database authentication methods are supported:

- Windows Integrated Security
- Windows Account
- SQL Server Authentication

You must provide a standard connection string that is created from keyword = value pairs separated by semicolons.

Follow this link for a list of connection string keywords:

http://msdn2.microsoft.com/en-us/library/ms254499.aspx

For Windows Integrated Security specify the following parameter in the connection string:

```
Integrated Security=true;
```

For Windows Account authentication, specify a connection string that adheres to the following syntax:

```
User ID=<domain>\<username>;Password=<password>;
```

For SQL Server Authentication, specify a connection string that adheres to the following syntax:

```
User ID=<UserName>;Password=<password>;
```

Connecting to databases other than SQL Server

Use the overloaded **GetConnection()** method to access a data source other than Microsoft SQL Server.

Syntax

```
aaDBConnection.GetConnection(
string ConnectionString,
aaDBConnectionType ConnectionType)
```

Parameters

Acceptable values for **ConnectionType** are as follows:

- OLEDB
- Oracle

Note: Use OLEDB to connect to the Microsoft Access data source.

GetDiagnostics()

Use the **GetDiagnostics()** method to return diagnostic information about all connections in a dataset.

Syntax

```
void GetDiagnostics()
```

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Remarks

The returned dataset contains multiple tables. The table with index 0 contains the global diagnostic information. The remaining tables in the dataset correspond to each DCM connection object. For details about diagnostic properties, see the *SQLData Object Help*.

GetTransaction()

Use the **GetTransaction()** method to obtain a reference to an **aaDBTransaction** object created previously in the same or a different script. Do not create a new **aaDBConnection** object to access a previously created transaction.

Syntax

```
aaDBTransaction GetTransaction(
string TransactionID)
```

Parameters

TransactionID

The ID is generated internally by the SQLData Script Library. See GetID() on page 89.

Remarks

If **TransactionID** does not represent a valid ID, **GetTransaction()** returns a null reference.

LogDiagnostics()

Use the **LogDiagnostics()** method to create a snapshot of all diagnostics available for a connection to be dumped to the logger. For details about diagnostic properties, see the *SQLData Object Help*.

Syntax

```
void LogDiagnostics()
```

RemoveCommand()

Use the **RemoveCommand()** method to instruct the SQLData Script Library to remove internal references to the **aaDBCommand** object referenced by **CommandID**, release all resources used by the object, and clean up all references to the object in memory.

Syntax

```
void RemoveCommand(
string CommandID)
```

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Parameters

CommandID

The unique ID, generated internally by the SQLData Script Library. For details, see GetId() on page 42.

Remarks

You must call **RemoveCommand()** if you previously requested **CommandID**. Otherwise, the memory cannot be released until the engine process is shut down.

Do not keep this command object in memory, especially when it is associated with a large dataset.

RemoveTransaction()

Use the **RemoveTransaction()** method to instruct the SQLData Script Library to remove all references to the **aaDBTransaction** object referred to by **TransactionID**, release all resources used by the object, and clean up all references to the object in memory.

Syntax

```
void RemoveTransaction(
string TransactionID)
```

Parameters

TransactionID

The unique ID, generated internally by the SQLData Script Library. See aaDBTransaction.GetID() on page 89.

Remarks

Internally, the SQLData Script Library ensures that all **aaDBCommand objects** explicitly added to this object are removed.

You must call this method if you previously requested the transaction ID. Otherwise, the memory cannot be released until the engine process is shut down.

Do not keep this command object in memory, especially when it is associated with a large dataset.

ResetDiagnostics

Use the **ResetDiagnostics()** method to reset the current diagnostic values associated with the **DCMConnectionMgr** and all **DCMConnections**.

Syntax

```
void ResetDiagnostics()
```

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Shutdown()

Use the **Shutdown()** method to gracefully cancel outstanding command object requests and release references to all persisted **aaDBCommand** and **aaDBTransaction** objects.

Call this method just once from a shutdown script in the hosting engine.

Syntax

```
void Shutdown()
```

aaDBCommand object

Use objects of type **aaDBCommand** to process SQL statements and stored procedures or to access a single table or view.

Create instances of type **aaDBCommand** by calling **CreateCommand()** on an instance of the **aaDBConnection object**. For example, assuming that the **aaDBConnection** instance is called **Connection**:

```
Connection.CreateCommand()
```

When a script requests the command object ID, the command object is flagged to be persisted.

The **aaDBCommand** objects are persisted across scripts and scan cycles but not across failover or shutdown.

Note: You must call Dispose() on each instance of aaDBCommand where an ID was created. You can also call **aaDBAccess.RemoveCommand()** with the ID.

You can retrieve an **aaDBCommand** object at any time by calling the static method aaDBAccess.GetCommand() and passing the previously acquired string ID.

The **aaDBCommand** object provides support for types that make sense for QuickScript. Other types may be supported by database columns, such as large text files or generic binary large objects (BLOBs), but a script might not be able to generate or analyze them. In general, if the script cannot manipulate objects of a particular type, such as a BLOB of type char[] or byte[], it might still be possible to read or write an object of that type to or from some alternate script library while storing it as type object within the script. In these cases, the object may be blindly written to a database using **SetCurrentRowColumnByName()** or **SetCurrentRowColumnByIndex()** and may be blindly read from a database using **GetCurrentRowColumnByName()** or **GetCurrentRowCollumnByIndex()**.

The life cycle of the aaDBCommand object follows this pattern:

1. The object is created by calling the **aaDBConnection.CreateCommand() method**.
2. Parameters are added to the command object as necessary.
3. The command object is run synchronously or asynchronously.
4. The **dataset** object wrapped by the **aaDBCommand** object is accessed and manipulated. This process may involve writing the **dataset** object back to the database.
5. Steps 2 through 4 can be repeated as needed.

Methods

You can use the following methods with the aaDBCommand object.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

AddRow()

Use the **AddRow()** method to add an empty row to a memory table and make it current.

You can then add values to the row by using the following methods:

- **SetCurrentRowColumnByIndex()**
- **SetCurrentRowColumnByName()**
- **SetCurrentRow()**

Syntax

```
result AddRow()
```

Remarks

The actual update to the data source is delayed until you call **SaveChangesSync()** or **SaveChangesAsync()**.

DeleteCurrentRow()

Use the **DeleteCurrentRow()** method to mark the currently selected row for deletion. The current row is not deleted from the data source until you call **SaveChangesSync()** or **SaveChangesAsync()**.

Syntax

```
result DeleteCurrentRow()
```

Dispose()

Use the **Dispose()** method to instruct the SQLData Script Library to free all memory resources associated with the command object. If the command is running, **Dispose()** cancels it. It is preferred to cancel the command before calling **Dispose()**.

Syntax

```
Void Dispose()
```

Remarks

After **Dispose()** is called, subsequent method calls to the command fail.

If an ID has been retrieved for this command object, you must call **Dispose()** or **aaDBAccess.RemoveCommand()**.

If you requested the ID of this command object, it is very important that you call **Dispose()** so it can flag the SQLData Script Library to clear all the references to this object.

When you request a **dataset** it is very important for you to call the **Dispose()** method when you are no longer interested in the results of a specific command object.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

If you have made changes to the memory **dataset** such as updating, deleting, or adding, but did not issue SaveChangesSync() or SaveChangesAsync(), all changes are discarded.

Subsequent method calls to request scrolling or to modify the memory **dataset** fail and return either a null object or error code 1016.

ExecuteAsync()

Use the **ExecuteAsync()** method to queue a command object in the connection for later processing. **ExecuteAsync()** returns immediately, and processing occurs in the background.

Syntax

```
result ExecuteAsync( )
```

Remarks

You can check for status by reading the ExecutionState read-only property.

After **ExecuteAsync()** is processed, you can still obtain a reference to the command objects and analyze their **ExecutionState** and **LastError** properties.

To free all resources allocated for the command object, you must call **Dispose()**.

ExecuteAsyncCancel()

Use the **ExecuteAsyncCancel()** asynchronous method to cancel the operations of a running command. If the command has already started running, it may run to completion. If the command is queued while waiting for other commands, it is removed from the queue without running.

Syntax

```
result ExecuteAsyncCancel( )
```

Remarks

To check for status, you can request the ExecutionState read-only property.

After the method has successfully completed, the command object is canceled.

ExecuteSync()

Use the **ExecuteSync()** method to run the command object. This method runs synchronously and blocks the engine thread.

Syntax

```
result ExecuteSync( )
```

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Remarks

Use the ExecutionState and **LastExecutionError** properties to check for status of this method.

After **ExecuteSync()** is processed you can still obtain a reference to the command object and analyze its **ExecutionState** and **LastError** properties.

Use the **LastExecutionError** property to check if the command failed or succeeded (success is indicated by a blank string), since ExecuteSync() may return 0 in the case of a failure instead of an error message.

To free all resources allocated for the **command** object, you must call Dispose().

GetCurrentRowColumnByIndex()

Use the **GetCurrentRowColumnByIndex()** method to obtain a specific column value from the current row. Use this method to read a single column value. For more information, see SelectRow().

You must provide a zero-based column index.

Syntax

```
object GetCurrentRowColumnByIndex(
int ColumnNumber)
```

Parameters

ColumnNumber

Zero-based index to the table column.

Return value

The object returned on failure is **Null**. The requested index is either negative, larger than the number of columns, or there is no valid current row.

GetCurrentRowColumnByName()

Use the **GetCurrentRowColumnByName()** method to obtain a specific column value from the current row by column name. Use this method to read a single column value. For more information, see SelectRow().

You must provide a column name.

Syntax

```
object GetCurrentRowColumnByName(
string ColumnName)
```

Parameters

ColumnName

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Column name.

GetDataSet()

Use the **GetDataSet()** method to retrieve the **dataset** stored in memory that was generated when the command object finished processing.

Syntax

```
DataSet GetDataset( )
```

Remarks

It is not recommended that you use this method under normal conditions, because the actual **dataset** may contain large amount of data (huge number of rows). This method is provided for advanced users that want to directly use the SQLClient namespace to interact with the **dataset**.

Note: If you execute this method in a synchronous script with a large number of rows, the script may time out. If you plan to use this method, it is recommended that you configure the script to run asynchronously.

ArchestrA QuickScript uses 1-based indexing when square-bracket notation is used with numeric indexes, such as **DataSet.Tables[7]**.

To avoid confusion, use square-bracket notation with collections that support strong name indexing, such as **DataSet.Tables["Customers"]**. You can also bypass the **dataset** object by using the wrapping accessor functions described in the next sections.

GetId()

Use the **GetID()** method to retrieve the ID of an **aaDBCommand** object instance for use in a different script or scan. The SQLData Script Library generates a unique command object ID, which remains unique across all scripts on the engine.

Syntax

```
string GetId()
```

Return value

On failure, this method returns the value **Null**.

Remarks

The SQLData Script Library returns a string value, but the script engine automatically attempts to cast this value to other types. If the script assigns the returned ID to any type other than string, the ID is corrupted and does not work in future **GetCommand(ID)** calls.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

GetParameterByIndex()

Use the **GetParameterByIndex()** method to retrieve output or to return parameters after the command object has been processed. If the parameter cannot be evaluated, the returned value is **null**.

Syntax

```
object GetParameterByIndex(
int Index)
```

Parameters

Index

1-based parameter index.

Remarks

You must provide a one-based parameter index.

Note: Use this method only for parameters defined in an OLEDB-type query, as specified by the connection object. If the connection type is not OleDb, null is returned.

GetParameterByName()

Use the **GetParameterByName()** method to retrieve output or return parameters after the command object has been processed. If the parameter cannot be evaluated, the returned value is **null**.

You must provide a parameter name.

Syntax

```
object GetParameterByName(
string ParameterName)
```

Parameters

ParameterName

Parameter name.

Remarks

It is not considered an error if the **ResultSet** returned from a query contains no rows. Zero-row results can also be expected as results of queries that contain parameters, so this might be an expected result, depending on the purpose of the query.

Note: Use this method only for parameters defined in a non-OLEDB-type query, as specified by the connection object. If the connection type is not OleDb, null is returned.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

GetRow()

Use the **GetRow()** method to quickly scroll through the records in the memory table and examine the row values and determine an index to be used for selecting a row of interest. For more information, see SelectRow().

You must provide a zero-based row index.

Syntax

```
aaDBRow GetRow(
int RowNum)
```

Parameters

RowNum

A zero-based row index less than the row count in the memory **dataset**.

Remarks

This method executes synchronously.

The **aaDBRow** returned object has two members:

- **columnNames**: An array filled with column names.
- **columnValues**: An array filled with column values corresponding to the row index.

The array items must be converted to strings before you can use them in string manipulation.

After the command object is successfully run you can use the **columnValues** array to set ArchestrA attributes.

Note: Before setting ArchestrA attributes you may need to cast the individual items from the columnValues array.

SaveChangesAsync()

Use the **SaveChangesAsync()** method to instruct the SQLData Script Library to write back to the data source all changes made to the memory **dataset**. The **dataset** must have been acquired by calling **ExecuteAsync()** or **ExecuteSync()**.

Syntax

```
result SaveChangesAsync()
```

Remarks

This method runs asynchronously. The request for updating the data source is queued and the method completes immediately.

You must check for status by getting the ExecutionState property value.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

SaveChangesSync()

This method is similar to SaveChangesAsync() except that it runs synchronously.

Syntax

```
result SaveChangesSync()
```

Remarks

This method blocks the engine thread when the script itself is synchronous.

It is highly recommended that you use this method only in asynchronous scripts.

When you configure the asynchronous script, make sure that the **TimeoutLimit** value is large enough to accommodate the time that this method may take to run the command object.

SelectRow()

Use the **SelectRow()** method to select the row with the index **rowNumber** in the memory table.

Syntax

```
result SelectRow(
long RowNumber)
```

Parameters

RowNum

The zero-based index of the record in the memory table.

Remarks

After you select a row, you can then read, delete or update the row using the following methods:

- **GetCurrentRowColumnByIndex()**
- **GetCurrentRowC()**
- **SetCurrentRowColumnByIndex()**
- **SetCurrentRowColumnByName()**
- **SetCurrentRow()**

SelectTable()

Use the **SelectTable()** method to select the table with the index TableIndex in the memory **dataset**.

You can then read, delete, or update this **DataTable**.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Syntax

```
result SelectTable(
long RowNumber)
```

Parameters

TableNumber

A zero-based index of the table in the memory **dataset**.

Remarks

By default, immediately after a command object returns a non- empty **dataset** that runs successfully the table with index 0 is selected.

SetCurrentRow()

Use the **SetCurrentRow()** method to set values for multiple columns at the same time. You must provide the column names to set and the corresponding values to be set, respectively, in the following two members of the **Row** parameter:

- **columnNames**
- **columnValues**

Syntax

```
result SetCurrentRow (
aaDBRow Row)
```

Parameters

Row

You can construct two **ArrayList objects** by using string constants or valid Archestra reference strings.

Remarks

When you construct the input **ArrayList** object, you can specify constants as well as valid ArchestrA reference strings.

Necessary conversions are performed internally to cast values to the specific column type.

The actual update to the data source is delayed until you call SaveChangesSync() or SaveChangesAsync().

If **SetCurrentRow()** encounters a failure to change a column, some columns may have already been changed in the internal dataset. In this case, the internal dataset reverts to the state it had immediately after the last update or **SaveChanges()** call. It is possible that this reversion of the internal dataset could undo changes applied prior to the current **SetCurrentRow()** call. For instance, if the script has a loop to modify 10 rows with **SetCurrentRow()** and nine rows return without a bad error code but the tenth row returns a bad error code, the internal dataset reverts to its state before the first **SetCurrentRow()** call.

Example

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

```
Dim inputCol as System.Collections.ArrayList;
Dim inputVal as System.Collections.ArrayList;
Dim inputRow as aaDBClient.aaDBRow;
inputCol = new System.Collections.ArrayList();
inputCol.Add("Name");
inputCol.Add("Description");
inputCol.Add("DateTime");
inputVal = new System.Collections.ArrayList();
inputVal.Add("Name");
inputVal.Add(me.strValue);
inputVal.Add(me.DT);
inputRow.columnNames = inputCol;
inputRow.columnValues = inputVal;
```

SetCurrentRowColumnByIndex()

Use the **SetCurrentRowColumnByIndex()** method to update the column at index **columnNumber** in the currently selected row.

Syntax

```
result SetCurrentRowColumnByIndex (
int ColumnNumber,
object NewValue)
```

Parameters

ColumnNumber

The zero-based index of the column in the memory table.

NewValue

You can specify a constant or a valid ArchestrA reference string. Both relative references and fully qualified references are supported. For example:

- me.ShortDesc – relative reference
- UD1.status – fully qualified reference string
- "John Smith" – constant

Remarks

Necessary conversions are performed internally to cast **newValue** to the column type.

When you try to change a value using this method and the type passed in does not match the type specified for the column in the database table, one of the following outcomes occurs:

- The value is written with an automated type conversion
- The transaction fails completely because the conversion would result in loss of data.

The actual update to the data source is delayed until you call SaveChangesSync() or SaveChangesAsync().

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

SetCurrentRowColumnByName()

Use the **SetCurrentRowColumnByName()** method to update the column named **ColumnName** in the currently selected row.

Syntax

```
result SetCurrentRowColumnByName (
string ColumnName,
object NewValue)
```

Parameters

ColumnName

String constant that is enclosed in quotation marks or another ArchestrA reference that can evaluate to a string. For example:
"LastName" – Spaces are allowed.
For more details see SQL Server documentation regarding column naming rules.

NewValue

You can specify a constant or a valid ArchestrA reference string. Both relative references and fully qualified references are supported. For example:

- me.ShortDesc – relative reference
- UD1.status – fully qualified reference string
- "John Smith" – constant

Remarks

Necessary conversions are performed internally to cast newValue to the column type.

When you try to change a value using this method and the type passed in does not match the type specified for the column in the database table, one of the following outcomes occurs:

- The value is written with an automated type conversion
- The transaction fails completely because the conversion would result in loss of data.

The actual update to the data source is delayed until you call **SaveChangesSync()** or **SaveChangesAsync()**.

SetParam type methods for SQL Server and Oracle

Frequently, the SQL statement that you use to create a command has parameters encoded in it. In such cases, you must set values for all parameters needed as input by the SQL statement before you run the command.

For SQL Server databases, parameters are indicated by @<ParameterName> so that each parameter is named. The following method call accommodates named parameters:

```
Set<Type>ParameterByName()
```

You must use the version of the method that matches the way that you created your connection object. When

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

you create a connection object with **aaDBConnectionTypeOleDb**, you must use the method **Set<Type>ParameterByIndex()**. For all other connections, use the method **Set<Type>ParameterByName()**.

Parameters are usually mapped to a table column, which always has a specific type. The method properties that you must supply with each of the methods in this subsection depend on the column type that they map to.

Separate methods are necessary because different column types require that you specify different method parameters.

For example, a column of type NVarChar requires a length be specified while a column type of Decimal requires precision and scale.

The method parameters that are common across all methods are:

- Parameter Name (parameterName)
- Parameter Value (parameterValue)

You can specify null for **parameterValue** when the Parameter Direction is Output or ReturnValue.

Parameter Direction (parameterDirection) enumerated values are:

- Input
- InputOutput
- Output
- ReturnValue

Output parameters

The **GetParameterByIndex()** and **GetParameterByName()** methods return values to the script through InputOutput, Output, or ReturnValue parameters.

It is not possible to return values directly to ArchestrA attributes through InputOutput, Output or ReturnValue parameters, only DIM script variables. This is a constraint of the scripting infrastructure implemented in Application Server version 3.0.

If you want the value returned by **GetParameterByIndex()** or **GetParameterByName()** to set a value to an ArchestrA attribute reference, perform the following steps:

1. Declare a local script variable and return the InputOutput, Output, or ReturnValue parameter by making a **GetParameterByIndex()** or **GetParameterByName()** method call.
2. Use this local variable to set the desired ArchestrA reference.

SetBitParameterByName()

Note: Do not use this method for the Oracle data type Boolean value. Use **SetIntParameterByName** instead.

Use the **SetBitParameterByName()** method to configure a bit parameter by the name encoded into the text of a SQL statement using the @ character.

Syntax

```
result SetBitParameterByName (
string ParameterName,
```

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

```
object ParameterValue,
aaDBParameterDirection ParameterDirection)
```

Parameters

ParameterName

String identifier used in the SQL statement.
In the following SQL statement the parameter name is boolValue.

```
   "SELECT StateProvinceID
      ,StateProvinceCode
      ,CountryRegionCode
      ,IsOnlyStateProvinceFlag
      ,Name
      ,TerritoryID
      ,rowguid
      ,ModifiedDate
   FROM AdventureWorks.Person.StateProvince
   WHERE IsOnlyStateProvinceFlag = @boolValue"
```

ParameterValue
A discrete value or a valid ArchestrA reference string. For example:

- 0
- false
- me.boolValue

Example

```
SetBitParameterByName ("boolValue", me.boolValue, aaDBParameterDirection.Input)
```

SetCharParameterByName()

Use the **SetCharParameterByName()** method to configure a character string parameter by the name encoded into the text of a SQL statement using the @ character.

Syntax

```
result SetCharParameterByName (
string ParameterName,
object ParameterValue,
aaDBParameterDirection ParameterDirection,
int Length)
```

Parameters

ParameterName

The string identifier that is used in the SQL statement.
In the following SQL statement the input parameter is "lastName."

```
   "SELECT * FROM Person.Contact WHERE (LastName = @lastName)"
```

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

ParameterValue

A string constant or valid ArchestrA reference string. For example:

- "Smith"
- me.LastName
- null - Output and ReturnValue parameters.

ParameterDirection

For possible values, see aaDBParameterDirection.

Length

Specifies the maximum length of the parameter.
If the length of the parameterValue is greater, the **parameterValue** is truncated to the specified length.

Example

```
SetCharParameterByName ("lastName", "Smith",aaDBParameterDirection.Input, 50)
```

SetDecimalParameterByName()

Use the **SetDecimalParameterByName()** method to configure a decimal parameter by the name encoded into the text of a SQL statement using the @ character.

Syntax

```
result SetDecimalParameterByName (
string ParameterName,
object ParameterValue,
aaDBParameterDirection ParameterDirection,
short Precision,
short Scale)
```

Parameters

ParameterName

The string identifier used in the SQL statement.
In the following SQL statement the name of the parameter is "RejectedQuantity".

```
   "INSERT INTO Purchasing.PurchaseOrderDetail
      (PurchaseOrderID
      ,DueDate
      ,OrderQty
      ,ProductID
      ,UnitPrice
      ,ReceivedQty
      ,RejectedQty
      ,ModifiedDate)
      VALUES
      (4,'2008-01-28',4,4,4,12.89,@RejectedQuantity,
      '2008-01-28') "
```

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

ParameterValue

This parameter supports float, double or string values.
You can also specify a valid ArchestrA reference string. For example:

- 123.333
- "123.333333333333333333"
- me.Quantity

ParameterDirection

For possible values, see aaDBParameterDirection.

Precision

A number that indicates the total number of digits.

Scale

A number that indicates the number of digits to the right of the decimal point.

Example

SetDecimalParameterByName ("RejectedQuantity", me.Quantity,aaDBParameterDirection.Input, 8, 2)

SetDateTimeParameterByName()

Use the **SetDateTimeParameterByName()** method to configure a **DateTime** parameter by the name encoded into the text of a SQL statement using the @ character.

Syntax

```
result SetDateTimeParameterByName (
string ParameterName,
object ParameterValue,
aaDBParameterDirection ParameterDirection)
```

Parameters

ParameterName

The string identifier used in the SQL statement. For example, in the following SQL statement, the name of the parameter is "NewDate"

```
   "INSERT INTO Purchasing.PurchaseOrderDetail
      (PurchaseOrderID
      ,DueDate
      ,OrderQty
      ,ProductID
      ,UnitPrice
      ,ReceivedQty
      ,RejectedQty
      ,ModifiedDate)
      VALUES
      (4,'2008-01-28',4,4,4,12.89,12.56,@NewDate)"
```

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

ParameterValue

The **DateTime,** string value, or a valid ArchestrA reference string.
For example:

- '2004-03-11 10:17:21.587'
- me.dtValue

ParameterDirection

For possible values, see aaDBParameterDirection.

Example

SetDateTimeParameterByName ("NewDate",me.dtValue,aaDBParameterDirection.Input)

SetDoubleParameterByName()

Use the **SetDoubleParameterByName()** method to configure a double parameter by the name encoded into the text of a SQL statement using the @ character.

Syntax

```
result SetDoubleParameterByName (
string ParameterName,
object ParameterValue,
aaDBParameterDirection ParameterDirection)
```

Parameters

ParameterName

The string identifier used in the SQL statement.
In the following SQL statement the output parameter name is "AvgReject"

```
   "SELECT @AvgReject = Avg(RejectedQty)
   FROM Purchasing.PurchaseOrderDetail "
```

ParameterValue

A double-precision floating number or a valid ArchestrA reference string. For example:

- 123.333
- me.Limit

ParameterDirection

For possible values, see aaDBParameterDirection on.

Example

```
SetDoubleParameterByName ("AvgReject",null,aaDBParameterDirection.Output)
```

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

SetFloatParameterByName()

Use the **SetFloatParameterByName()** method to configure a float parameter by the name encoded into the text of a SQL statement using the @ character.

Syntax

```
result SetFloatParameterByName (
string ParameterName,
object ParameterValue,
aaDBParameterDirection ParameterDirection
```

Parameters

ParameterName

String identifier used in the SQL statement.
In the following SQL statement the name of the output parameter is "AvgReject."

```
   "SELECT @AvgReject = Avg(RejectedQty)
   FROM Purchasing.PurchaseOrderDetail "
```

ParameterValue

A floating number or a valid ArchestrA reference string. For example:

- 123.333
- me.Limit

ParameterDirection

For possible values, see aaDBParameterDirection.

Example

```
SetFloatParameterByName ("AvgReject", null, aaDBParameterDirection.Output)
```

SetIntParameterByName()

Use the **SetIntParameterByName()** method to configure an integer parameter by the name encoded into the text of a SQL statement using the @ character.

Note: Use this method for the Oracle data type Boolean.It is specific to Oracle.

Syntax

```
result SetIntParameterByName (
string ParameterName,
object ParameterValue,
aaDBParameterDirection ParameterDirection)
```

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Parameters

ParameterName

The string identifier that is used in the SQL statement.
In the following SQL statement the name of the output parameter is "Cnt"

```
"SELECT @Cnt=Count (*)(RejectedQty)
FROM Person.Contact "
```

ParameterValue

An integer value or a valid ArchestrA reference string. For example:

- 10
- me.OrderCount
- Null - Output

ParameterDirection

For possible values, see aaDBParameterDirection.

Example

```
SetIntParameterByName ("Cnt", null, aaDBParameterDirection.Output)
```

SetLongParameterByName()

Use the **SetLongParameterByName()** method to configure a long parameter by the name encoded into the text of a SQL statement using the @ character.

Syntax

```
result SetLongParameterByName (
string ParameterName,
object ParameterValue,
aaDBParameterDirection ParameterDirection)
```

Parameters

ParameterName

The string identifier that is used in the SQL statement.
In the following SQL statement the name of the output parameter is "ProdID"

```
"SELECT ProductID, SUM(WorkOrderID)
AS OrderCnt
FROM Production.WorkOrder
WHERE ProductID = @ProdID
GROUP BY ProductID"
```

ParameterValue

An integer value or a valid ArchestrA reference string. For example:

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

- 10
- me.ProductID
- Null - Output

ParameterDirection

For possible values, see aaDBParameterDirection.

Example

```
SetLongParameterByName ("ProdID",me.ProductID,aaDBParameterDirection.Output)
```

SetParam type methods for OLEDB

Frequently, the SQL statement that you use to create a command has parameters encoded in it. In such cases, you must set values for all parameters needed as input by the SQL statement before you run the command.

For data providers accessed through OLEDB, parameters are indicated by identical placeholder characters, and parameters must be indicated by index. The method call **Set<Type>ParameterByIndex()** accommodates the indexed parameters of OLEDB.

Parameters are usually mapped to a table column, which always has a specific type. The method parameters that you supply with each of the methods in this subsection vary depending on the column type that they map to.

Note: Because parameters are sequentially numbered, it is important that there be no gaps in the sequence. The sequence is checked when the command object with parameters is executed. Gaps cause the command object to fail.

It is necessary to have separate methods because different column types require that you specify different method parameters.

For example a column of type NVarChar requires that you specify a length, while a column type of Decimal requires that you specify precision and scale.

The following method parameters are common across all methods:

- Parameter Index (**Index**) – this is 1-based
- Parameter Value (**parameterValue**)

  You can specify **null** for **parameterValue** when the **parameterDirection** is **Output** or **ReturnValue**.
- Parameter Direction (**parameterDirection**)

The following are enumerated values:

- Input
- InputOutput
- Output
- ReturnValue

SetBitParameterByIndex()

Use the **SetBitParameterByIndex()** method to configure a bit parameter as encoded into the text of an OLEDB

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

SQL statement.

Syntax

```
result SetBitParameterByIndex (
int Index,
object ParameterValue,
aaDBParameterDirection ParameterDirection)
```

Parameters

Index

The sequential index of the parameter as used in the SQL statement.
In the following SQL statement, the parameter corresponds to the question mark (?) character:

```
   "SELECT StateProvinceID
      ,StateProvinceCode
      ,CountryRegionCode
      ,IsOnlyStateProvinceFlag
      ,Name
      ,TerritoryID
      ,rowguid
      ,ModifiedDate
   FROM AdventureWorks.Person.StateProvince
   WHERE IsOnlyStateProvinceFlag = ?"
```

ParameterValue

A discrete value or a valid ArchestrA reference string. For example:

- 0.
- false.
- me.boolValue.
- ParameterDirection. For possible values, see aaDBParameterDirection.

Example

```
SetBitParameterByIndex (0, me.boolValue, aaDBParameterDirection.Input)
```

SetCharParameterByIndex()

Use the **SetCharParameterByIndex()** method to configure a character string parameter as encoded into the text of an OLEDB SQL statement.

Syntax

```
result SetCharParameterByIndex (
int Index,
string ParameterValue,
aaDBParameterDirection ParameterDirection,
int Length)
```

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Parameters

Index

The sequential index of the parameter as used in the SQL statement.
In the following SQL statement the input parameter corresponds to the question mark (?) character.

```
   "SELECT * FROM Person.Contact WHERE (LastName > ?)"
```

ParameterValue

A string constant or valid ArchestrA reference string. For example:

- "Smith"
- me.LastName
- null - Output and ReturnValue parameters

ParameterDirection

For possible values, see aaDBParameterDirection.

Length

The maximum length of the parameter.
If the length of the parameterValue is greater, the parameterValue is truncated to the specified length.

Example

```
SetCharParameterByIndex (0, "Smith",aaDBParameterDirection.Input, 50)
```

SetDateTimeParameterByIndex()

Use the **SetDateTimeParameterByIndex()** method to configure a DateTime parameter as encoded into the text of an OLEDB SQL statement.

Syntax

```
result SetDateTimeParameterByIndex (
int Index,
object ParameterValue,
aaDBParameterDirection Parameterdirection)
```

Parameters

Index

The sequential index of the parameter as used in the SQL statement. For example:
In the following SQL statement the parameter corresponds to the question mark (?) character:

```
   "INSERT INTO Purchasing.PurchaseOrderDetail
      (PurchaseOrderID
      ,DueDate
      ,OrderQty
      ,ProductID
      ,UnitPrice
```

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

```
      ,ReceivedQty
      ,RejectedQty
      ,ModifiedDate)
   VALUES
      (4,'2008-01-28',4,4,4,12.89,12.56,?)"
```

ParameterValue

DateTime, string value, or a valid ArchestrA reference string.
For example:

- '2004-03-11 10:17:21.587'
- me.dtValue

ParameterDirection

For possible values, see aaDBParameterDirection.

Example

```
SetDateTimeParameterByIndex (1, me.dtValue, aaDBParameterDirection.Input)
```

SetDecimalParameterByIndex()

Use the **SetDecimalParameterByIndex()** method to configure a decimal parameter as encoded into the text of an OLEDB SQL statement.

Syntax

```
result SetDecimalParameterByIndex (
int Index,
object ParameterValue,
aaDBParameterDirection ParameterDirection,
short Precision,
short Scale)
```

Parameters

Index

Sequential index of the parameter as used in the SQL statement. For example:
In the following SQL statement the parameter corresponds to the question mark (?) character:

```
   "INSERT INTO Purchasing.PurchaseOrderDetail
      (PurchaseOrderID
      ,DueDate
      ,OrderQty
      ,ProductID
      ,UnitPrice
      ,ReceivedQty
      ,RejectedQty
      ,ModifiedDate)
   VALUES
      (4,'2008-01-28',4,4,4,12.89,?,'2008-01-28') "
```

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

ParameterValue

This parameter supports float, double, or string values.
You can also specify a valid ArchestrA reference string. For example:

- 123.333
- "123.333333333333333333"
- me.Quantity

ParameterDirection

For possible values, see aaDBParameterDirection.

Precision

A number that indicates the total number of digits.

Scale

A number that indicates the number of digits to the right of the decimal point.

Example

```
SetDecimalParameterByIndex (1, me.Quantity,aaDBParameterDirection.Input, 8, 2)
```

SetDoubleParameterByIndex()

Use the **SetDoubleParameterByIndex()** method to configure a double parameter as encoded into the text of an OLEDB SQL statement.

Syntax

```
result SetDoubleParameterByIndex (
int Index,
object ParameterValue,
aaDBParameterDirection ParameterDirection)
```

Parameters

Index

The sequential index of the parameter as used in the SQL statement.
In the following SQL statement, the output parameter corresponds to the question mark (?) character:

```
   "SELECT ? = Avg(RejectedQty)
   FROM Purchasing.PurchaseOrderDetail "
```

ParameterValue

A double-precision floating number or a valid ArchestrA reference string. For example:

- 123.333
- me.Limit

ParameterDirection

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

For possible values, see aaDBParameterDirection.

Example

```
SetDoubleParameterByIndex (1, null, aaDBParameterDirection.Output)
```

SetFloatParameterByIndex()

Use the **SetFloatParameterByIndex()** method to configure a float parameter as encoded into the text of an OLEDB SQL statement.

Syntax

```
result SetFloatParameterByIndex (
int Index,
object ParameterValue,
aaDBParameterDirection ParameterDirection)
```

Parameters

Index

The sequential index of the parameter as used in the SQL statement.
In the following SQL statement the output parameter corresponds to the question mark (?) character:

```
   "SELECT ? = Avg(RejectedQty)
   FROM Purchasing.PurchaseOrderDetail "
```

ParameterValue

A double-precision floating number or a valid ArchestrA reference string. For example:

- 123.333
- me.Limit

ParameterDirection

For possible values, see aaDBParameterDirection.

Example

```
SetFloatParameterByIndex (1, null, aaDBParameterDirection.Output)
```

SetIntParameterByIndex()

Use the **SetIntParameterByIndex()** method to configure a integer parameter as encoded into the text of an OLEDB SQL statement.

Syntax

```
result SetIntParameterByIndex (
```

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

```
int Index,
object ParameterValue,
aaDBParameterDirection ParameterDirection)
```

Parameters

Index

The sequential index of the parameter as used in the SQL statement.
In the following SQL statement the output parameter corresponds to the question mark (?) character.:

```
   "SELECT ? = Avg(RejectedQty)
   FROM Purchasing.PurchaseOrderDetail "
```

ParameterValue

A double-precision floating number or a valid ArchestrA reference string. For example:

- 123.333
- me.Limit

ParameterDirection

For possible values, see aaDBParameterDirection.

Example

```
SetIntParameterByIndex (1, null, aaDBParameterDirection.Output)
```

SetLongParameterByIndex()

Use the **SetLongParameterByIndex()** method to configure a long parameter as encoded into the text of an OLEDB SQL statement.

Syntax

```
result SetLongParameterByIndex (
int Index,
object ParameterValue,
aaDBParameterDirection ParameterDirection)
```

Parameters

Index

The sequential index of the parameter as used in the SQL statement.
In the following SQL statement the output parameter corresponds to the question mark (?) character:

```
   " SELECT ProductID, SUM(WorkOrderID) AS OrderCnt
   FROM Production.WorkOrder
   WHERE ProductID = ?
   GROUP BY ProductID"
```

ParameterValue

A double-precision floating number or a valid ArchestrA reference string. For example:

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

- 123.333
- me.ProductID

ParameterDirection

For possible values, see aaDBParameterDirection.

Example

```
SetLongParameterByIndex (1, me.ProductID, aaDBParameterDirection.Output)
```

Properties

You can use the following properties with the **aaDBCommand** object.

CommandTimeout

This property accesses the underlying **CommandTimeout** property of the DCM command object. The DCM command object in turn, accesses the **ADO.NetDbCommand.CommandTimeout** property. According to MSDN, this **CommandTimeout** property gets or sets the wait time before terminating the attempt to run a command object and generating an error.

Notes:

- If the DCM command object has been removed or is otherwise invalid, reading this property returns **0**; writing it has no effect. An exception is not thrown.
- When aaDBConnectionType=Oracle, this property is not supported. Setting a command object timeout has no effect, and the value returned is always **0.**

CurrentRowNumber

This read-only property returns the row value most recently set by the last call to **SelectRow()** or the new row number that has been created if **AddRow()** has been called since **SelectRow()**.

This property returns a value of **-1** if any of the following conditions occurred:

- A call to **SelectRow()** was not made.
- An invalid row number was passed in the last call to **SelectRow()**.
- **DeleteCurrentRow()** was called.

Syntax

```
long CurrentRowNumber;
```

CurrentTableNumber

This read-only property returns the table index most recently set by the last call to **SelectTable()**.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

This property returns 0 by default immediately after a command object (returning a dataset that is not empty) is run successfully.

This property returns -1 if any of the following conditions occurred:

- An invalid index was passed in the last call to **SelectTable()**
- The resulting dataset is empty

Syntax

```
long CurrentTableNumber;
```

Disposed

This read-only property returns a Boolean value that indicates whether the command object has been disposed. If disposed, the command object can no longer be used.

ExecutionState

This read-only property returns the execution state of a SQL command. Because command processing can be asynchronous, you must determine whether the command has been processed before you request the execution state.

Syntax

```
aaDBCommandState.ExecutionState
```

LastExecutionError

This property returns the last error, if any, that is the error string generated by the provider.

Use this property in conjunction with the ExecutionState property.

The **LasExecutionError** property returns the error generated by the provider for the failed command object.

Syntax

```
string LastExecutionError
```

RowCount

Use this read-only property to obtain the number of records in the memory **dataset** that was retrieved by a SQL command. When it is used with a SQL action command such as **INSERT**, **DELETE**, or **UPDATE**, the property returns the number of records affected.

This property returns a value of zero in the following cases:

- The SQL command has not been queued.
- The SQL command has been queued but is not completed.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

- The SQL command has completed with an error.

Syntax

```
int RowCount;
```

Public enumerations

The following public enumerations apply to the **aaDBCommand** object.

aaDBCommandState

This public enumeration indicates the state of an **aaDBCommand** object.

Created

This value indicates that the command has been created. However, the command has not yet run, and you can assign parameters to the command.

Queued

This value indicates that the command is queued for execution. This state occurs immediately after the call to **aaDBCommand.ExecuteAsync() or aaDBCommand.ExecuteSync()**. **Queued** is a transitional state that changes to either **Failed** or **Completed**.

Failed

This value indicates that the command failed.

Completed

This value indicates that the command completed successfully.

Canceled

This value indicates that the command has been canceled.

Disposed

This value indicates that **Dispose()** has been called for the command object. **Disposed** is automatically set when **aaDBAccess.RemoveCommand()** is called, or when **Dispose()** is called on the connection object or transaction object that owns the command.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

aaDBCommandType

This public enumeration describes how the command text for an **aaDBCommand** object is used.

Note: When you use the aaDBCommandType enumeration, be aware that not all requests work with all data providers. For example, Microsoft Access does not support stored procedures. The InputOutput direction is not supported by all data providers. Always check the documentation for the data providers about supported options.

sqlStatement

Indicates that the command text is a valid SQL statement.

storedProcedure

Indicates that the command text is a stored procedure name.

aaDBParameterDirection

This public enumeration indicates the direction of parameters used for SQL statements.

Note: When you use the **aaDBParameterDirection** enumeration, be aware that not all requests work with all data providers. For example, Microsoft Access does not support stored procedures. The InputOutput direction is not supported by all data providers. Always check the documentation for the data providers about supported options.

Input

This value indicates that the parameter is input only.

InputOutput

This value indicates that the parameter is capable of both input and output.

Output

This value indicates that the parameter is output only.

ReturnValue

This value indicates the presence of a return value from an operations such as a stored procedure, built-in function, or user-defined function.

aaDBConnection object

Use objects of type **aaDBConnection** to create a new **aaDBCommand** object or **aaDBTransaction** object. You can

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

create instances of **aaDBConnection** only through the static methods **aaDBAccess.CreateConnection()** or **aaDBAccess.GetConnection()**.

Creating an instance of the **aaDBConnection** object is not equivalent to creating and maintaining a physical connection to the data source.

The actual physical connection opens on demand when a request is made to run **aaDBCommand** objects or **aaDBTransaction** objects. The physical connections are controlled by the Database Connection Manager (DCM).

The **SQLData** Script Library does not expose any mechanism to allow you to fine-tune its behavior, such as defining how many connections to use.

Connection pooling

When you request a SQL command or transaction to be run on the **aaDBConnection** object, an actual physical connection is opened on demand and closed when the command or transaction finishes processing.

The SQLData Script Library uses the DCM common component in regard to connection pooling. When an **aaDBConnection** object is first created, no attempt is made to achieve a physical connection. The **ConnectionState** property remains in the **Disconnected** state. The physical connection is attempted only after a command object is run (a stand-alone command object or a command object that is part of a transaction).

After the first command object that requires a physical connection runs, the physical connection remains open for an unspecified period (depending on loading) and closes automatically after a time.

If a script requires confirmation of a successful physical connection, it must issue some benign SQL statement and then check the **ConnectionState** property of the connection.

Note: You must call **Dispose()** on each instance of aaDBConnection.

Methods

You can use the following methods with the **aaDBConnection** object.

CreateCommand()

Use the **CreateCommand()** method to create a new **aaDBCommand** object.

Syntax

```
aaDBCommand CreateCommand(
string CommandText,
aaDBCommandType CommandType,
bool ReturnDataset)
```

Parameters

You can also configure the newly created object by adding parameters and then executing them.
The parameters are identified in the **SQL statement** by the at sign (@) character when **ProviderType** is **SQL**.
The following SQL statement has one parameter, **lastName**:

```
   "SELECT * FROM Person.Contact WHERE (LastName > @lastName)"
```

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

CreateCommand for Oracle

The parameters are identified in the SQL statement by the colon (:) character when **ProviderType** is **Oracle**.

```
"SELECT * FROM Person.Contact WHERE (LastName > :lastName)"
```

CreateCommand for OLEDB

The parameters are identified in the SQL statement by the question mark (?) character and no name when **ProviderType** is **OLEDB**.

```
"SELECT * FROM Person.Contact WHERE (LastName > ?)"
```

In this case, you configure the parameters by index and not by name.

Be aware that you cannot mix configuration parameters by index and by name.

Return Value

If a failure occurs, this method returns **null**.

Getting the **ExecutionState** property of the newly-created command object returns Created.

Parameters

CommandText

One of the following:
SQL Statement
Stored Procedure name

CommandType

Specifies how the SQL statement is handled:
sqlStatement
storedProcedure

ReturnDataset

Boolean, indicates if the command object is to return a **dataset**
If **True,** you want a data table to be returned. For example:

sqlStatement ="SELECT * FROM Person.Contact"
If **False**, you want to modify only the database or get the **Output** or **ReturnValue** parameter. Examples are INSERT, DELETE, and UPDATE SQL statements.

The data that is returned upon successful execution of a query (Table, View or StoredProcedure) is stored in memory. Scalar values are not returned, other than by being part of the returned dataset.

You can then access this data by using the interface provided by the aaDBCommand object.

CreateTransaction()

Use the **CreateTransaction()** method to create a new **aaDBTransaction** object.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Syntax

```
aaDBTransaction CreateTransaction()
```

Return Value

If a failure occurs, this method returns **null**.

Getting the **ExecutionState** property of the newly-created **aaDBTransaction** object returns Created.

Remarks

You add **aaDBCommand** objects to this object that are processed as a whole in the order that they were added.

Dispose()

Use the **Dispose()** method to free all memory resources associated this database connection object.

Syntax

```
void Dispose()
```

Remarks

If commands or transactions are running when **Dispose()** is called, **Dispose()** cancels the command or transaction object.

GetDiagnostics()

Use the **GetDiagnostics()** method to return a set of diagnostic information about all connections in a **dataset**.

Syntax

```
void GetDiagnostics()
```

Remarks

The returned dataset contains multiple tables. The table with index 0 contains the global diagnostics. The rest of the tables in the **dataset** correspond to each **DCMConnection** object. For details about diagnostic properties, see the *SQLData Object Help*.

LogDiagnostics()

Use the **LogDiagnostics()** method to dump a snapshot of all available connection diagnostic information to the logger. For details about diagnostic properties, see the *SQLData Object Help*.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Syntax

```
void LogDiagnostics()
```

ResetDiagnostics

Use the **ResetDiagnostics()** method to reset the current diagnostic values of the connection object.

Syntax

```
void ResetDiagnostics()
```

Properties

You can use the following properties with the **aaDBConnection** object.

ConnectionName

Use this property to enable the log of diagnostics that was generated by calling **aaDBAccess.LogDiagnostics()** to include a meaningful name. Having a meaningful name may be necessary if a debugging effort requires you to distinguish one connection from another. To see the connection name in the logged diagnostics, look for the line that reads as follows:

```
aaDBIntegration ConnectionName, <Name>, , Clilent assigned conection name,
```

The <Name> part of the line is replaced with the name that the script has applied to the ConnectionName property of the aaDBConnection object. If the ConnectionName property has not been assigned a name by the script, it defaults to a name of the form SQLScriptConnection<N>, where <N> is an incrementing integer value.

Syntax

```
String ConnectionName
```

ConnectionState

Use this read-only property to verify that the connection string specified by **GetConnection** successfully establishes the initial connection to the data source.

Note: You cannot use this property to determine the status of an actual physical connection to the data source. No polling mechanism is available to detect a broken connection.

If intermittent network failures occur, **ConnectionState** does not indicate the failure until a command or transaction is run. However, when a command is run, the **ConnectionState** property updates the current connection state.

The return type is an enumerated value of type **aaDBConnectionState**.

Syntax

```
aaDBConnectionState ConnectionState
```

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Disposed

This read-only property indicates whether **Disposed()** has been called for a connection object. If the property returns **True**, no other method or property can be called on this instance.

Syntax

```
Bool Disposed
```

LastError

Use this read-only property with the **ExecutionState** property. It returns the last error string generated by the provider.

This property shows a description of errors that were encountered while parsing the connection string. The string comes from exceptions thrown by the Microsoft object **or data provider objects.**

Syntax

```
string LastError
```

Public enumerations

The following public enumerations apply to the **aaDBConnection** objects.

aaDBConnectionState

This public enumeration determines the state of the **aaDBConnection** object. Each value reflects the current state of the connection in the DCM.

Disconnected

Disconnected is the state of the **aaDBConnection** object immediately after it transitions through Connected or fails to connect. The disconnect timing is controlled by the underlying ADO.Net communication pooling logic. The **Disconnected** state remains active until the DCM is required to create a physical connection in response to a command.

A previous connected state can transition to a disconnected state if a command is run and the physical connection has been lost.

Connecting

This value indicates a transition state. It indicates that the DCM has started a physical connection to a database based on its internal connection pooling logic and the connection is not complete or has not yet failed.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Connected

This value indicates that the DCM has established a physical connection to a database in response to a command.

Created

This value indicates the initial state for a connection after creation until a transaction or command is run on it. After the state changes to something else, the state can never return to **Created**.

Disposed

This value indicates that **Dispose()** has been called for the connection.

aaDBConnectionType

This public enumeration determines the type of connection that an **aaDBConnection** object is being created for.

Sql

This value indicates a connection for a Microsoft SQL Server database.

OleDb

This value indicates a connection using Microsoft OLEDB.

Oracle

This value indicates a connection for an Oracle database server.

aaDBRow object

This section describes the aaDBRow object and the methods and properties that you can use with it.

Use objects of type **aaDBRow** to access (set or get) a single row from the memory **dataset** generated while an **aaDBCommand object** is running.

An instance of the **aaDBRow** object is returned by **aaDBCommand.**GetRow(). SetCurrentRow() also requires an instance of this type as an input parameter.

You can modify the **aaDBRow** object returned by **GetRow()** and feed it back to **SetCurrentRow()**, or you can construct a new **aaDBRow** object specifically for **SetCurrentRow()**.

The **aaDBRow** object has two public members both of type **ArrayList**:

- **columnName**
- **columnValue**

Note: You can use this type to update a row in the memory table by configuring only a subset of the

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

columns.

The two arrays must have the same size.

Note: You must call **Dispose()** on each instance of aaDBRow.

Methods

You can use the following methods with the **aaDBRow** object.

GetColumnName()

Use the **GetColumnName()** method to retrieve the name of the column specified by ColumnNum.

Syntax

```
string GetColumnName(
int ColumnNum)
```

Return value

If ColumnNum is equal to or greater than the number of columns stored in the row, as returned by the ColumnCount property, null is returned.

GetColumnValue()

Use the **GetColumnValue()** method to retrieve the value of the column specified by ColumnNum.

Syntax

```
object GetColumnValue(
int ColumnNum)
```

Return value

If ColumnNum is equal to or greater than the number of columns stored in the row, as returned by the ColumnCount property, null is returned.

Note: The value is returned as a generic object and must be cast by the script to the intended type.

Properties

You can use the following properties with the **aaDBRow** object.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

ColumnCount

Use this read-only property to retrieve the number of columns stored in the row object.

Syntax

```
int ColumnCount
```

ColumnNames

Use this read-only property to retrieve the ArrayList that internally stores the column names for the row.

Syntax

```
ArrayList ColumnNames
```

ColumnValues

Use this read-only property to retrieve the ArrayList that internally stores the column values for the row.

Syntax

```
ArrayList ColumnValues
```

aaDBTransaction object

This section explains how to use the **aaDBTransaction** object and the methods and properties associated with it.

Use objects of type **aaDBTransaction** to process multiple **aaDBCommand objects** as a single unit.

Create instances of type **aaDBTransaction** by calling the **CreateTransaction()** method on an instance of the **aaDBConnection object**. For example, assuming that the **aaDBConnection** instance is called **Connection**:

```
Connection.CreateTransaction()
```

If you want to ensure that all SQL commands are run as a whole or not run at all, you must create an instance of the **aaDBTransaction** object. Use the instance to create **aaDBCommand** objects.

When the transaction object is run, all **aaDBCommand** objects that are added to this **aaDBTransaction** object are run in the order that they were added. If the transaction is rolled back, none of the objects are run.

Commands added to a transaction cannot be run as stand-alone command objects but are automatically processed when the transaction runs.

When a script requests the transaction object ID, the transaction object is flagged to be persisted. The SQLData Script Library persists the object across scripts and scan cycles.

You can retrieve an **aaDBTransaction object** at any time by calling the static method **aaDBAccess.GetTransaction ()** and passing the previously acquired string ID.

Note: You must call **Dispose()** on each instance of **aaDBTransaction**.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Methods

You can use the following methods with the **aaDBTransaction** object.

CreateCommand()

Use the **CreateCommand()** method to create a new **aaDBCommand** object. You can configure the new object by adding parameters and then running it.

Syntax

```
aaDBCommand CreateCommand(
string CommandText,
aaDBCommandType CommandType,
bool ReturnDataset)
```

Remarks

For more details, see aaDBConnection Object.

Dispose()

Use the **Dispose()** method to instruct the SQLData Script Library to free all resources allocated for the transaction object and all commands that were added as part of this transaction.

Syntax

```
void Dispose()
```

Remarks

If the transaction is currently running, **Dispose()** automatically cancels the transaction before removing it.

The SQLData Script Library issues **Dispose()** calls for every command in the transaction.

If an ID has been retrieved for this command, you must call Dispose() or aaDBAccess.RemoveCommand().

ExecuteAsync()

Use the **ExecuteAsync()** method to instruct the SQLData Script Library to queue all command objects in the transaction that are queued for later processing. **ExecuteAsync()** returns immediately and processing occurs in the background.

Syntax

```
Result ExecuteAsync()
```

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Remarks

You can use the ExecutionState property to check for status.

If any commands complete with an error, the SQLData Script Library issues a transaction rollback to prevent changes to the data source.

After **ExecuteAsync()** is processed, you can still obtain a reference to any commands that were added and analyze their ExecutionState and LastExecutionError properties.

When the transaction object runs, all **aaDBCommand** objects that are added to the **aaDBTransaction** are run in the order that they were added. The following outcomes can occur as the result of processing a transaction:

- **Success**: All commands succeeded. Each command shows an **ExecutionState** of **Completed**. Any dataset for that command is associated with it.
- **Cancellation**: Each command in the transaction shows an **ExecutionState** of **Canceled**. Any data associated with the command is removed.
- **Failure**: When one command in a transaction fails, the command has an **ExecutionState** of **Failed**. All commands preceding it show an **ExecutionState** of **Completed** and retain any datasets that were part of their successful processing.

To free all resources allocated for the transaction object and the commands, you must call **Dispose()**.

ExecuteAsyncCancel()

Use the **ExecuteAsyncCancel()** method to instruct the SQLData Script Library to roll back all commands that are queued as part of a transaction.

Syntax

```
Result ExecuteAsyncCancel()
```

Remarks

You can use the ExecutionState property to check for status.

As a result of successful execution, all command objects created from this transaction are canceled.

ExecuteSync()

Use the **ExecuteSync()** method to instruct the SQLData Script Library to run all commands that are queued as part of a transaction.

Syntax

```
Result ExecuteSync()
```

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Remarks

Use the ExecutionState and **LastExecutionError** properties to check for status of this method.

If any one of the commands completes with an error, the SQLData Script Library issues a rollback action to guarantee that the data source does not see any of the changes.

When the transaction object runs, all **aaDBCommand** objects that are added to the **aaDBTransaction** are run in the order that they were added. The following outcomes can occur as the result of processing a transaction:

- **Success**: All commands succeeded. Each command shows an **ExecutionState** of **Completed**. Any dataset for that command is associated with it.
- **Cancellation**: Each command in the transaction shows an **ExecutionState** of **Canceled**. Any data associated with the command is removed.
- **Failure**: When one command in a transaction fails, that command object has an **ExecutionState** of **Failed**. All command objects preceding it show an **ExecutionState** of **Completed** and retain any datasets that were part of their successful processing.

After **ExecuteSync()** is processed, you can still obtain a reference to the command objects that were added and analyze their **ExecutionState** and **LastExecutionError** properties.

Use the **LastExecutionError** property to check if the command failed or succeeded (success is indicated by a blank string), since ExecuteSync() may return 0 in the case of a failure instead of an error message.

To free all resources allocated for the **aaDBTransaction** object and the **aaDBCommand** objects you must call **Dispose()**.

GetID()

Use the **GetID()** method if you want to retrieve the ID of an **aaDBTransaction** object instance to get a reference to this object at a later time in a different script or scan.

The SQLData Script Library generates a unique transaction ID and persists the transaction object in memory.

Syntax

```
string GetID()
```

Return value

If a failure occurs, the **GetID()** method returns **null**.

Note: The SQLData Script Library returns a string value, but the script engine automatically attempts to cast this value to other types. If the script assigns the returned ID to any other type than string, the ID is corrupted and does not work in future GetTransaction(ID) calls.

Properties

Use the following properties with the **aaDBTransaction** object.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Disposed

This read-only property indicates whether **Disposed()** has been called for a transaction object. If the property returns **True**, no other method or property can be called on this instance.

Syntax

```
Bool Disposed
```

ExecutionState

Use this read-only property to return the state of this transaction object.

Because the processing of the transaction is asynchronous, you must determine if it has finished processing before you request the results.

Syntax

```
aaDBTransactionState ExecutionState
```

FailedCommandID

Use this property to return the ID of the first failed command object during transaction processing.

If the processing action succeeds, **FailedCommandID = 0**.

Syntax

```
string FailedCommandID
```

LastExecutionError

During processing, this property is set to the last error, if any. The error string is the error string generated by the provider.

Use this property in conjunction with the ExecutionState property.

LastExecutionError indicates the error generated by the provider for the first command object in the transaction that failed.

To find the command object that failed to process, check the FailedCommandID property.

LastExecutionError is blank when the transaction runs successfully.

Syntax

```
string LastExecutionError
```

Public enumeration

The following public enumeration applies to the **aaDBTransaction** object.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

aaDBTransactionState

This public enumeration indicates the state of an **aaDBTransaction**.

Created

This value indicates that a transaction has been created. The transaction has not yet run. You can add commands to the transaction.

Queued

This value indicates that the transaction has been queued to run. This state occurs immediately after an **aaDBTRansaction.ExecutedAsync()** or **aaDBTRansaction.ExecutedSync()** call. **Queued** is a transitional state that changes to either **Failed** or **Completed**.

Failed

This value indicates that the transaction has failed.

Completed

This value indicates that the transaction completed successfully.

Canceled

This value indicates that the transaction was canceled before it could complete.

Disposed

This value indicates that the **Dispose()** method has been called for the command object. **Disposed** is automatically set when **aaDBAccess.RemoveTransaction()** is called or when the connection object that owns the transaction is disposed.

Error codes

Various API scripting methods return an error code as a numeric value.

Methods with syntax descriptions that begin with "result" return these numeric values.

This table shows the numeric value, its corresponding error, and meaning of each error code.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Numeric value	Error text and description
-1	**Unknown Failure** This value indicates an exception whose reason could not be determined at run time.
0	**Success** This value indicates a good result; no error occurred.
100	**SoftwareError** This indicates that the internal code returned an unexpected result, but the case was handled.
1000	**StatementNotReady** This error is returned when an **aaDBCommand** object is run, but it is not properly prepared for processing.
1001	**StatementFailed** This error is returned when an **aaDBCommand** object runs and fails to complete properly. This result occurs when the SQL query is malformed.
1002	**DatasetIsNull** This error is returned when the results of an **aaDBCommand** object are being examined or manipulated, but no dataset is associated with the object. This message can mean that the command has not yet run, the SQL query was malformed, or the object was not of a type to return data.
1003	**DatasetIsEmpty** This error occurs when the currently selected row or table number is beyond the end of the data in the dataset that is associated with the **aaDBCommand** object.
1004	**InvalidConnection** This error is returned when an **aaDBCommand** attempts to run and the connection object it is attached to has not established a good connection to the database.
1005	**InvalidRowNum** This error is returned by **aaDBCommand.SelectRow()** when the row number specified is negative or larger than the number of rows in the dataset that is

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Numeric value	Error text and description
	associated with the **aaDBCommand** object.
1006	**InvalidColValue** This error is returned when a column is examined or manipulated by the index and the index does not represent a valid column number.
1007	**InvalidTableNum** This error is returned by **aaDBCommandSelectTable()** when the table number does not represent a table that is currently stored in the dataset that is associated with the **aaDBCommand** object.
1008	**MissingParameter** This error is returned when an OLEDB style **aaDBCommand** object is run and not all parameter indexes were supplied.
1009	**NamedParametersNotSupported** This error is returned when an attempt is made to call **SetXXXParameterByName()** for a non-OLEBB style **aaDBCommand** object.
1010	**ParameterNameIsRequired** This error is returned when an attempt is made to call **SetXXXParameterByIndex()** for a non-OLEBB style **aaDBCommand** object.
1011	**InvalidRequestOperationInProgress** This error is returned when an attempt is made to run an **aaDBCommand** object that is currently processing. If a command or transaction is currently in progress through **ExecuteAsync()**, that same object cannot be executed again with **ExecuteAsync()** or **ExecuteSync()** until the previous process is complete. Only a single object can be in the processing queue at any time.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Numeric value	Error text and description
1012	**SaveChangesNotSupportedForStoredProcedure** This error is returned when either **SaveChangesSync()** or **SaveChangesAsync()** is called after manipulating the dataset associated with an **aaDBCommand** object whose query was originally against a stored procedure.
1013	**InvalidRequestNotSupportedInCurrentState** This error is returned when the **aaDBCommand** object is not in a state where the requested run or cancel can be honored.
1014	**InvalidRequestPartOfTransaction** This error indicates that you attempted to run or save changes to an **aaDBCommand** object that was created to be part of an **aaDBTransaction**. This activity is not permitted.
1015	**DCMObjectInvalid** This error indicates that the DCM cannot provide an object to support **aaDBConnection**, **aaDBCommand**, **aaDBTransaction**.
1016	**ObjectIsDisposed** The script attempted to use an object after calling **Dispose().**

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

API and extensions reference

The topics in this section describe the API toolkits and IDE extensions available with Application Server.

GRAccess Toolkit

The AVEVA™ GRAccess object model provides a way to automate configuration of local and remote Galaxies. The Galaxy configuration is exposed to .NET and COM clients through a programmable object mode.

How to use the GRAccess Toolkit

You can use the AVEVA™ GRAccess object model to write programs that automate configuration of local and remote Galaxies. The Galaxy configuration is exposed to .NET and COM clients through a programmable object model called GRAccess.

Target clients

C# is the recommended programming language for GRAccess. GRAccess also works with other COM or .NET programming languages such as VB6, C++, and VB.NET.

GRAccess provides a .NET primary interop assembly for use in Microsoft .NET languages, and a COM type library that can be used in VB6 or other COM clients.

Microsoft Visual Studio

Microsoft Visual Studio is required. The version you need depends on the applications you will be creating:

- Microsoft Visual Studio 2012 Update 1 or later is required for .NET applications
- Microsoft Visual Studio 2019 is required for C++ applications.

You must run with administrative privileges in order to build the projects using Visual Studio or run an application (.exe) built using the GRAccess Toolkit.

Configuration files

When you create a GRAccess application, a configuration file for application should be created in the same location as the GRAccess application (EXE). The configuration file name should be in the format:
<ApplicationName>.exe.config

Add the following xml into the file, under:
configuration > runtime > assemblyBinding

```
<configuration>
<startup useLegacyV2RuntimeActivationPolicy="true">
```

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

```
<requiredRuntime version="v2.0.50727" safemode="true"/>
<supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.7.1"/>
</startup>
<runtime>
<assemblyBinding>
<dependentAssembly>
<assemblyIdentity name="ArchestrA.GRAccess"
publicKeyToken="23106a86e706d0ae"
culture=""/>
<bindingRedirect oldVersion="1.0.0.0-1.9.0.0" newVersion="2.0.0.0"/>
</dependentAssembly>
</assemblyBinding>
</runtime>
</configuration>
```

For more information about assembly binding, refer to:

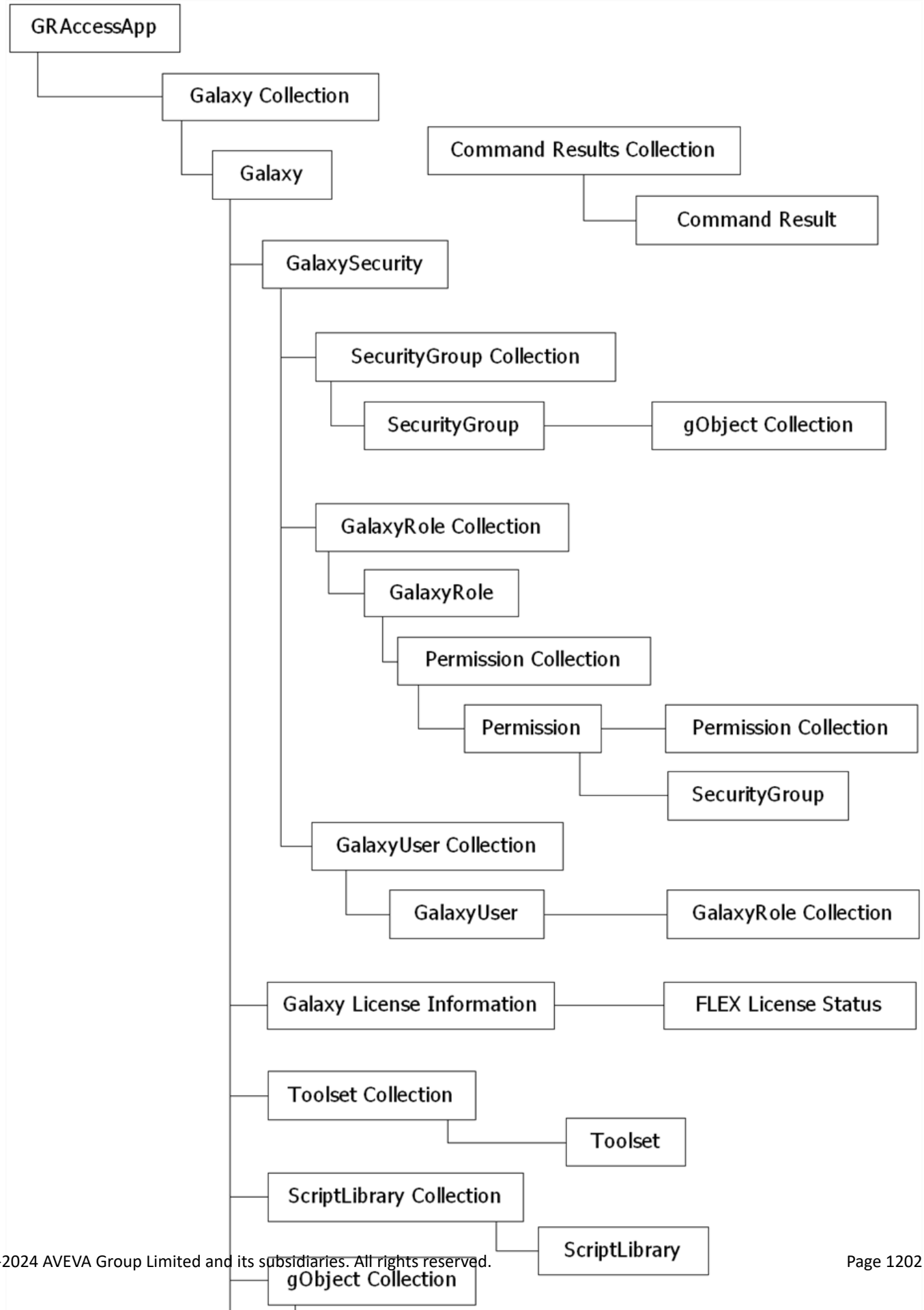
https://docs.microsoft.com/en-us/dotnet/framework/configure-apps/redirect-assembly-versions#specifying-assembly-binding-in-configuration-files

GRAccess object model

GRAccess provides a hierarchical object model. At the root of this object model is the GRAccessAppClass. The GRAccessAppClass provides access to other ArchestrA objects including Galaxies, gObject Templates and Instances.

Typically, only the GRAccessAppClass and the MxValue objects are created directly. Other objects are obtained by navigating the GRAccessAppClass object object model.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.



© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

GRAccess methods to set license mode

Galaxies can be configured to use one of two license modes: either Flex, which requires an annual subscription, or perpetual, which is a traditional one-time purchase of a software license.

Flex licenses are applied to platform objects. The Flex licenses are single-engine or multi-engine, meaning that a Flex-configured platform object is allowed to have either a single AppEngine assigned to it, or if a multi-engine is applied, the licensing does not constrain the number of AppEngines.

GRAccess contains two methods to support Flex licenses programmatically.

- **PlatformLicenseType:** This is a IgObject property that can be used to retrieve or update (get/set) the Flex license type of a platform.
- **FlexLicensesInformation:** This is an IGalaxy property that can be used to obtain the status, including type and number, of Flex licenses that are available to the Galaxy Repository. Status is returned as an array and includes:
  - License type: single engine or multi-engine
  - Number of licenses available to the GR
  - Number of licenses deployed by the GR
  - Number of licenses configured for the GR
  - Number of licenses available for deployment by the GR

API reference

You can use the GRAccess API to write programs that manipulate the configurations of local and remote Galaxies.

COM classes

This section describes the COM classes that are defined and exposed through GRAccess. These include:

- GRAccess
- GRAccessApp
- MxValue

GRAccess

GRAccess is similar to GRAccessApp but is cocreated in-process to the client. Restrictions include:

- EXE program hosting the component must reside in the AppServer Bin folder.
- Single-stepping over GRAccess method calls with the Visual Basic 6.0 debugger may cause execution to run-away and not stop at the next line.

Interfaces implemented

IgObject

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

GRAccessApp

Clients use the GRAccess Component to gain access to the Galaxy Repository.

Interfaces implemented

IgObject

MxValue

The MxValue provides storage and retrieval of System Platform data types.

MxValues are passed as parameters in the AutomationObject Get and Set methods.

Interfaces implemented

IMxValue

Interfaces

This section describes the COM interfaces that are defined and exposed through GRAccess. These interfaces include:

Objects	Collections of
IAssociatedFileAccess	
IAttribute	IAttributes
ICommandResult	ICommandResults
ICondition	IConditions
IGalaxy	IGalaxies
IGalaxyRole	IGalaxyRoles
IGalaxySecurity	
IGalaxyUser	IGalaxyUsers
IGraphicAccess	
IInstance	
IMxValue	
IgObject	IgObjects
IPermission	IPermissions

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

IScriptLibrary	IScriptLibraries
ISecurityGroup	ISecurityGroups
ISettings	
ITemplate	
IToolset	IToolsets

IAssociatedFileAccess

Internal use only.

IAttribute

Represents a single attribute of an object.

IAttribute class members

This class has the following members.

Operations

CommandResult property

Returns the CommandResult Object, which has the last method call's result.

RtSethandler property

Returns the RtSethandler property of the Attribute.

CfgSethandler property

Returns the CfgSethandler property of the Attribute.

SecurityClassification property

Returns the SecurityClassification property of the Attribute.

Name property

Returns the Name property of the Attribute.

DataType property

Returns the Type property of the Attribute.

SetLocked method

Sets the Locked property of the Attribute.

UpperBoundDim1 property

Returns the UpperBoundDim1 property of the Attribute.

SetSecurityClassification method

Sets the SecurityClassification property of the Attribute.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Locked property

Returns the Locked property of the Attribute.

SetValue method

Sets the Value of the Attribute.

Value property

Returns the Value of the Attribute.

AttributeCategory property

Returns the Category of the Attribute.

HasBuffer property

Returns the HasBuffer property of the Attribute.

SetHasBuffer method

Sets the HasBuffer property of the Attribute.

CommandResult property

Returns the CommandResult object, which has the last method call's result.

Class

IAttribute

Syntax

[C#]

```
ICommandResult CommandResult { get; };
```

[Visual C++]

```
HRESULT CommandResult(
   [out, retval] ICommandResult** CommandResult
);
```

Parameters

CommandResult

Returns the CommandResult object.

RtSethandler property

Returns the RtSethandler property of the attribute.

Class

IAttribute

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Syntax

[C#]

```
bool RtSethandler { get; };
```

[Visual C++]

```
HRESULT RtSethandler(
   [out, retval] VARIANT_BOOL* hasRtSethandler
);
```

Parameters

hasRtSethandler

Returns the RtSethandler property of the attribute.

CfgSethandler property

Returns the CfgSethandler property of the attribute.

Class

IAttribute

Syntax

[C#]

```
bool CfgSethandler { get; };
```

[Visual C++]

```
HRESULT CfgSethandler(
   [out, retval] VARIANT_BOOL* hasCfgSethandler
);
```

Parameters

hasCfgSethandler

Returns the CfgSethandler property of the attribute.

SecurityClassification property

Returns the SecurityClassification property of the attribute.

Class

IAttribute

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Syntax

[C#]

```
MxSecurityClassification SecurityClassification { get; };
```

[Visual C++]

```
HRESULT SecurityClassification(
   [out, retval] MxSecurityClassification* securityClassification
);
```

Parameters

securityClassification

Returns the SecurityClassification property of the attribute.

Name property

Returns the name property of the attribute.

Class

IAttribute

Syntax

[C#]

```
string Name { get; };
```

[Visual C++]

```
HRESULT Name(
   [out, retval] BSTR* name
);
```

Parameters

name

Returns the name of the attribute.

DataType property

Returns the type property of the attribute.

Class

IAttribute

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Syntax

[C#]

```
MxDataType DataType { get; };
```

[Visual C++]

```
HRESULT DataType(
   [out, retval] MxDataType* dataType
);
```

Parameters

dataType

Returns the type property of the attribute.

SetLocked method

Sets the locked property of the attribute.

Class

IAttribute

Syntax

[C#]

```
void SetLocked(
MxPropertyLockedEnum NewMxPropertyLockedEnum
);
```

[Visual C++]

```
HRESULT SetLocked(
   [in] MxPropertyLockedEnum
);
```

Parameters

MxPropertyLockedEnum

The new locked property of the attribute.

UpperBoundDim1 property

Returns the UpperBoundDim1 property of the attribute.

Class

IAttribute

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Syntax

[C#]

```
short UpperBoundDim1 { get; };
```

[Visual C++]

```
HRESULT UpperBoundDim1(
   [out, retval] short* upperBoundDim1
);
```

Parameters

upperBoundDim1

Returns the UpperBoundDim1 property of the attribute.

SetSecurityClassification method

Sets the SecurityClassification property of the attribute.

Class

IAttribute

Syntax

[C#]

```
void SetSecurityClassification(
MxSecurityClassification NewMxSecurityClassification
);
```

[Visual C++]

```
HRESULT SetSecurityClassification(
   [in] MxSecurityClassification
);
```

Parameters

MxSecurityClassification

The new SecurityClassification property of the attribute.

Locked property

Returns the Locked property of the attribute.

Class

IAttribute

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Syntax

[C#]

```
MxPropertyLockedEnum Locked { get; };
```

[Visual C++]

```
HRESULT Locked(
   [out, retval] MxPropertyLockedEnum*
);
```

Parameters

MxPropertyLockedEnum

Returns the Locked property of the attribute.

SetValue method

Sets the value of the attribute.

Class

IAttribute

Syntax

[C#]

```
void SetValue(
MxValue NewValue
);
```

[Visual C++]

```
HRESULT SetValue(
   [in] IMxValue* NewValue
);
```

Parameters

NewValue

The new value of the attribute.

Value property

Returns the value of the attribute.

Class

IAttribute

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Syntax

[C#]

```
MxValue value { get; };
```

[Visual C++]

```
HRESULT Value(
   [out, retval] IMxValue** value
);
```

Parameters

value

Returns the value of the attribute.

AttributeCategory property

Returns the category of the attribute.

Class

IAttribute

Syntax

[C#]

```
MxAttributeCategory AttributeCategory { get; };
```

[Visual C++]

```
HRESULT AttributeCategory(
   [out, retval] MxAttributeCategory* attributeCategory
);
```

Parameters

attributeCategory

Returns the category of the attribute.

HasBuffer property

Returns the HasBuffer property of the attribute.

Class

IAttribute

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Syntax

[C#]

```
bool HasBuffer { get; };
```

[Visual C++]

```
HRESULT HasBuffer(
   [out, retval] VARIANT_BOOL* hasBuffer
);
```

Parameters

hasBuffer

Returns the HasBuffer property of the attribute.

SetHasBuffer method

Sets the HasBuffer property of the attribute.

Class

IAttribute

Syntax

[C#]

```
void SetHasBuffer(bool hasBuffer);
```

[Visual C++]

```
HRESULT SetHasBuffer(
   [in] VARIANT_BOOL hasBuffer
);
```

Parameters

hasBuffer

The new value of the attribute.

IAttributes

Represents a collection of attribute objects.

IAttributes class members

This class has the following members.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Operations

Item property

Returns the Attribute object by name or index.

Count property

Returns the number of attributes in the collection.

ShortDescription property

Returns the short description for this object.

ExecutionOrder property

Returns the name of the execution order of this object.

SecurityGroup property

Returns the securitygroup for this object.

ExecutionRelatedObject property

Returns the name of the execution related object of this object.

Item property

Return the attribute object by name or index.

Class

IAttributes

Syntax

[C#]

```
IAttribute this[object attributeIdentifier] { get; };
```

[Visual C++]

```
HRESULT Item(
   [in] VARIANT attributeIdentifier,
   [out, retval] IAttribute**
);
```

Parameters

attributeIdentifier

The name of the attribute object to return if a string is specified. If a number is specified, it is interpreted as the index into this collection and is in the range from 1 to count.

IAttribute

The returned attribute object or nothing if attribute doesn't exist.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Count property

Returns the number of attributes in the collection.

Class

IAttributes

Syntax

[C#]

```
int count { get; };
```

[Visual C++]

```
HRESULT Count(
   [out, retval] LONG* count
);
```

Parameters

count

The number attributes in the collection.

ShortDescription property

Returns the short description of this object.

Class

IAttributes

Syntax

[C#]

```
string ShortDescription { set; get; };
```

[Visual C++]

```
[propget]
HRESULT ShortDescription(
   [out, retval] BSTR*
);
[propput]
HRESULT ShortDescription(
   [in] BSTR
);
```

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

ExecutionOrder property

Returns the name of the execution order of this object.

Class

IAttributes

Syntax

[C#]

```
EExecutionOrder ExecutionOrder { set; get; };
```

[Visual C++]

```
[propget]
HRESULT ExecutionOrder(
   [out, retval] EExecutionOrder*
);
[propput]
HRESULT ExecutionOrder(
   [in] EExecutionOrder
);
```

SecurityGroup property

Retrieves the security group for this object.

Class

IAttributes

Syntax

[C#]

```
string SecurityGroup SecurityGroup { set; get; };
```

[Visual C++]

```
[propget]
HRESULT SecurityGroup(
   [out,retval] BSTR*
);
[propget]
HRESULT SecurityGroup(
   [in] BSTR*
);
```

ExecutionRelatedObject property

Returns the name of the execution-related object of this object.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Class

IAttributes

Syntax

[C#]

```
string ExecutionRelatedObject { set; get; };
```

[Visual C++]

```
[propget]
HRESULT ExecutionRelatedObject(
   [out, retval] BSTR*
);
[propput]
HRESULT ExecutionRelatedObject(
   [in] BSTR newExecutionRelatedObject
);
```

Parameters

newExecutionRelatedObject

The name of the execution-related object.

ICommandResult

Indicates the result of a given operation (for instance, success, fail and why).

ICommandResult Class Members

This class has the following members.

Operations

Successful property

Indicates whether the operation was successful.

Text property

Textual description corresponding to the result code.

ID property

Reason code for the result.

CustomMessage property

Custom message associated with the result.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Successful property

Indicates whether the operation was successful.

Class

ICommandResult

Syntax

[C#]

```
bool Successful { get; };
```

[Visual C++]

```
HRESULT Successful( [out, retval] VARIANT_BOOL* );
```

Text property

Textual description corresponding to the result code.

Class

ICommandResult

Syntax

[C#]

```
string Text { get; }
```

[Visual C++]

```
HRESULT Text( [out, retval] BSTR* );
```

ID property

Reason code for the result.

Class

ICommandResult

Syntax

[C#]

```
EGRCommandResult ID { get; };
```

[Visual C++]

```
HRESULT ID( [out, retval] EGRCommandResult* );
```

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

CustomMessage property

Custom message associated with the result.

Class

ICommandResult

Syntax

[C#]

```
string CustomMessage { get; };
```

[Visual C++]

```
HRESULT CustomMessage( [out, retval] BSTR* );
```

ICommandResults

Represents a collection of CommandResult objects.

ICommandResults class members

This class has the following members.

Operations

Count property

Returns the number of CommandResults in the collection.

Item method

Returns the CommandResult object by index.

CompletelySuccessful property

Indicates that all CommandResult objects in the collection are successful.

Count property

Returns the number of CommandResult objects in the collection.

Class

ICommandResults

Syntax

[C#]

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

```
int count { get; };
```

[Visual C++]

```
HRESULT Count(
   [out, retval] LONG*
);
```

Item method

Returns the CommandResult object by index.

Class

ICommandResults

Syntax

[C#]

```
ICommandResult get_Item(
int index
);
```

[Visual C++]

```
HRESULT Item(
   [in] LONG index,
   [out, retval] ICommandResult** customResult
);
```

Parameters

index

The Index of the CommandObject. This is a numeric value from 1 to count.

customResult

The returned CommandResult object.

CompletelySuccessful property

Indicates that all CommandResult objects in the collection are successful.

Class

ICommandResults

Syntax

[C#]

```
bool CompletelySuccessful { get; };
```

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

[Visual C++]

```
HRESULT CompletelySuccessful(
   [out, retval] VARIANT_BOOL* completelySuccessful
);
```

Parameters

completelySuccessful

Indicates that all CommandResult objects in the collection are successful.

ICondition

Indicates a condition used when querying for a list of objects.

ICondition class members

This class has the following members.

Operations

Kind property

Indicates the type of condition to be used by the query.

Value property

The value to be searched for. Meaning is based on the *Kind* parameter.

Negation property

Negation. Specify False to "NOT" the expression.

Kind property

Indicates the type of condition to be used by the query.

Class

ICondition

Syntax

[C#]

```
EConditionType Kind { get; };
```

[Visual C++]

```
HRESULT Kind( [out, retval] EConditionType* Kind );
```

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Value property

The value to be searched for. Meaning is based on the Kind parameter.

Class

ICondition

Syntax

[C#]

```
object value { get; };
```

[Visual C++]

```
HRESULT Value( [out, retval] VARIANT* Value );
```

Negation property

Negation. Specify False to "NOT" the expression.

Class

ICondition

Syntax

[C#]

```
EConditionType Negation { get; };
```

[Visual C++]

```
HRESULT Negation([out, retval] VARIANT_BOOL* Value );
```

IConditions

Represents a collection of condition objects.

IConditions class members

This class has the following members.

Operations

CommandResult property

Returns the CommandResult Object, which has the last method call's result.

Remove method

Removes a condition object from the collection.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Add method

Adds a condition object to the collection.

Count property

Returns the number of condition objects in the collection.

Item property

Returns the condition object by index.

Join method

Joins two condition objects.

CommandResult property

Returns the CommandResult object, which has the last method call's result.

Class

IConditions

Syntax

[C#]

```
ICommandResult CommandResult { get; };
```

[Visual C++]

```
HRESULT CommandResult(
   [out, retval] ICommandResult** CommandResult
);
```

Parameters

CommandResult

Returns the CommandResult object.

Remove method

Remove a condition object from the collection.

Class

IConditions

Syntax

[C#]

```
void Remove(
   int index
```

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

```
);
```

[Visual C++]

```
HRESULT Remove(
   [in] LONG index
);
```

Parameters

index

The index of the condition object. This is a numeric value from 1 to count.

Add method

Add a condition object to the collection.

Class

IConditions

Syntax

[C#]

```
void Add(
   EConditionType Kind,
   object value,
   bool Negation
);
```

[Visual C++]

```
HRESULT Add(
   [in] EConditionType Kind,
   [in] VARIANT Value,
   [in, optional, defaultvalue( -1 )]
   VARIANT_BOOL negation
);
```

Parameters

Kind

Indicates the type of condition to be used by the query.

Value

The value to be searched for. Meaning is based on the *Kind* parameter.

negation

Negation. To NOT the expression, pass False for negation.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Count property

Returns the number of condition objects in the collection.

Class

IConditions

Syntax

[C#]

```
int count { get; };
```

[Visual C++]

```
HRESULT Count(
   [out, retval] LONG*
);
```

Item property

Returns the condition object by index.

Class

IConditions

Syntax

[C#]

```
ICondition this[int index] { get; };
```

[Visual C++]

```
HRESULT Item(
   [in] LONG index,
   [out, retval] ICondition**
);
```

Parameters

index

The index of the condition object. This is a numeric value from 1 to Count.

ICondition

The returned condition object.

Join method

Join two conditions.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Class

IConditions

Syntax

[C#]

```
void Join(
   IConditions conditions
);
```

[Visual C++]

```
HRESULT Join(
   [in] IConditions* conditions
);
```

Parameters

conditions

The returned condition object.

IGalaxy

Represents a Galaxy.

IGalaxy class members

This class has the following members.

Operations

CommandResult property

Returns the CommandResult object, which has the last method call's result.

QueryObjectsMultiCondition method

Retrieves a gObjects collection using multiple conditions.

QueryScriptLibraries method

Query Script Libraries.

Name property

Retrieves the name of the Galaxy.

Backup method

Backs-up the given Galaxy to a backup file.

VersionString property

Retrieves the version string of the Galaxy.

Login method

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Logs in to an ArchestrA Galaxy and performs a forced synchronization of the client with the Galaxy Repository. This method must be called before any other Galaxy method.

LoginEx method

Logs in to an ArchestrA Galaxy and offers the option to perform synchronization at a later point in time and cancel synchronization. This method must be called before any other Galaxy method.

LoginWithAccessMode method

Logs into a Galaxy. This method must be called before any other Galaxy method. If GRAccess is already logged into a Galaxy, the session is logged off and restablished with the specified Galaxy.

SynchronizeClient method

Synchronizes the client with the server.

GetTimeMasterSettings method

Retrieves TimeMaster settings.

CreategObjectCollection method

Creates an empty gObjects collection object

ImportObjects method

Imports objects into the Galaxy.

ImportObjectsEx method

Imports objects into the Galaxy.

ImportObjectsEx2 method

Imports one or more of the objects contained in an aaPDF or aaPKG file into the Galaxy Repository.

ImportArchestrAApp method

Imports AVEVA OMI apps from a specified folder.

QueryToolsets method

Retrieves the list of available Toolsets.

QueryObjectsByName method

Retrieves a gObjects collection based on a set of tagnames.

GetUserDefaults method

Get the logged in user's defaults from the Galaxy repository.

VersionNumber property

Retrieves the version number of the Galaxy.

SetUserDefaults method

Set the logged in user's defaults.

Restore method

Restores a Galaxy from a backup file.

Logout method

Logs out from an ArchestrA Galaxy.

UpgradeRequired property

Retrieves the Boolean indicated if the Galaxy needs to be upgraded.

MigrateGalaxy method

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Upgrade a Galaxy.

CdiVersionString property

Retrieves the version string of the Cdi.

GetSecuritySettings method

Retrieves the Galaxy's security settings.

CreateConditionsObject method

Creates a condition object used with QueryObjectsMultiCondition.

QueryObjects method

Retrieves a gObjects collection based on the given criteria.

ImportScriptLib method

Imports this script library.

ImportScriptLibEx method

Imports a script library (aaSLib file).

GetLocaleSettings method

Retrieves the Galaxy's local settings.

ExportAll method

Exports all objects in the Galaxy.

GetReadOnlySecurity method

Accesses the Galaxy security in read-only mode.

CommandResults property

Returns a collection of CommandResult objects.

GRLoad method

Loads the Galaxy from a CSV file.

GetFlexLicensesInformation method

Returns a collection of Flex license information for single and multi-engine platforms.

IGalaxyQueryToolsetsEx method

Retrieves the list of available Toolsets.

CommandResult property

Returns the CommandResult object, which has the last method call's result.

Class

IGalaxy

Syntax

[C#]

```
ICommandResult CommandResult { get; };
```

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

[Visual C++]

```
HRESULT CommandResult(
   [out, retval] ICommandResult** CommandResult
);
```

Parameters

CommandResult

Returns the CommandResult object.

QueryObjectsMultiCondition method

Retrieves a gObjects collection using multiple conditions.

Class

IGalaxy

Syntax

[C#]

```
IgObjects QueryObjectsMultiCondition(
   EgObjectIsTemplateOrInstance templateOrInstance, IConditions conditions
);
```

[Visual C++]

```
HRESULT QueryObjectsMultiCondition(
   [in] EgObjectIsTemplateOrInstance templateOrInstance,
   [in] IConditions* Conditions,
   [out, retval] IgObjects**
);
```

Parameters

templateOrInstance

Indicates whether a gObject is a template or an instance.

Conditions

A condition collection object containing a list of conditions to be ANDed together.

IgObjects

Returns a gObjects collection.

QueryScriptLibraries method

Query Script Libraries.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Class

IGalaxy

Syntax

[C#]

```
IScriptLibraries QueryScriptLibraries();
```

[Visual C++]

```
HRESULT QueryScriptLibraries(
   [out, retval] IScriptLibraries** scriptLibraries
);
```

Parameters

scriptLibraries

The list of script libraries.

Name property

Retrieves the name of the Galaxy.

Class

IGalaxy

Syntax

[C#]

```
string Name { get; };
```

[Visual C++]

```
HRESULT Name(
   [out, retval] BSTR* Galaxy
);
```

Parameters

Galaxy

Returns the name of the Galaxy.

Backup method

Backs up the given Galaxy to a backup file.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Class

IGalaxy

Syntax

[C#]

```
void Backup(
   int lProcessId,
   string backupFilename,
   string GRNodeName,
   string Galaxy
);
```

[Visual C++]

```
HRESULT Backup
(
   [in]LONG lProcessId,
   [in] BSTR backupFilename,
   [in, optional] BSTR GRNodeName,
   [in, optional] BSTR Galaxy
);
```

Parameters

lProcessId

Proccess ID of the calling process.

backupFilename

Name of the backup file. File extension .cab is appended if not specified. If a folder name is specified, the contents of the folder is deleted and the backup files are placed directly in the folder.

GRNodeName

The name of the GR node (computer name). Blank or omitted means the current computer is the GR.

Galaxy

The name of the Galaxy (application database).

VersionString property

Retrieves the version string of the Galaxy.

Class

IGalaxy

Syntax

[C#]

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

```
string VersionString { get; };
```

[Visual C++]

```
HRESULT VersionString(
   [out, retval] BSTR* VersionString
);
```

Parameters

VersionString

Returns the version string of the Galaxy.

Login method

Logs in to an ArchestrA Galaxy and performs a forced synchronization of the client with the Galaxy Repository. This method must be called before any other Galaxy method.

Class

IGalaxy

Syntax

[C#]

```
void Login(
   string UserName, string Password
);
```

[Visual C++]

```
HRESULT Login(
   [in, optional] BSTR UserName,
   [in, optional] BSTR Password
);
```

Parameters

UserName

The name of the user.

- When **Galaxy** Authentication mode is configured, blank or omitted *UserName* means log in as "admin".
- When **OS User based** or **OS Group based** Authentication mode is configured, *UserName* is ignored and the current logged in user or current user group membership is used to authenticate the user.

Password

The user's password.

- When **Galaxy** Authentication mode is configured, blank or omitted *Password* means use a blank password.
- When **OS User based** or **OS Group based** Authentication mode is configured, *Password* is ignored and the

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

current logged in user or current user group membership is used to authentication the user.

Remarks

If GRAccess is already logged into a Galaxy, the session is logged off and re-established with the specified Galaxy.

LoginEx method

Logs in to an ArchestrA Galaxy and offers the option to perform synchronization at a later point in time and cancel synchronization. This method must be called before any other Galaxy method.

Class

IGalaxy

Syntax

[C#]

```
void LoginEx(
   string UserName, string Password, bool bForceSynchronization
);
```

[Visual C++]

```
HRESULT LoginEx(
   [in, optional] BSTR UserName,
   [in, optional] BSTR Password,
   [in, optional] VARIANT_BOOL bForceSynchronization
);
```

Parameters

UserName

The name of the user.

- When **Galaxy** Authentication mode is configured, blank or omitted means log in as "admin".
- When **OS User based** or **OS Group based** Authentication mode is configured, *UserName* is ignored and the current logged in user or current user group membership is used to authenticate the user.

Password

The user's password.

- When **Galaxy** Authentication mode is configured, blank or omitted means use a blank password.
- When **OS User based** or **OS Group based** Authentication mode is configured, *Password* is ignored and the current logged in user or current user group membership is used to authenticate the user.

bForceSynchronization

Set to true to synchronize the client with the server if out of sync is detected.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Remarks

If GRAccess is already logged into a Galaxy, the session is logged off and re-established with the specified Galaxy.

LoginWithAccessMode method

Logs into a Galaxy. This method must be called before any other Galaxy method. If GRAccess is already logged into a Galaxy, the session is logged off and re-established with the specified Galaxy. This API does the following:

- Passes the Package Operation Mode while logging in.
- Avoids the need for GRAccess to log in to the Galaxy while configuring Security through the System Platform IDE. This method calls the IGalaxyBase::Login method.

Class

IGalaxy

Syntax

[C#]

```
void LoginWithAccessMode(string UserName], string Password, int bModeReadWrite = 0);
```

[Visual C++]

```
HRESULT LoginWithAccessMode
   (
      [in, optional] BSTR Username,
      [in, optional] BSTR Password,
      [in, defaultvalue(FALSE)] BOOL bModeReadWrite
   );
```

Parameters

Username

Name of the user. If Username is blank/omitted, log in as “admin.”

Password

Password associated with the Username. If password is blank/omitted, password is none.

bModeReadWrite

The Package Operation Mode (read/write mode) to be applied to the user.

0: Indicates Read/Write mode (default).

1: Indicates Read-Only mode.

Blank/omitted: Read-Only mode.

SynchronizeClient method

Synchronizes the client with the server.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Class

IGalaxy

Syntax

[C#]

```
void SynchronizeClient()
```

[Visual C++]

```
HRESULT SynchronizeClient();
```

Remarks

User should be logged in.

GetTimeMasterSettings method

Retrieves TimeMaster settings.

Class

IGalaxy

Syntax

[C#]

```
ISettings GetTimeMasterSettings( );
```

[Visual C++]

```
HRESULT GetTimeMasterSettings(
   [out, retval] ISettings**
);
```

Parameters

ISettings

Returns a gObjects collection object.

Remarks

Automatically check-out the instance for the settings object. While the instance is checked out, no other object may be checked out. User should be logged in.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

CreategObjectCollection method

Creates an empty gObjects collection object.

Class

IGalaxy

Syntax

[C#]

```
IgObjects CreategObjectCollection( );
```

[Visual C++]

```
HRESULT CreategObjectCollection(
   [out, retval] IgObjects** gObjects
);
```

Parameters

gObjects

The newly created gObjects object.

ImportObjects method

Imports objects into the Galaxy.

Class

IGalaxy

Syntax

[C#]

```
void ImportObjects(
   string inputFile,
   bool OverwritesAllowed
);
```

[Visual C++]

```
HRESULT ImportObjects(
   [in] BSTR inputFile,
   [in, optional, defaultvalue( 0 )]
   VARIANT_BOOL OverwritesAllowed
);
```

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Parameters

inputFile

The name of the input file -- can be aaPDF or aaPKG file. aaPDF is base Template file format and aaPKG is exported objects file format.

OverwritesAllowed

Indicates whether existing objects can be overwritten.

Remarks

User should be logged in.

ImportObjectsEx method

Class

IGalaxy

Syntax

[C#]

```
void ImportObjectsEx(
   string inputFile,
   E_RESOLVE_VERSION_CONFLICT_ACTION versionMismatchPreference,
   E_RESOLVE_NAME_CONFLICT_ACTION nameConflictPreference,
   string appendToObjectName
);
```

[Visual C++]

```
HRESULT ImportObjectsEx(
   // The name of the input file -- can be a PDF or //.csv file.
   [in] BSTR inputFile,
   [in] E_RESOLVE_VERSION_CONFLICT_ACTION versionMismatchPreference,
   [in] E_RESOLVE_NAME_CONFLICT_ACTION nameConflictPreference,
   [in] BSTR appendToObjectName
);
```

ImportObjectsEx2 method

Imports one or more of the objects contained in an aaPDF or aaPKG file into the Galaxy Repository.

Class

IGalaxy

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Syntax

[C#]

```
void ImportObjectsEx2
(
   string inputFile,
   ArchestrA.GRAccess.E_RESOLVE_VERSION_CONFLICT_ACTION versionMismatchPreference,
   ArchestrA.GRAccess.E_RESOLVE_NAME_CONFLICT_ACTION nameConflictPreference,
   string appendToObjectName, ArchestrA.GRAccess.E_RESOLVE_PROTECTION_CONFLICT_ACTION
   eProtectionConflict
);
```

[Visual C++]

```
HRESULT ImportObjectsEx2
   (
      [in] BSTR inputFile,
      [in] E_RESOLVE_VERSION_CONFLICT_ACTION versionMismatchPreference,
      [in] E_RESOLVE_NAME_CONFLICT_ACTION nameConflictPreference,
      [in] BSTR appendToObjectName,
      [in] E_RESOLVE_PROTECTION_CONFLICT_ACTION eProtectionConflict
   );
```

Parameters

inputFile

Name of the file to be imported. The file can be PDF or .csv format.

Returns:

S_OK: Import successful

E_UNEXPECTED: The IImportExport file content is not formatted as aaPDF or aaPKG content. See Notes, below.

E_FAIL: Unable to add objects to the Galaxy Repository

Notes

E_RESOLVE_PROTECTION_CONFLICT_ACTION eProtectionConflict is supported for handling protection conflicts.

The System Platform IDE gets a reference to IImportExport by calling the GetIImportExport() method on IGalaxyConfiguration.

ImportArchestrAApp method

Imports AVEVA OMI apps from a specified folder.

Class

IGalaxy

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Syntax

[C#]

```
void ImportArchestrAApp(string appPath);
```

[Visual C++]

```
HRESULT ImportArchestrAApp
   (
      [in] BSTR appPath
   );
```

Parameters

appPath

String that sets the path to the folder containing the AVEVA OMI app(s).

QueryToolsets method

Retrieves a collection object containing the available Toolsets.

Class

IGalaxy

Syntax

[C#]

```
IToolsets QueryToolsets( );
```

[Visual C++]

```
HRESULT QueryToolsets(
   [out, retval] IToolsets** toolsets
);
```

Parameters

toolsets

The Toolsets collection object.

Remarks

User should be logged in.

QueryObjectsByName method

Retrieves a gObjects collection based on a set of tagnames.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Class

IGalaxy

Syntax

[C#]

```
IgObjects QueryObjectsByName(
   EgObjectIsTemplateOrInstance templateOrInstance, ref string[] tagnames
);
```

[Visual C++]

```
HRESULT QueryObjectsByName(
   [in] EgObjectIsTemplateOrInstance templateOrInstance,
   [in, out] SAFEARRAY(BSTR)* Tagnames,
   [out, retval] IgObjects**
);
```

Parameters

templateOrInstance

Indicates whether a gObject is a template or an instance.

Tagnames

An array of tagnames. Any tagname found in the Galaxy will be included in the returned collection.

IgObjects

Returns a gObjects collection object.

Remarks

User should be logged in.

Note: If a tagname is not found, it is simply not included in the returned collection object.

GetUserDefaults method

Get the logged in user's defaults from the Galaxy repository.

Class

IGalaxy

Syntax

[C#]

```
string GetUserDefaults(
   EUserDefault userDefault
);
```

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

[Visual C++]

```
HRESULT GetUserDefaults(
   [in] EUserDefault userDefault,
   [out, retval] BSTR* defaultValue
);
```

Parameters

userDefault

The default.

defaultValue

The value of the default.

VersionNumber property

Retrieves the version number of the Galaxy.

Class

IGalaxy

Syntax

[C#]

```
int VersionNumber { get; };
```

[Visual C++]

```
HRESULT VersionNumber(
   [out, retval] LONG* VersionNumber
);
```

Parameters

VersionNumber

Returns the version number of Galaxy.

SetUserDefaults method

Set the logged in user's defaults.

Class

IGalaxy

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Syntax

[C#]

```
void SetUserDefaults(
   EUserDefault userDefault,
   string defaultValue
);
```

[Visual C++]

```
HRESULT SetUserDefaults(
   [in] EUserDefault userDefault,
   [in] BSTR defaultValue
);
```

Parameters

userDefault

The default.

defaultValue

The value of the default.

Restore method

Restores a Galaxy from a backup file.

Class

IGalaxy

Syntax

[C#]

```
void Restore(
   int lProcessId,
   string backupFilename,
   string GRNodeName,
   string Galaxy,
   bool bRestoreOlderVersion
);
```

[Visual C++]

```
HRESULT Restore(
   [in]LONG lProcessId,
   [in] BSTR backupFilename,
   [in, optional] BSTR GRNodeName,
   [in, optional] BSTR Galaxy,
   [in, optional] VARIANT_BOOL bRestoreOlderVersion
);
```

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Parameters

lProcessId

Proccess ID of the calling process.

backupFilename

Name of the backup file. File extension cab is appended if not specified. If a folder name is specified, the Galaxy is restored from the contents of the folder.

GRNodeName

The name of the GR node (computer name). Blank or omitted means the current computer is the GR.

Galaxy

The name of an exiting Galaxy (application database) into which the backed up Galaxy is to be restored into.

bRestoreOlderVersion

Indicates whether to restore the database even if backup is an older version. If omitted, this field is set to false indicating an older database that should not be restored.

Remarks

User should not be logged in.

Logout method

Logs out from an ArchestrA Galaxy.

Class

IGalaxy

Syntax

[C#]

```
void Logout( );
```

[Visual C++]

```
HRESULT Logout( );
```

Remarks

Logs out from an ArchestrA Galaxy.

UpgradeRequired property

Retrieves a Boolean that indicates whether the Galaxy needs to be upgraded.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Class

IGalaxy

Syntax

[C#]

```
bool UpgradeRequired { get; };
```

[Visual C++]

```
HRESULT UpgradeRequired(
   [out, retval] VARIANT_BOOL* vtUpgradeRequired
);
```

Parameters

vtUpgradeRequired

Returns a Boolean indicating whether the Galaxy needs to be upgraded.

MigrateGalaxy method

Upgrade a Galaxy.

Class

IGalaxy

Syntax

[C#]

```
void MigrateGalaxy(
   string galaxyName,
   string GRNodeName
);
```

[Visual C++]

```
HRESULT MigrateGalaxy(
   [in] BSTR galaxyName,
   [in, optional] BSTR grNodeName
);
```

Parameters

galaxyName

The name of the Galaxy.

grNodeName

The name of the GR node (computer name). Blank or omitted means the current computer is the GR.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

CdiVersionString property

Retrieves the version string of the Cdi.

Class

IGalaxy

Syntax

[C#]

```
string CdiVersionString { get; }
```

[Visual C++]

```
HRESULT CdiVersionString(
   [out, retval] BSTR* VersionString
);
```

Parameters

VersionString

Returns the version string of Cdi.

GetSecuritySettings method

Retrieves the security settings.

Class

IGalaxy

Syntax

[C#]

```
ISettings GetSecuritySettings( );
```

[Visual C++]

```
HRESULT GetSecuritySettings(
   [out, retval] ISettings** settings
);
```

Parameters

settings

Returns a gObjects collection.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Remarks

Automatically checks-out the instance of the settings object. While the instance is checked out, no other object may be checked out. User should be logged in.

CreateConditionsObject method

Creates a conditions object used with QueryObjectsMultiCondition.

Class

IGalaxy

Syntax

[C#]

```
IConditions CreateConditionsObject( );
```

[Visual C++]

```
HRESULT CreateConditionsObject(
   [out, retval] IConditions** conditions
);
```

Parameters

conditions

The newly created conditions object.

QueryObjects method

Retrieves gObjects collection based on the given criteria.

Class

IGalaxy

Syntax

[C#]

```
IgObjects QueryObjects(
   EgObjectIsTemplateOrInstance templateOrInstance,
   EConditionType Kind,
   object value,
   EMatch MatchCondition
);
```

[Visual C++]

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

```
HRESULT QueryObjects(
   [in] EgObjectIsTemplateOrInstance templateOrInstance,
   [in] EConditionType Kind,
   [in] VARIANT Value,
   [in, optional, defaultvalue( MatchCondition )] EMatch MatchCondition,
   [out, retval] IgObjects**
);
```

Parameters

templateOrInstance

Indicates whether a gObject is a template or an instance.

Kind

Indicates the type of condition used by the query.

Value

The value to be searched for. Meaning is based on the Kind parameter.

MatchCondition

To negate the condition, pass False.

IgObjects

Returns a gObjects collection.

Remarks

User should be logged in.

ImportScriptLib method

Import a script library.

Class

IGalaxy

Syntax

[C#]

```
void ImportScriptLib(
   string path
);
```

[Visual C++]

```
HRESULT ImportScriptLib(
   [in] BSTR path
);
```

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Parameters

path

Path to the .aaslib file.

ImportScriptLibEx method

Imports a Script Library (aaSLib file).

Class

IGalaxy

Syntax

[C#]

```
void ImportScriptLibEx(string path, ArchestrA.GRAccess.EImportDLLOptions options);
```

[Visual C++]

```
HRESULT ImportScriptLibEx
   (
      [in] BSTR path,
      [in] EImportDLLOptions options
   );
```

Parameters

path

Filepath to the Script Library file (.aaSLib).

EImportDLLOptions

Override options. The overrides will block the import of a DLL if it is deployed to a run-time node.

- **BlockIfDLLIsDeployed = 0:** Block if dependencies exist; allow unconditionally if there are none.
- **AllowImportIfDLLIsDeployed = 1:** Allow import unconditionally.
- **AllowImportIfDLLIsDeployedAndHasLowerVersionInGalaxy = 2:** Allow import even if dependencies exist, provided that the version of the DLL in the Galaxy is less than that which is being imported.
- **BlockImportIfImportVersionNotHigherThanVersionInGalaxy = 3:** Block import if dependencies exist, OR if Import DLL version is less than or equal to Galaxy DLL version.

GetLocaleSettings method

Retrieves locale settings.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Class

IGalaxy

Syntax

[C#]

```
ISettings GetLocaleSettings( );
```

[Visual C++]

```
HRESULT GetLocaleSettings(
   [out, retval] ISettings**
);
```

Parameters

ISettings

Returns a gObjects collection.

Remarks

Automatically check-out the instance for the settings object. While the instance is checked out, no other object may be checked out. User should be logged in.

ExportAll method

Exports all objects in the Galaxy.

Class

IGalaxy

Syntax

[C#]

```
void ExportAll(
   EExportType exportType,
   string outputFile
);
```

[Visual C++]

```
HRESULT ExportAll(
   [in] EExportType exportType,
   [in] BSTR outputFile
);
```

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Parameters

exportType

Indicates whether to export to aaPKG file.

outputFile

The name of the output file. Appends extension if one is not specified.

Remarks

User should be logged in.

GetReadOnlySecurity method

Accesses the Galaxy security in read-only mode. The user can not modify the Galaxy security.

Class

IGalaxy

Syntax

[C#]

```
IGalaxySecurity GetReadOnlySecurity( );
```

[Visual C++]

```
HRESULT GetReadOnlySecurity(
   [in] IGalaxySecurity** galaxySecurity
);
```

Parameters

galaxySecurity

IGalaxySecurity provides an API interface to read the Galaxy security information configured using the security dialog box in aaIDE.

CommandResults property

Returns the CommandResults object.

Class

IGalaxy

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Syntax

[C#]

```
ICommandResults CommandResults { get; };
```

[Visual C++]

```
HRESULT CommandResults(
   [out, retval] ICommandResults** CommandResults
);
```

GRLoad method

Class

IGalaxy

Syntax

[C#]

```
void GRLoad(
   string CSVFileNameWithPath,
   GRLoadMode GRLoadMode
);
```

[Visual C++]

```
HRESULT GRLoad(
   [in, string] LPCOLESTR CSVFileNameWithPath,
   [in] GRLoadMode grLoadMode
);
```

GetFlexLicensesInformation method

Returns an array that contains FLEX licenses available to the Galaxy Repository. The array includes the license type, the number of available licenses, number of deployed licenses, number of configured licenses, and number available for deployment (undeployed but available licenses).

One array is returned for each license type (single-engine platform licenses and multiple-engine platform licenses).

Class

IGalaxy

Syntax

[C#]

```
FLEXLICENSESINFORMATION[] GetFlexLicensesInformation();
```

[Visual C++]

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

```
HRESULT GetFlexLicensesInformation
(
[out, retval]SAFEARRAY(struct FLEXLICENSESINFORMATION)* response
);
struct FLEXLICENSESINFORMATION
{
int licenseType;
int availableToGR;
int deployed;
int configured;
int availableForDeploy;
}FLEXLICENSESINFORMATION;
```

Parameters

licenseType

Integer that defines the FLEX License Type:

- 1 = single engine platform
- 2 = multiple-engine platform

Each license type outputs an array that lists the number of licenses in each the following categories(number available to GR, number deployed, number configured, number available). There will be one array for single engine platform licenses, and one array for multiple engine licenses.

availableToGR

Total number of licenses (single or multiple) available for the Galaxy Repository to use.

deployed

Number of licenses (single or multiple) deployed from the Galaxy Repository.

configured

Number of licenses (single or multiple) that have been configured in the AVEVA™ License Manager.

availableForDeploy

Number of licenses (single or multiple) that are available to be deployed (total available minus number already deployed).

Example

```
// create GRAccessAppClass object
GRAccessApp grAccess = new GRAccessAppClass();
// Query for galaxies
IGalaxies gals = grAccess.QueryGalaxies(nodeName);
if (gals == null || grAccess.CommandResult.Successful == false)
{
Console.WriteLine(grAccess.CommandResult.CustomMessage + grAccess.CommandResult.Text);
return;
}
IGalaxy galaxy = gals[galaxyName];
FLEXLICENSESINFORMATION[] flex = galaxy.GetFlexLicensesInformation();
if (flex != null && flex.Length > 0)
{
```

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

```
Console.WriteLine(
"LicenceType:\t\t" + ((flex[0].licenceType == 1) ? "Single AppEngine":"Unlimited
AppEngine") + "\n" +
"AvailableToGR:\t\t" + flex[0].availableToGR.ToString() + " \n" +
"Deployed:\t\t" + flex[0].deployed.ToString() + "\n" +
"Configured:\t\t" + flex[0].configured.ToString() + "\n" +
"AvailableForDeploy:\t" + flex[0].availableForDeploy.ToString() + ");
}
for(int i= 0; i < flex.Length; i++)
{
Console.WriteLine(
"LicenceType:\t\t" + ((flex[i].licenceType == 1) ? "Single AppEngine" : "Unlimited
AppEngine") + "\n" +
"AvailableToGR:\t\t" + flex[i].availableToGR.ToString() + " \n" +
"Deployed:\t\t" + flex[i].deployed.ToString() + "\n" +
"Configured:\t\t" + flex[i].configured.ToString() + "\n" +
"AvailableForDeploy:\t" + flex[i].availableForDeploy.ToString());
Console.WriteLine("============================================================");
}
```

IGalaxyQueryToolsetsEx method

Class

IGalaxy

Syntax

[C#]

```
IToolsets QueryToolsetsEx(
   FolderType eFolderType
);
```

[Visual C++]

```
HRESULT QueryToolsetsEx
(
   // The list of toolsets.
   [in] enum FolderType eFolderType,
   [out, retval] IToolsets** toolsets
);
```

IGalaxies

Represents a collection of Galaxies.

IGalaxies class members

This class has the following members.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Operations

Item property

Return a Galaxy

Count property

Returns the number of Galaxy objects in the collection.

Item property

Return a toolset.

Class

IGalaxies

Syntax

[C#]

```
IGalaxy this[object GalaxyIdentifier] { get; };
```

[Visual C++]

```
HRESULT Item(
   [in] VARIANT GalaxyIdentifier,
   [out, retval] IGalaxy**
);
```

Parameters

GalaxyIdentifier

The index of a Galaxy. This is a numeric value from 1 to count, or the name of the Galaxy.

IGalaxy

The returned Galaxy object.

Count property

Returns the number of Galaxy objects in the collection.

Class

IGalaxies

Syntax

[C#]

```
int count { get; };
```

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

[Visual C++]

```
HRESULT Count(
   [out, retval] LONG*
);
```

IGalaxyRole

Accesses the role object and its properties, including role name, access level, list of general permissions, and operational permissions.

IGalaxyRole class members

This class has the following members.

Operations

RoleName property

Returns the name of the Galaxy role.

AccessLevel property

Returns the access level of the Galaxy role.

Permissions property

Returns the general permissions of the Galaxy role.

OperationalPermissions property

Returns the operational permissions of the Galaxy role.

RoleName property

Returns the name of the Galaxy role.

Class

IGalaxyRole

Syntax

[C#]

```
string RoleName { get; };
```

[Visual C++]

```
HRESULT RoleName(
   [out,retval]BSTR *roleName
);
```

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

AccessLevel property

Returns the access level of the Galaxy role.

Class

IGalaxyRole

Syntax

[C#]

```
int AccessLevel { get; };
```

[Visual C++]

```
HRESULT AccessLevel(
   [out,retval]LONG *accessLevel
);
```

Permissions property

Returns the general permissions of the Galaxy role.

Class

IGalaxyRole

Syntax

[C#]

```
IPermissions Permissions { get; };
```

[Visual C++]

```
HRESULT Permissions(
   [out,retval]IPermissions **generalPermissions
);
```

OperationalPermissions property

Returns the operational permissions of the Galaxy role.

Class

IGalaxyRole

Syntax

[C#]

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

```
IPermissions OperationalPermissions { get; };
```

[Visual C++]

```
HRESULT OperationalPermissions(
   [out,retval]IPermissions **operationalPermissions
);
```

IGalaxyRoles

Returns information about the roles in the Galaxy.

IGalaxyRoles class members

This class has the following members.

Operations

Item property

Returns a Galaxy role.

Count property

Returns the number of roles in the collection.

Item property

Return a Galaxy role.

Class

IGalaxyRoles

Syntax

[C#]

```
IGalaxyRole this[object galaxyIdentifier] { get; };
```

[Visual C++]

```
HRESULT Item(
   [in] VARIANT galaxyIdentifier,
   [out, retval] IGalaxyRole** GalaxyRole
);
```

Parameters

galaxyIdentifier

The index of a role. This is a numeric value from 1 to count, or the name of the role.

GalaxyRole

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

The returned Galaxy role object.

Count property

Returns the number of rolesin the collection.

Class

IGalaxyRoles

Syntax

[C#]

```
int count { get; };
```

[Visual C++]

```
HRESULT Count(
   [out, retval] LONG* count
);
```

Parameters

count

Returns the number of roles in the collection.

IGalaxySecurity

Accesses top-level security information and Galaxy security objects, including security groups, roles, and users configured for the Galaxy.

IGalaxySecurity class members

This class has the following members.

Operations

AuthenticationMode property

Retrieves the authentication mode of the Galaxy.

SecurityGroupsAvailable property

Retrieves the available security groups of the Galaxy.

RolesAvailable property

Retrieves the collection of roles configured for this Galaxy.

UsersAvailable property

Retrieves the collection of users configured for this Galaxy.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

LoginTime property

Retrieves the configured login time.

RoleUpdateInterval property

Retrieves role update interval time.

AuthenticationMode property

Retrieves the authentication mode of the Galaxy.

Class

IGalaxySecurity

Syntax

[C#]

```
EAUTHMODE AuthenticationMode { get; };
```

[Visual C++]

```
HRESULT AuthenticationMode(
   [out,retval]EAUTHMODE *authMode
);
```

Parameters

AuthenticationMode

Indicates whether Galaxy security mode is none, Galaxy, OS User, or OS Group.

SecurityGroupsAvailable property

Retrieves the available security groups of the Galaxy.

Class

IGalaxySecurity

Syntax

[C#]

```
ISecurityGroups SecurityGroupsAvailable { get; };
```

[Visual C++]

```
HRESULT SecurityGroupsAvailable(
   [out,retval]ISecurityGroups** securityGroups
);
```

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Parameters

SecurityGroupsAvailable

The collection of security groups configured for this Galaxy.

RolesAvailable property

Retrieves the collection of roles configured for this Galaxy.

Class

IGalaxySecurity

Syntax

[C#]

```
IGalaxyRoles RolesAvailable { get; };
```

[Visual C++]

```
HRESULT RolesAvailable(
   [out,retval]IGalaxyRoles** roles
);
```

Parameters

roles

The collection of roles configured for this Galaxy.

UsersAvailable property

Retrieves the collection of users configured for this Galaxy.

Class

IGalaxySecurity

Syntax

[C#]

```
IGalaxyUsers UsersAvailable { get; };
```

[Visual C++]

```
HRESULT UsersAvailable(
   [out,retval]IGalaxyUsers** users
);
```

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Parameters

users

The collection of users configured for this Galaxy.

LoginTime property

Retrieves the configured login time.

Class

IGalaxySecurity

Syntax

[C#]

```
string LoginTime { get; };
```

[Visual C++]

```
HRESULT LoginTime(
   [out,retval]BSTR* loginTime
);
```

Parameters

LoginTime

The timeout period during which the system validates the user's membership against the OS groups selected as ArchestrA roles. Minimum value is 0 (zero), maximum is 9,999,999. Default value is 0 (zero), which turns off this feature so the operation does not time out. Specify a value based on the speed of your network and the number of groups configured in ArchestrA roles. The slower the network or the larger the number of groups, the greater the value.

RoleUpdateInterval property

Retrieves role update interval time.

Class

IGalaxySecurity

Syntax

[C#]

```
string RoleUpdateInterval { get; };
```

[Visual C++]

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

```
HRESULT RoleUpdateInterval(
   [out,retval]BSTR* roleUpdateInterval
);
```

Parameters

RoleUpdateInterval

The time between each validation attempt per OS group for the user's membership when a logon is attempted. The user membership update is done one role per Role Update interval to minimize network usage. The minimum allowed value is 0 (zero) and the maximum is 9,999,999. The default value is 0 (zero), which turns off this feature so the operation does not pause between validating user membership and groups. This option operates independently of the Login Time option. Even if Login Time times out, the role update operation continues in the background and eventually updates user-to-role relationships for this user in the local cache.

IGalaxyUser

Provides user information: user name, full name, and associated role list.

If you are using OS group authentication mode, you can not configure Galaxy users. During login from client utilities, user information is updated in the Galaxy, which can then be obtained using the IGalaxyUser interface.

The FullName property is only valid when a user is an OS (domain or local) user.

IGalaxyUser class members

This class has the following members.

Operations

UserName property

Returns the user name along with the domain name in the form <*domain_name*>\<*user_name*>.

FullName property

Returns the full name of the user retrieved from the domain controller in the form <*first_name*> <*last_name*>.

AssociatedRoles property

Returns collection of roles associated with this user.

UserName property

Returns the user name along with the domain name in the form <*domain_name*>\<*user_name*>.

Class

IGalaxyUser

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Syntax

[C#]

```
string UserName { get; };
```

[Visual C++]

```
HRESULT UserName(
   [out,retval]BSTR* *userName
);
```

FullName property

Returns the full name of the user retrieved from the domain controller in the form *<first_name> <last_name>*.

Class

IGalaxyUser

Syntax

[C#]

```
string FullName { get; };
```

[Visual C++]

```
HRESULT FullName(
   [out,retval]BSTR* *fullName
);
```

AssociatedRoles property

Returns collection of roles associated with this user.

Class

IGalaxyUser

Syntax

[C#]

```
IGalaxyRoles AssociatedRoles { get; };
```

[Visual C++]

```
HRESULT AssociatedRoles(
   [out,retval]IGalaxyRoles **associatedRoles
);
```

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

IGalaxyUsers

Represents a collection of Galaxy users.

IGalaxyUsers class members

This class has the following members.

Operations

Item property

Returns a Galaxy user.

Count property

Returns the number of roles in the collection.

Item property

Return a Galaxy user.

Class

IGalaxyUsers

Syntax

[C#]

```
IGalaxyUser this[object galaxyIdentifier] { get; };
```

[Visual C++]

```
HRESULT Item(
   [in] VARIANT galaxyIdentifier,
   [out, retval] IGalaxyUser** galaxyUser
);
```

Parameters

galaxyIdentifier

The index of a user. This is a numeric value from 1 to count, or the name of the user.

galaxyUser

The returned Galaxy user.

Count property

Returns the number of users in the collection.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Class

IGalaxyUsers

Syntax

[C#]

```
int count { get; };
```

[Visual C++]

```
HRESULT Count(
   [out, retval] LONG* UserCount
);
```

Parameters

UserCount

Returns the number of users in the collection.

IGraphicAccess

The set of GraphicAccess application programming interfaces (API) allow users to export graphics to an XML file programmatically. Use the same set of APIs to create an Industrial Graphic in another galaxy or to overwrite an existing graphic(s) by importing the graphic definition from an XML file. The APIs can also be used to export the configured references or GPI for a graphic and importing multiple XML files from a folder location.

Before using the set of GraphicAccess APIs, you must use the GRAccess Toolkit to log in and connect to an Galaxy repository.

The APIs are available through the following interfaces:

- IGraphicAccess – Export and import an Industrial Graphic
- IGraphicAccess2 – Export the substitutable strings and configured references for an Industrial Graphic
- IGraphicAccess3 – Export the GPI of an Industrial Graphic
- IGraphicAccess4 – Import Industrial Graphics from a folder location

About programmatic graphic export and import

The GraphicAccess programmatic API exports or imports an extensive set of properties of an Industrial graphic. For standard Industrial Graphics, an exported or imported graphic can contain the following:

- Custom properties
- Configured References
- Graphic elements
- Connector lines
- Graphic groups
- Graphic animations

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

- Element styles
- Named scripts
- Predefined scripts
- Overridden text strings
- Numeric format styles
- DataStatus elements
- Trend Pen
- Alarm Client
- Trend Client

In addition to the properties of a standard Industrial graphic, an exported or imported Symbol Wizard can contain the following:

- Wizard Options
- Choice groups
- Choices
- Layers
- Rules

The IGraphicAccess API

The GraphicAccess API is implemented by the ArchestrA.Visualization.GraphicAccess.dll file, located in: \Program Files (x86)\ArchestrA\Framework\Bin

IGraphicAccess is used to export an Industrial Graphic to an XML file, or to import an XML file into a Galaxy as an Industrial Graphic. IGalaxy points to the Galaxy where the import/export operation is performed.

```
public interface IGraphicAccess
{
/// <summary>Export an Industrial Graphic to XML file</summary>
/// <param name="galaxy">IGalaxy obtained from GRAccess</param>
/// <param name="graphicName">The graphic name</param>
/// <param name="xmlFilePath">The XML file path</param>
/// <returns>Result of the method</returns>
IGraphicAccessResult ExportGraphicToXml(IGalaxy galaxy, string graphicName, string
xmlFilePath);
/// <summary>Import the XML file and generate an Industrial Graphic</summary>
/// <param name="galaxy">IGalaxy obtained from GRAccess</param>
/// <param name="graphicName">The graphic name</param>
/// <param name="xmlFilePath">The XML file path</param>
/// <param name="bOverWrite">Flag which indicates if existing graphic should be
overwritten</param>
/// <returns>Result of the method</returns>
IGraphicAccessResult ImportGraphicFromXml(IGalaxy galaxy, string graphicName, string
xmlFilePath, bool bOverWrite);
void ValidateSymboltoExport(IGalaxy galaxy, string graphicName);
}
public interface IGraphicAccessResult : ICommandResult
{
```

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

```
XMLOperations.StatusCode Status { get; }
}
```

If the Galaxy has security enforced, you must first log in to the Galaxy Repository with proper credentials. Also, the **Can Export Graphics** and **Can Import Graphics** Galaxy role permissions must be set active before attempting an import or export operation.

You can set the **Can Export Graphics** and **Can Import Graphics** Galaxy role permissions from the IDE **Configure Security** dialog box. For more information about setting Galaxy permissions, see the Application Server Help.

GraphicAccess interface methods

The IGraphicAccess interface includes separate graphic export and import methods.

ExportGraphicToXml method

Exports an Industrial Graphic to an XML file. Galaxy, graphic name, and the XML file path are passed as parameters to the ExportGraphicToXml function.

Syntax

```
IGraphicAccessResult ExportGraphicToXml(IGalaxy galaxy, string graphicName, string
xmlFilePath);
```

Parameters

galaxy

IGalaxy obtained from GRAccess, returns the galaxy containing the graphic to export.

graphicName

Name of the graphic to export.

xmlFilePath

Directory folder to place the XML file containing the exported graphic.

ImportGraphicFromXml method

Imports a graphic from an existing XML file. Galaxy, graphic name, the XML file path, and an overwrite flag are passed as parameters to the ImportGraphicfromXml function.

Syntax

```
IGraphicAccessResult ImportGraphicFromXml(IGalaxy galaxy, string graphicName, string
xmlFilePath, bool bOverWrite);
```

Parameters

galaxy

IGalaxy obtained from GRAccess, returns the galaxy where the graphic will be imported.

graphicName

Name of the graphic to import.

xmlFilePath

Directory folder location of the XML file.

bOverWrite

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Boolean flag that indicates if an existing graphic can be overwritten by an imported graphic with the same name.

ValidateSymboltoExport method

Validates the graphic before exporting it to an XML file. For more information, see *Validate Objects in Application Server Help*.

Syntax

```
void ValidateSymboltoExport(IGalaxy galaxy, string graphicName);
```

Parameters

galaxy

IGalaxy obtained from GRAccess, returns the galaxy that contains the graphic that requires validation.

graphicName

Name of the graphic to validate.

After an import or export operation is complete, the results are set to IGraphicAccessResult which is derived from ICommandResult. A message appears and indicates if the operation succeeded or failed. For detailed information about the ICommandResult interface, see the *GRAccess Toolkit API User's Guide*.

A succeeded message only means the import or export operation finished successfully. It does not indicate the quality of the exported XML file or the imported graphic. Check the SMC log file for any warning or error messages after each export or import operation.

About programmatic string and reference export and import

The GraphicAccess2 API exports all of a graphic's substitutable strings to a specified XML file, regardless of whether some of the strings have been overridden. All graphic strings that appear from a **Ctrl+L** or **Substitute Strings** operation from within the Graphic Editor will be exported, allowing for bulk editing of the graphic strings.

The GraphicAccess2 API can also be used to export a list of configured references for a selected object. The list is exported to a XML string. The API will export the following configured references:

- Mx references
- InTouch tags, including those configured from the argument of attribute

  Example: attribute("InTouch:zhmi-abc")
- Custom properties configured with animation links
- Client scripts
- Custom property overrides
- Client controls
- All applicable references configured within an embedded graphic, regardless of the number of times it is embedded

The IGraphicAccess2 API

The GraphicAccess2 API is implemented by the ArchestrA.Visualization.GraphicAccess.dll file, located in: \Program Files (x86)\ArchestrA\Framework\Bin

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

IGraphicAccess2 is used to export all of a graphic's substitutable strings to a specified XML file, regardless of whether some of the strings have been overridden.

The GraphicAccess2 API can also be used to export a list of configured references for a selected graphic to an XML string.

IGraphicAccess2 passes galaxy, graphicName, the XML file path, and substitute strings as parameters.

```
public interface IGraphicAccess2 : IGraphicAccess
{
/// <summary>Export an Industrial Graphic to XML file</summary>
/// <param name="galaxy">IGalaxy obtained from GRAccess</param>
/// <param name="graphicName">The graphic name</param>
/// <param name="xmlFilePath">The XML file path</param>
/// <param name="bExportSubstituteStrings">Option to export SubstituteStrings</param>
/// <returns>Result of the method</returns>
IGraphicAccessResult ExportGraphicToXml(IGalaxy galaxy, string graphicName, string
xmlFilePath, bool bExportSubstituteStrings);
/// <summary>Get references configured on an Industrial Graphic</summary>
/// <param name="galaxy">IGalaxy obtained from GRAccess</param>
/// <param name="graphicName">The graphic name</param>
/// <param name="sConfiguredReferencesXML">Returns all the references in XML format</param>
/// <returns>Result of the method</returns>
IGraphicAccessResult GetConfiguredReferences(IGalaxy galaxy, string graphicName, out string
sConfiguredReferencesXML);
}
```

If the Galaxy has security enforced, you must first log in to the Galaxy Repository with proper credentials. Also, the **Can Export Graphics** and **Can Import Graphics** Galaxy role permissions must be set active before attempting an import or export operation.

You can set the **Can Export Graphics** and **Can Import Graphics** Galaxy role permissions from the IDE **Configure Security** dialog box. For more information about setting Galaxy permissions, see the Application Server Help.

GraphicAccess2 interface methods

The GraphicAccess2 interface uses separate methods to export substitutable strings and configured references for an Industrial Graphic.

ExportGraphicToXml method

Exports all substitutable strings in an Industrial Graphic to an XML file. Galaxy, graphic name, the XML file path and Export Substitute Strings option are the parameters for the ExportGraphictoXML function. This API will not export empty strings and strings used in invisible graphics.

Syntax

```
IGraphicAccessResult ExportGraphicToXml(IGalaxy galaxy, string graphicName, string
xmlFilePath, bool bExportSubstituteStrings);
```

Parameters

galaxy

IGalaxy obtained from the GRAccess and returns the galaxy.

graphicName

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Name of the graphic to export.

xmlFilePath

Directory folder to place the XML file containing the exported graphic.

bExportSubstituteStrings

When set 'True' will include the substitute string from graphic in the export XML file.

GetConfiguredReferences method

Exports all of the configured references within an Industrial Graphic to an XML file.

Syntax

```
IGraphicAccessResult GetConfiguredReferences(IGalaxy galaxy, string graphicName, out string sConfiguredReferencesXML);
```

Parameters

galaxy

IGalaxy obtained from the GRAccess and returns galaxy containing the graphic object.

graphicName

Name of the graphic with references to export.

sConfiguredReferencesXML

The configured references for the graphic will be returned to this parameter in a string.

Programmatic GPI export

The IGraphicAccess3 API exports the Graphics Performance Index (GPI) rating for a graphic and related details to a specified XML file. The details are as follows:

Item	Relevance	Example
Description	**Component type**	**Number of Trend Pens**
Count	Number of items comprising the component type	3
Impact Score	Call up time in milliseconds for the component type	2.9

The IGraphicAccess3 API

The IGraphicAccess3 API is implemented by the ArchestrA.Visualization.GraphicAccess.dll, located in: \Program Files (x86)\ArchestrA\Framework\Bin

The IGraphicAccess3 API is used to export the Graphics Performance Index (GPI) of the Industrial Graphic.

IGraphicAccess3 passes galaxy, graphicName and the XML file path as parameters.

```
public interface IGraphicAccess3 : IGraphicAccess2
{
```

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

```
/// <summary>Get GPI XML of an Industrial Graphic</summary>
/// <param name="galaxy">IGalaxy obtained from GRAccess</param>
///<param name="graphicName">The graphic name</param>
///<param name="sGPIXML">Returns GPI details in XML format</param>
/// <returns>Result of the method</returns
ICommandResult GetGPI (IGalaxy galaxy, string graphicName, out string sGPIXML);
}
```

GraphicAccess3 interface methods

GetGPI Method

Exports the GPI of a graphic to an XML file.

Syntax

```
ICommandResult GetGPI (IGalaxy galaxy, string graphicName, out string sGPIXML);
```

Parameters

galaxy

IGalaxy obtained from the GRAccess and returns the galaxy.

graphicName

Name of the graphic for which the GPI is needed.

sGPIXML

The GPI information (XML) for the graphic will be returned to this parameter in a string.

```
<?xml version="1.0" encoding="utf-8"?>
<Metrices>
  <GPI>5.0</GPI>
  <Description>Sub-second call-up time on standard performance hardware</Description>
  <Details>
      <Item Description="Number of external References" count="4", score = "3.9"/>
      <Item Description="Number of Paths" count="3", score = "1.1"/>
      <Item Description="Number of Rectangles" count="10", score = "<1"/>
      <Item Description="Number of Value Display Animations" count="1", score = "<1"/>
      <Item Description="Number of Action Scripts" count="1", score = "<1"/>
  </Details>
</Metrices>
```

About programmatic import from a folder location

The GraphicAccess4 programmatic API and can import Industrial Graphics from multiple XML files available in a folder.

The IGraphicAccess4 API

The GraphicAccess4 API is implemented by the ArchestrA.Visualization.GraphicAccess.dll file, located in: \Program Files (x86)\ArchestrA\Framework\Bin

IGraphicAccess4 is used to import graphic XML files. IGalaxy points to a Galaxy where the import operation is

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

performed.

```
public interface IGraphicAccess4 : IGraphicAccess3
{
/// <summary>Import the XML files from the folder and generate Industrial
Graphics</summary>
/// <param name="galaxy">IGalaxy obtained from GRAccess</param>
/// <param name="strFolderPath">The folder/directory path where graphic XML files are
present</param>
/// <param name="bOverWrite">Flag which indicates if existing graphic should be
overwritten</param>
/// <param name="opNotifier">Returns the status of the import operation to the calling
function (client)</param>
/// <param name="cancelToken">Sets the token which indicates if the operation needs to be
canceled during import</param>
/// <returns>Result of the method</returns>
Task<IGraphicAccessResult> ImportGraphics(IGalaxy galaxy, string strFolderPath, bool
bOverWrite, IGraphicOperationStatus opNotifier = null, CancellationToken cancelToken =
default);
}
```

The IGraphicAccess4 uses the IGraphicOperationStatus callback interface to determine if the operation is in progress, cancelled or completed.

```
namespace ArchestrA.Visualization.GraphicAccess
{
public interface IGraphicOperationStatus
{
void OnOperationProgressMessage(string message);
void OnOperationComplete();
void OnOperationCancel();
}
}
```

GraphicAccess4 interface methods

ImportGraphics method

Imports graphics from existing XML files available in a folder. Galaxy, folder path, an overwrite flag, progress of the operation and a token to determine if the operation is cancelled are passed as parameters to the ImportGraphics function.

Syntax

```
Task<IGraphicAccessResult> ImportGraphics(IGalaxy galaxy, string strFolderPath, bool
bOverWrite, IGraphicOperationStatus opNotifier = null, CancellationToken cancelToken =
default);
```

Parameters

galaxy

Points to a galaxy to which graphics will be imported or where the API will be run.

strFolderPath

Directory folder location of the XML files.

bOverWrite

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Boolean flag that indicates if an existing graphic can be overwritten by an imported graphic with the same name.

opNotifier

Returns the status of the import operation to the calling function (client)

cancelToken

Calling function (Client) can set this token to cancel the import graphics operation. For more information on CancellationToken, see the MSDN help.

After the import operation is complete, the results are set to Task<IGraphicAccessResult> which is derived from ICommandResult. A message appears and indicates if the operation succeeded or failed. For detailed information about the ICommandResult interface, see the G*RAccess Toolkit API Help*.

A succeeded message only means the import operation finished successfully. It does not indicate the quality of the exported XML file or the imported graphic. Check the SMC log file for any warning or error messages after each import operation.

IgObject

Represents a gObject that could be either a template or an instance.

IgObject class members

This class has the following members.

Operations

Tagname property

Returns or sets the tagname of this object.

SortOrder property

Returns or sets the sort order of this object.

ContainedName property

Returns or sets the contained name of this object (for example, Inlet).

HierarchicalName Property

Returns the hierarchical name of this object (for example, Reactor.Tank.Inlet).

Alias property

Returns or sets the alias of this object.

Attributes property

The collection of attributes of this object.

ConfigurableAttributes property

The collection of attributes of this object that can be configured.

ValidationStatus property

The validation status of this object based on the last time the object was validated and saved.

Errors property

The list of errors that was generated the last time the object was validated and saved.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Warnings property

The list of warnings that was generated the last time the object was validated and saved.

CheckedOutBy property

The name of the user that this object is checked out by.

CheckoutStatus property

The check-out status of the object.

ConfigVersion Property

The configuration version of the object.

Category property

The category of the object.

DerivedFrom property

Returns the name of the template that this object was created from.

BasedOn property

The name of the base template (root ancestor) of this object.

Container property

Returns or sets the name of the container object, or blank if object is not contained.

Area property

Returns or sets name of the area object, or blank if the object does not have an area.

Host property

Returns or sets the name of the host for this object, or blank if unassigned.

GetObjectHelpURL method

Returns the URL where the object's help is stored in the Galaxy Repository.

Save method

Saves the object after it is configured.

CheckIn method

Checks in an object.

CheckOut method

Checks out an object.

UndoCheckOut method

Undoes check-out of this object.

Unload method

Unloads the gObject cache.

EditStatus property

The edit status of the object.

AddExtensionPrimitive method

Add an extension primitive.

DeleteExtensionPrimitive method

Delete an extension primitive.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

RenameExtensionPrimitive method

Rename an object extension primitive.

AddUDA method

Add a UDA.

DeleteUDA method

Delete a UDA by its name.

RenameUDA method

Renames a UDA.

UpdateUDA method

Update a UDA.

CommandResult property

Returns the CommandResult object, which has the last method call's result.

CategoryGUID property

Returns the Category GUID of the object.

GetExtendedAttributes method

Returns the Extended Attributes in the hierarchy.

PlatformLicenseType property

Returns or sets the FLEX licensing state.

Tagname property

Returns or sets the tagname of this object.

Class

IgObject

Syntax

[C#]

```
string Tagname { set; get; };
```

[Visual C++]

```
[propget]
HRESULT Tagname(
   [out, retval] BSTR* theTagname
);
[propput]
HRESULT Tagname(
   [in] BSTR newTagName
);
```

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Parameters

theTagname

The current tagname of this object.

newTagName

The new tagname of this object. The object does not need to be checked out. The operation is immediate.

ContainedName property

Sets or returns the contained name of this object (for example, Inlet).

Class

IgObject

Syntax

[C#]

```
string ContainedName { set; get; };
```

[Visual C++]

```
[propget]
HRESULT ContainedName(
   [out, retval] BSTR* theContainedName
);
[propput]
HRESULT ContainedName(
   [in] BSTR newContainedName
);
```

Parameters

theContainedName

The current contained name of this object.

newContainedName

The new contained name of this object.

HierarchicalName Property

Returns the hierarchical name of this object (for example, Reactor.Tank.Inlet).

Class

IgObject

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Syntax

[C#]

```
string HierarchicalName { get; };
```

[Visual C++]

```
HRESULT HierarchicalName(
   [out, retval] BSTR* theHierarchicalName
);
```

Parameters

theHierarchicalName

The current hierarchical name of this object.

Attributes property

Returns the collection of attributes of this object.

Class

IgObject

Syntax

[C#]

```
IAttributes Attributes { get; };
```

[Visual C++]

```
HRESULT Attributes(
   [out, retval] IAttributes** theAttributes
);
```

Parameters

theAttributes

The collection of attributes of this object.

ConfigurableAttributes property

The collection of this object's attributes that can be configured. Configuring attributes requires that the object be checked out.

Class

IgObject

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Syntax

[C#]

```
IAttributes ConfigurableAttributes { get; };
```

[Visual C++]

```
HRESULT ConfigurableAttributes(
   [out, retval] IAttributes** theConfigurableAttributes
);
```

Parameters

theConfigurableAttributes

The collection of attributes that can be configured.

ValidationStatus property

The validation status of this object based on the last time the object was validated and saved.

Class

IgObject

Syntax

[C#]

```
EPACKAGESTATUS ValidationStatus { get; };
```

[Visual C++]

```
HRESULT ValidationStatus(
   // The validation status of the object.
   [out, retval] enum EPACKAGESTATUS* theValidationStatus
);
```

Parameters

theValidationStatus

The validation status of the object.

Errors property

Retrieves the list of errors generated when the object was last validated and saved.

Class

IgObject

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Syntax

[C#]

```
string[] Errors { get; };
```

[Visual C++]

```
HRESULT Errors(
   [out, retval] SAFEARRAY(BSTR)* theErrors
);
```

Parameters

theErrors

The list of errors.

Warnings property

Returns the list of warnings generated the last time the object was validated and saved.

Class

IgObject

Syntax

[C#]

```
string[] Warnings { get; };
```

[Visual C++]

```
HRESULT Warnings(
   [out, retval] SAFEARRAY(BSTR)* theWarnings
);
```

Parameters

theWarnings

The list of warnings.

CheckedOutBy property

Returns the name of the user who has checked out this object.

Class

IgObject

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Syntax

[C#]

```
string checkedOutBy { get; };
```

[Visual C++]

```
HRESULT CheckedOutBy(
   [out, retval] BSTR* CheckedOutByUserName
);
```

Parameters

CheckedOutByUserName

The user's name.

CheckoutStatus property

Returns the check out status of the object.

Class

IgObject

Syntax

[C#]

```
ECheckoutStatus CheckoutStatus { get; };
```

[Visual C++]

```
HRESULT CheckoutStatus(
   [out, retval] enum ECheckoutStatus* theCheckoutStatus
);
```

Parameters

theCheckoutStatus

The check-out status.

ConfigVersion Property

Returns the configuration version number of the object . The configuration version is incremented each time the object is checked in.

Class

IgObject

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Syntax

[C#]

```
int ConfigVersion { get; };
```

[Visual C++]

```
HRESULT ConfigVersion(
   [out, retval] LONG* theConfigVersion
);
```

Parameters

theConfigVersion

The configuration version.

Category property

Returns the category of the object.

Class

IgObject

Syntax

[C#]

```
ECATEGORY category { get; };
```

[Visual C++]

```
HRESULT Category(
   [out, retval] ECATEGORY* theCategory
);
```

Parameters

theCategory

The category.

DerivedFrom property

Returns the name of the template that this object was created from.

Class

IgObject

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Syntax

[C#]

```
string DerivedFrom { get; };
```

[Visual C++]

```
HRESULT DerivedFrom(
   [out, retval] BSTR* theParentTemplate
);
```

Parameters

theParentTemplate

The name of the object's parent template.

BasedOn property

Returns the name of the base template (root ancestor) of this object.

Class

IgObject

Syntax

[C#]

```
string basedOn { get; };
```

[Visual C++]

```
HRESULT BasedOn(
   [out, retval] BSTR* theBaseTemplate
);
```

Parameters

theBaseTemplate

The name of the object's base template.

Container property

Returns or sets the name of the container object, or blank if object is not contained.

Class

IgObject

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Syntax

[C#]

```
string Container { set; get; };
```

[Visual C++]

```
[propget]
HRESULT Container(
   [out, retval] BSTR* theContainer
[propput]
HRESULT Container(
   [in] BSTR NewContainer
);
```

Parameters

theContainer

The name of the current container object, or blank if object is not contained.

NewContainer

The new name of the container, or blank if uncontained. If this object is an ApplicationObject, its area and host are also updated. The object does not need to be checked out. The operation is immediate.

Area property

Returns or sets the name of the Area object, or blank if the object does not have an area.

Class

IgObject

Syntax

[C#]

```
string Area { set; get; };
```

[Visual C++]

```
[propget]
HRESULT Area(
   [out, retval] BSTR* theArea
);
[propput]
HRESULT Area(
   [in] BSTR NewArea
);
```

Parameters

theArea

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

The name of the current Area object, or blank if the object doesn't have an area.

NewArea

The new name of the area, or blank to unassign the object from its existing area.

Host property

Returns or sets the name of the host for this object, or blank if unassigned.

Class

IgObject

Syntax

[C#]

```
string Host { set; get; };
```

[Visual C++]

```
[propget]
HRESULT Host(
   [out, retval] BSTR* theHost
);
[propput]
HRESULT Host(
   [in] BSTR NewHost
);
```

Parameters

theHost

The current name of the host for this object.

NewHost

The new name of the host, or blank to unassign the object from its existing host.

GetObjectHelpURL method

Returns the URL where the object's help is stored on the Galaxy repository. The client can display the URL in Internet Explorer or use web publishing to change the data.

Class

IgObject

Syntax

[C#]

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

```
string GetObjectHelpURL();
```

[Visual C++]

```
HRESULT GetObjectHelpURL(
   [out, retval] BSTR*
);
```

Save method

Save the object after it is configured. Object must be checked out.

Class

IgObject

Syntax

[C#]

```
void Save();
```

[Visual C++]

```
HRESULT Save();
```

CheckIn method

Checks in an object. Invoked for a checked-out object after it is configured and saved.

Class

IgObject

Syntax

[C#]

```
void CheckIn(
   string CheckInComment
);
```

[Visual C++]

```
HRESULT CheckIn(
   // Check in an object.
   [in, optional] BSTR CheckInComment
);
```

Parameters

CheckInComment

Check in an object.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

CheckOut method

Checks out an object.

Class

IgObject

Syntax

[C#]

```
void CheckOut();
```

[Visual C++]

```
HRESULT CheckOut();
```

UndoCheckOut method

Reverses the check out operation without making changes.

Class

IgObject

Syntax

[C#]

```
void UndoCheckOut();
```

[Visual C++]

```
HRESULT UndoCheckOut();
```

Unload method

Unloads the gObject cache. This method does not affect functionality. It is used to release internal resources obtained by the object during operations. Unload() can be called to free those resources. After this method is called, the object is still usable and will reload the internal resources as needed.

Class

IgObject

Syntax

[C#]

```
void Unload();
```

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

[Visual C++]

```
HRESULT Unload();
```

EditStatus property

Edit the status of the object.

Class

IgObject

Syntax

[C#]

```
EEditStatus EditStatus { get; };
```

[Visual C++]

```
HRESULT EditStatus(
   [out, retval] enum EEditStatus* theEditStatus
);
```

Parameters

theEditStatus

The edit status of the object.

AddExtensionPrimitive method

Adds an extension primitive to the object.

Class

IgObject

Syntax

[C#]

```
void AddExtensionPrimitive(
   string ExtensionType,
   string ExtensionPrimitiveName,
   bool IsObjectExtension
);
```

[Visual C++]

```
HRESULT AddExtensionPrimitive(
   [in] BSTR ExtensionType,
   [in] BSTR ExtensionPrimitiveName,
   [in, optional, defaultvalue( -1 )]
```

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

```
   VARIANT_BOOL IsObjectExtension
);
```

Parameters

ExtensionType

Type of extension.

ExtensionPrimitiveName

Name of extension primitive to be added. For an attribute extension primitive, this parameter is the fully qualified name of the attribute being extended. For an object extension primitive, it is the new name specified by user for object extension primitive.

IsObjectExtension

If ExtensionPrimitiveName is the new name specified by the user, this field is set to true (for example, when add script pass is true).

DeleteExtensionPrimitive method

Deletes an extension primitive.

Class

IgObject

Syntax

[C#]

```
void DeleteExtensionPrimitive(
   string ExtensionType,
   string ExtensionPrimitiveName
);
```

[Visual C++]

```
HRESULT DeleteExtensionPrimitive(
   [in] BSTR ExtensionType,
   [in] BSTR ExtensionPrimitiveName
);
```

Parameters

ExtensionType

Type of extension.

ExtensionPrimitiveName

Name of extension primitive to be deleted.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

RenameExtensionPrimitive method

Renames an object extension primitive.

Class

IgObject

Syntax

[C#]

```
void RenameExtensionPrimitive(
   string OldPrimitiveName,
   string NewPrimitiveName
);
```

[Visual C++]

```
HRESULT RenameExtensionPrimitive(
   [in] BSTR OldPrimitiveName,
   [in] BSTR NewPrimitiveName
);
```

Parameters

OldPrimitiveName

Old name of primitive extension.

NewPrimitiveName

New name specified by user.

AddUDA method

Adds a UDA.

Class

IgObject

Syntax

[C#]

```
void AddUDA(
   string UDAName,
   MxDataType DataType,
   MxAttributeCategory category,
   MxSecurityClassification Security,
   bool IsArray,
   object ArrayElementCount
);
```

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

[Visual C++]

```
HRESULT AddUDA
(
   [in] BSTR UDAName,
   [in, optional, defaultvalue( MxBoolean )] MxDataType Datatype,
   [in, optional, defaultvalue( MxCategoryWriteable_UC_Lockable )] MxAttributeCategory
   Category,
   [in, optional, defaultvalue( MxSecurityOperate )] MxSecurityClassification Security,
   [in, optional, defaultvalue( 0 )] VARIANT_BOOL IsArray,
   [in, optional] VARIANT ArrayElementCount
);
```

Parameters

UDAName

The new UDA's name.

Datatype

The new UDA's datatype.

Category

The new UDA's category.

Security

The new UDA's security classification.

IsArray

The new UDA is an array.

ArrayElementCount

The new UDA's array element count.

DeleteUDA method

Deletes a UDA by its name.

Class

IgObject

Syntax

[C#]

```
void DeleteUDA(
   string UDAName
);
```

[Visual C++]

```
HRESULT DeleteUDA(
   [in] BSTR UDAName
);
```

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Parameters

UDAName

Delete a UDA by its name.

RenameUDA method

Changes the name of a UDA.

Class

IgObject

Syntax

[C#]

```
void RenameUDA(
   string OldUDAName,
   string NewUDAName
);
```

[Visual C++]

```
HRESULT RenameUDA(
   [in] BSTR OldUDAName,
   [in] BSTR NewUDAName
);
```

Parameters

OldUDAName

The old external name of the primitive to be renamed.

NewUDAName

The new external name of the primitive.

UpdateUDA method

Updates a UDA with new data that overwrites the old data.

Class

IgObject

Syntax

[C#]

```
void UpdateUDA(
```

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

```
   string wszUDAInfo,
   MxDataType DataType,
   MxAttributeCategory category,
   MxSecurityClassification Security,
   bool IsArray,
   object ArrayElementCount
);
```

[Visual C++]

```
HRESULT UpdateUDA
(
   [in] BSTR wszUDAInfo,
   [in, optional, defaultvalue( MxBoolean )] MxDataType Datatype,
   [in, optional, defaultvalue( MxCategoryWriteable_UC_Lockable )] MxAttributeCategory
   Category,
   [in, optional, defaultvalue( MxSecurityOperate )] MxSecurityClassification Security,
   [in, optional, defaultvalue( 0 )] VARIANT_BOOL IsArray,
   [in, optional] VARIANT ArrayElementCount);
);
```

Parameters

wszUDAInfo

The updated UDA data.

Datatype

The updated UDA's datatype.

Category

The updated UDA's category.

Security

The updated UDA's security classification.

IsArray

The updated UDA is an array.

ArrayElementCount

The updated UDA's array element count.

CommandResult property

Returns the CommandResult object, which has the last method call's result.

Class

IgObject

Syntax

[C#]

```
ICommandResult CommandResult { get; };
```

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

[Visual C++]

```
HRESULT CommandResult(
   [out, retval] ICommandResult** CommandResult
);
```

Parameters

CommandResult

Returns the CommandResult object.

CategoryGUID property

Returns the category GUID of the object.

Class

IgObject

Syntax

[C#]

```
VBGUID CategoryGUID { get; };
```

[Visual C++]

```
HRESULT CategoryGUID
(
   // The Category of the object.
   [out, retval] VBGUID *categoryGUID
);
```

GetExtendedAttributes method

Returns the extended attributes in the hierarchy.

Class

IgObject

Syntax

[C#]

```
IAttributes GetExtendedAttributes(
   string AttributeName,
   int upto_level,
   MxAttributeCategory[] ReturnOnlyTheseCategories
);
```

[Visual C++]

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

```
HRESULT GetExtendedAttributes(
   [in] BSTR AttributeName,
   [in] int upto_level,
   [in]SAFEARRAY(MxAttributeCategory) ReturnOnlyTheseCategories,
   [out,retval]IAttributes** theAttributes
);
```

Parameters

AttributeName

Attribute full name which is extended

upto_level

Up to how many level user needs extended primitives -1 will return all levels

ReturnOnlyTheseCategories

Return attributes which are only in these categories. If safe array is empty, method will return all attributes.

theAttributes

Returns the attributes

PlatformLicenseType property

Sets or returns the license type of the Platform object.

Class

IgObject

Syntax

[C#]

```
int PlatformLicenseType { get; set; }
```

[Visual C++]

To get the FLEX License type of the Platform (gObject):

```
[propget]
HRESULT PlatformLicenseType
   (
      [out, retval] int* platformLicenseType
   );
```

To set the FLEX License type of the Platform (gObject):

```
[propput]
HRESULT PlatformLicenseType
   (
      [in] int platformLicenseType
   );
```

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Parameters

PlatformLicenseType

Integer that defines the Platform License Type (single or multi-engine)

If PlatformLicenseType is not provided (blank), the existing Platform License Type is cleared.

- 1 = Single Engine
- 2 = Multi-Engine

IgObjects

Represents a collection of gObject objects.

IgObjects class members

This class has the following members.

Operations

CommandResults property

Returns the CommandResults Object, which has the last method call's result.

Area property

Sets the area for the given set of gObjects.

Host property

Sets the host for the given set of gObjects.

AddFromCollection method

Adds objects to the list from another list.

Upload method

Uploads AutomationObject configuration changes made in the runtime.

CheckIn method

Checks in the gObjects in the collection.

Add method

Adds an object to the list.

Container property

Sets the container for the given set of gObjects.

ExportObjects method

Exports object from the Galaxy.

ExportObjectsAsProtected method

Exports objects from the Galaxy as protected.

DeleteAllObjects method

Deletes all objects in the collection.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

CheckOut method

Checks out the gObjects in the collection.

SecurityGroup property

Sets the security group for the given set of gObjects.

Undeploy method

Undeploys the instances in the given collection of objects.

Deploy Method

Deploys the instances in the given collection of objects.

Item property

Returns the gObject object by index.

Count property

Returns the number of gObjects in the collection.

UndoCheckOut method

Undoes check-out on the gObjects in the collection.

DeployEx method

Deploy the instances in the given collection of objects.

UndeployEx method

Undeploy the instances in the given collection of objects.

UploadEx method

Uploads AutomationObject configuration changes made in the runtime.

CommandResults property

Returns the CommandResults object, which has the last method call's result.

Class

IgObjects

Syntax

[C#]

```
ICommandResults CommandResults { get; };
```

[Visual C++]

```
HRESULT CommandResults(
   [out, retval] ICommandResults** CommandResults
);
```

Parameters

CommandResults

Returns the CommandResults object.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Area property

Set the area for the given set of gObjects.

Class

IgObjects

Syntax

[C#]

```
string Area { set; };
```

[Visual C++]

```
HRESULT Area(
   in, retval] BSTR* NewArea
);
```

Parameters

NewArea

The name of the Area object, or blank or unspecified to unassign from existing area.

Remarks

The area of an AppObject gObject cannot be set directly. Set the container or host. The gObjects do not need to be checked out; the operation is immediate.

Host property

Sets the host for the given set of gObjects.

Class

IgObjects

Syntax

[C#]

```
string Host { set; };
```

[Visual C++]

```
HRESULT Host(
   [in] BSTR NewHost
);
```

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Parameters

NewHost

The name of the host object, or blank or unspecified to unassign from existing area.

Remarks

If the gObject is an ApplicationObject, its area and container are also updated. The gObjects do not need to be checked out; the operation is immediate.

AddFromCollection method

Add objects to the list from another list.

Class

IgObjects

Syntax

[C#]

```
void AddFromCollection(
   IgObjects gObjects
);
```

[Visual C++]

```
HRESULT AddFromCollection(
   [in] IgObjects* gObjects
);
```

Parameters

gObjects

Collection representing templates or instances.

Upload method

Uploads AutomationObject configuration changes made at run time.

Class

IgObjects

Syntax

[C#]

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

```
void Upload(
   EAutomaticallyUndocheckout automaticallyUndocheckout,
   ESkipOtherUsersCheckedOutObjects skipOtherUsersCheckedOutObjects,
   ESkipObjectsWithPendingUpdates skipObjectsWithPendingUpdates
);
```

[Visual C++]

```
HRESULT Upload(
   [in, optional, defaultvalue( doAutomaticallyUndocheckout )] EAutomaticallyUndocheckout
   automaticallyUndocheckout,
   [in, optional, defaultvalue( doSkipOtherUsersCheckedOutObjects )]
   ESkipOtherUsersCheckedOutObjects skipOtherUsersCheckedOutObjects,
   [in, optional, defaultvalue( doSkipObjectsWithPendingUpdates )]
   ESkipObjectsWithPendingUpdates skipObjectsWithPendingUpdates
);
```

Parameters

automaticallyUndocheckout

Automatically undo the check out.

skipOtherUsersCheckedOutObjects

Skip other users' checked out objects.

skipObjectsWithPendingUpdates

Skip objects with pending updates.

CheckIn method

Check in the gObjects in the collection.

Class

IgObjects

Syntax

[C#]

```
void CheckIn(
   string CheckInComment
);
```

[Visual C++]

```
HRESULT CheckIn(
   [in, optional] BSTR CheckInComment
);
```

Parameters

CheckInComment

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

The check-in comment.

Add method

Add an object to the list.

Class

IgObjects

Syntax

[C#]

```
void Add(
   igObject gObject
);
```

[Visual C++]

```
HRESULT Add(
   [in] igObject* gObject
);
```

Parameters

gObject

A gObject to be added to the list.

Container property

Sets the container for the given set of gObjects.

Class

IgObjects

Syntax

[C#]

```
string Container { set; };
```

[Visual C++]

```
HRESULT Container(
   [in] BSTR NewContainer
);
```

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Parameters

NewContainer

The name of the container, or blank to uncontain.

Remarks

If the gObject is an ApplicationObject, its area and host are also updated. The gObjects do not need to be checked out; the operation is immediate.

ExportObjects method

Exports an object from the Galaxy.

Class

IgObjects

Syntax

[C#]

```
void ExportObjects(
   EExportType exportType, string outputFile
);
```

[Visual C++]

```
HRESULT ExportObjects(
   [in] EExportType exportType,
   [in] BSTR outputFile
);
```

Parameters

exportType

Indicates whether to export to aaPKG file.

outputFile

The name of the output file. Appends extension if one is not specified.

ExportObjectsAsProtected method

Exports object templates and graphics from the Galaxy as protected.

Class

IgObjects

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Syntax

[C#]

```
void ExportObjectsAsProtected(
   string outputFile
);
```

[Visual C++]

```
HRESULT ExportObjectsAsProtected(
   [in] BSTR outputFile
);
```

Parameters

outputFile

The name of the output file. Appends extension if one is not specified.

Remarks

There is no exportType parameter as only objects and templates, not instances, can be exported as protected. The default exportType is "exportAsPDF".

DeleteAllObjects method

Delete all objects in the collection.

Class

IgObjects

Syntax

[C#]

```
void DeleteAllObjects();
```

[Visual C++]

```
HRESULT DeleteAllObjects();
```

CheckOut method

Check out the gObjects in the collection.

Class

IgObjects

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Syntax

[C#]

```
void CheckOut();
```

[Visual C++]

```
HRESULT CheckOut();
```

SecurityGroup property

Sets the security group for the given set of gObjects.

Class

IgObjects

Syntax

[C#]

```
string SecurityGroup { set; };
```

[Visual C++]

```
HRESULT SecurityGroup(
   [in] BSTR rha
);
```

Parameters

rhs

The name of the security group.

Undeploy method

Undeploy the instances in the given collection of objects.

Class

IgObjects

Syntax

[C#]

```
void Undeploy(
   EForceOffScan UndeployRule,
   bool markAsUndeployedOnFailure
);
```

[Visual C++]

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

```
HRESULT Undeploy(
   [in, optional, defaultvalue( dontForceOffScan )] EForceOffScan UndeployRule,
   [in, optional, defaultvalue( 0 )] VARIANT_BOOL markAsUndeployedOnFailure
);
```

Parameters

UndeployRule

Indicates whether to force an object off-scan if it is running on-scan prior to undeployment.

markAsUndeployedOnFailure

On failure, set deploy status as undeployed.

Deploy Method

Deploy the instances in the given collection of objects.

Class

IgObjects

Syntax

[C#]

```
void Deploy(
   EActionForCurrentlyDeployedObjects actionForDeployedObjects,
   ESkipIfCurrentlyUndeployed skipIfCurrentlyUndeployed,
   EDeployOnScan deployOnScan,
   EForceOffScan forceOffScan,
   bool markAsDeployedOnStatusMismatch
);
```

[Visual C++]

```
HRESULT Deploy
(
   [in, optional, defaultvalue( skipDeploy )] EActionForCurrentlyDeployedObjects
   actionForDeployedObjects,
   [in, optional, defaultvalue( dontSkipIfCurrentlyUndeployed )] ESkipIfCurrentlyUndeployed
   skipIfCurrentlyUndeployed,
   [in, optional, defaultvalue( doDeployOnScan )] EDeployOnScan deployOnScan,
   [in, optional, defaultvalue( doForceOffScan )] EForceOffScan forceOffScan,
   [in, optional, defaultvalue( 1 )] VARIANT_BOOL markAsDeployedOnStatusMismatch
);
```

Parameters

actionForDeployedObjects

Action for currently deployed object.

skipIfCurrentlyUndeployed

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Action for currently undeployed object.

deployOnScan

Deploy on scan.

forceOffScan

Force off scan.

markAsDeployedOnStatusMismatch

Deploy status mismatch.

Item property

Returns the gObject object by index.

Class

IgObjects

Syntax

[C#]

```
IgObject this[object gObjectIdentifier] { get; };
```

[Visual C++]

```
HRESULT Item(
   [in] VARIANT gObjectIdentifier,
   [out, retval] IgObject**
);
```

Parameters

gObjectIdentifier

The index of the gObject. This is a numeric value from 1 to Count, or the name of the gObject.

IgObject

The returned gObject object.

Count property

Returns the number of gObjects in the collection.

Class

IgObjects

Syntax

[C#]

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

```
int count { get; };
```

[Visual C++]

```
HRESULT Count(
   [out, retval] LONG*
);
```

UndoCheckOut method

Reverses the check out operation the gObjects in the collection without making changes.

Class

IgObjects

Syntax

[C#]

```
void UndoCheckOut();
```

[Visual C++]

```
HRESULT UndoCheckOut();
```

DeployEx method

Deploys the instances in the given collection of objects.

Class

IgObjects

Syntax

[C#]

```
void DeployEx(
   EActionForCurrentlyDeployedObjects actionForDeployedObjects,
   ESkipIfCurrentlyUndeployed skipIfCurrentlyUndeployed,
   EDeployOnScan deployOnScan,
   EForceOffScan forceOffScan,
   bool markAsDeployedOnStatusMismatch,
   string strCustomData
);
```

[Visual C++]

```
HRESULT DeployEx(
   // Action for currently deployed object
   [in, optional, defaultvalue( skipDeploy )] EActionForCurrentlyDeployedObjects
   actionForDeployedObjects,
   // Action for currently undeployed object
   [in, optional, defaultvalue( dontSkipIfCurrentlyUndeployed )] ESkipIfCurrentlyUndeployed
```

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

```
   skipIfCurrentlyUndeployed,
   // Deploy on scan
   [in, optional, defaultvalue( doDeployOnScan )] EDeployOnScan deployOnScan,
   // Force off scan
   [in, optional, defaultvalue( doForceOffScan )] EForceOffScan forceOffScan,
   // Deploy status mismatch
   [in, optional, defaultvalue( 1 )] VARIANT_BOOL markAsDeployedOnStatusMismatch,
   // Data used by custom category package
   [in, optional, defaultvalue("")] BSTR strCustomData
);
```

UndeployEx method

Undeploys the instances in the given collection of objects.

Class

IgObjects

Syntax

[C#]

```
void Undeploy(
   EForceOffScan UndeployRule,
   bool markAsUndeployedOnFailure
   string strCustomData
);
```

[Visual C++]

```
HRESULT UndeployEx(
   // Indicates whether to force an object off-scan
   // if it's running on-scan prior to undeployment.
   [in, optional, defaultvalue( dontForceOffScan )] EForceOffScan UndeployRule,
   // On Failure, set deploy status as undeployed
   [in, optional, defaultvalue( 0 )] VARIANT_BOOL markAsUndeployedOnFailure,
   // Data used by custom category package
   [in, optional, defaultvalue("")] BSTR strCustomData
);
```

UploadEx method

Uploads AutomationObject configuration changes made at run time.

Class

IgObjects

Syntax

[C#]

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

```
void UploadEx(
   EAutomaticallyUndocheckout automaticallyUndocheckout,
   ESkipOtherUsersCheckedOutObjects skipOtherUsersCheckedOutObjects,
   ESkipObjectsWithPendingUpdates skipObjectsWithPendingUpdates,
   string strCustomData
);
```

[Visual C++]

```
HRESULT UploadEx(
   // Automatically Undocheckout
   [in, optional, defaultvalue(doAutomaticallyUndocheckout )] EAutomaticallyUndocheckout
   automaticallyUndocheckout,
   // Skip other users CheckedOuts
   [in, optional, defaultvalue( doSkipOtherUsersCheckedOutObjects )]
   ESkipOtherUsersCheckedOutObjects skipOtherUsersCheckedOutObjects,
   // Skip objects with pending updates
   [in, optional, defaultvalue( doSkipObjectsWithPendingUpdates )]
   ESkipObjectsWithPendingUpdates skipObjectsWithPendingUpdates,
   // Data used by custom category package
   [in, optional, defaultvalue("")] BSTR strCustomData
);
```

IGRAccess

Main interface implemented by GRAccess used by clients to gain access to the Galaxy Repository.

IGRAccess class members

This class has the following members.

Operations

CommandResult property

Returns the *CommandResult* Object, which has the last method call's result.

CreateGalaxy method

Create a new Galaxy on the GR node.

DeleteGalaxy method

Delete a Galaxy.

QueryGalaxies method

Query available Galaxies on a given GR node.

QueryGalaxiesEx method

Queries a given Galaxy Repository (GR) node to list all available Galaxies, including the Modern InTouch ViewApp Galaxies. You do not need to be logged in to run this query.

CreateGalaxyFromTemplate method

Creates a new Galaxy from the template.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

CommandResult property

Returns the *CommandResult* object, which has the last method call's result.

Class

IGRAccess

Syntax

[C#]

```
ICommandResult CommandResult { get; };
```

[Visual C++]

```
HRESULT CommandResult(
   [out, retval] ICommandResult** CommandResult
);
```

Parameters

CommandResult

Returns the *CommandResult* object.

CreateGalaxy method

Creates a new Galaxy on the GR node.

Class

IGRAccess

Syntax

[C#]

```
void CreateGalaxy(
   string galaxyName,
   string GRNodeName,
   bool enableSecurity,
   EAuthenticationMode AuthenticationMode,
   string osUserName
);
```

[Visual C++]

```
HRESULT CreateGalaxy(
   [in] BSTR galaxyName,
   [in, optional] BSTR grNodeName,
   [in, optional, defaultvalue( -1 )]
   VARIANT_BOOL enableSecurity,
   [in, optional, defaultvalue ( osAuthenticationMode )] EAuthenticationMode
```

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

```
   authenticationMode,
   [in, optional] BSTR osUserName
);
```

Parameters

galaxyName

The name to give the new Galaxy.

grNodeName

The name of the GR node (computer name). Blank or omitted means the current computer is the Galaxy Repository.

enableSecurity

Indicates whether security should be enabled.(valid only for Galaxy authentication).

authenticationMode

Indicates whether OS or Galaxy authentication should be used.

osUserName

The name of the OS user if osAuthenticationMode is specified. Ignored otherwise. If not specified in OS authentication mode, the user currently logged in to the OS is used.

DeleteGalaxy method

Deletes a Galaxy.

Class

IGRAccess

Syntax

[C#]

```
void DeleteGalaxy(
   string galaxyName,
   string GRNodeName
);
```

[Visual C++]

```
HRESULT DeleteGalaxy(
   [in] BSTR galaxyName,
   [in, optional] BSTR grNodeName
);
```

Parameters

galaxyName

The name of the Galaxy to be deleted.

grNodeName

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

The name of the GR node (computer name). Blank or omitted means the current computer is the Galaxy Repository.

QueryGalaxies method

Queries available Galaxies on a given GR node.

Class

IGRAccess

Syntax

[C#]

```
IGalaxies QueryGalaxies(
   string GRNodeName
);
```

[Visual C++]

```
HRESULT QueryGalaxies(
   [in, optional] BSTR grNodeName,
   [out, retval] IGalaxies** galaxies
);
```

Parameters

grNodeName

The name of the GR node (computer name). Blank or omitted means the current computer is the Galaxy Repository.

galaxies

The list of galaxies.

QueryGalaxiesEx method

Queries a given Galaxy Repository (GR) node to list all available Galaxies, including the Modern InTouch ViewApp Galaxies. You do not need to be logged in to run this query.

Class

GRAccess

Syntax

[C#]

```
IGalaxies QueryGalaxiesEx(string GRNodeName);
```

[Visual C++]

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

```
HRESULT QueryGalaxiesEx
   (
      [in, optional] BSTR grNodeName,
      [out, retval] IGalaxies** galaxies
   );
```

Parameters

grNodeName

Name of the Galaxy Repository to be queried.

CreateGalaxyFromTemplate method

Creates a new Galaxy from the template.

Class

IGRAccess

Syntax

[C#]

```
   void CreateGalaxyFromTemplate(
   string createGalaxyTemplateName,
   string galaxyName,
   string GRNodeName
);
```

[Visual C++]

```
HRESULT CreateGalaxyFromTemplate(
   [in] BSTR CreateGalaxyTemplateName,
   [in] BSTR galaxyName,
   [in, optional] BSTR grNodeName,
[out, retval] IGalaxies** galaxies
);
```

Parameters

createGalaxyTemplateName

The template name to create the Galaxy.

galaxyName

The name to give the new Galaxy.

grNodeName

The name of the GR node (computer name). Blank or omitted means the current computer is the Galaxy Repository.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

IInstance

Represents an instance object.

IInstance class members

This class has the following members.

Operations

AddExtensionPrimitive method

Add an extension primitive.

AddUDA Method

Add a UDA.

Area property

Get or set the area of an object.

Attributes property

Gets the collection of attributes for this object.

BasedOn property

Returns the name of the base template for this object.

Category property

Returns the category of this object.

CategoryGUID property

Returns the category GUID for this object.

CheckedOutBy property

Returns the name of the user who has checked out this object.

CheckIn method

Checks in an object.

CheckOut method

Checks out an object.

CheckoutStatus property

Returns the check out status of the object.

CommandResult property

Returns the CommandResult object, which has the results from the object's last method call.

ConfigurableAttributes property

Returns a collection of this object's attributes that can be modified.

ConfigVersion property

Returns the this object's configuration version number.

ContainedName property

Sets or returns the object's contained name.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Container property

Sets or returns the object's container.

DeleteExtensionPrimitive method

Delete an extension primitive.

DeleteInstance method

Delete this instance.

DeleteUDA method

Delete a UDA.

Deploy method

Deploy this instance.

DeployedVersion property

The configuration version of this instance at the time it was last deployed.

DeployEx method

Deploy this instance.

DeploymentStatus property

Indicates whether the instance is currently deployed and if so, if it has pending changes.

DerivedFrom property

Returns the name of the object's template.

EditStatus property

Edit the status of this object.

Errors property

Retrieves the list of errors generated when the object was last validated and saved.

GetExtendedAttributes method

Get the extended attributes in the hierarchy.

GetObjectHelpURL method

Returns the URL to where the object help is stored on the Galaxy repository.

HierarchicalName property

Returns the hierarchical name of this object.

Host property

Returns the name of this object's host.

RenameExtensionPrimitive method

Rename the object extension primitive.

RenameUDA method

Change the name of the UDA.

Save method

Save the object.

Tagname property

Returns or sets the object's tagname.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Undeploy method

Undeploy this instance.

UndeployEx method

Undeploy this instance.

UndoCheckOut method

Reverse the check out operation without saving changes.

Unload method

Unload the object cache.

UpdateUDA method

Save the UDA information.

Upload method

Uploads AutomationObject configuration changes made in the runtime.

UploadEx method

Uploads the AutomationObject configuration changes.

ValidationStatus property

Returns this object's validation status.

Warnings property

Returns the list of warnings generated the last time the object was validated and saved.

AddExtensionPrimitive method

Adds an extension primitive.

Class

IInstance

Syntax

[C#]

```
void AddExtensionPrimitive(
   string ExtensionType,
   string ExtensionPrimitiveName,
   bool IsObjectExtension
);
```

[Visual C++]

```
HRESULT AddExtensionPrimitive(
   // Type of extension
   [in] BSTR ExtensionType,
   // Name of extension primitive to be added -
   // For attribute extension primitive - Fully
   // qualified name of attribute being extended
   // For object extension primitive - New name
   // specified by user for object extension primitive
```

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

```
   [in] BSTR ExtensionPrimitiveName,
   // If wszName is the new name specified by user,
   // this field will be set to true. for example
   //when add script pass true
   [in, optional, defaultvalue( -1 )]VARIANT_BOOL IsObjectExtension
);
```

AddUDA Method

Adds a UDA.

Class

IInstance

Syntax

[C#]

```
void AddUDA(
   string UDAName,
   MxDataType DataType,
   MxAttributeCategory category,
   MxSecurityClassification Security,
   bool IsArray,
   object ArrayElementCount
);
```

[Visual C++]

```
// Add a UDA
//
// Returns:
// S_OK - Successfully add a UDA
HRESULT AddUDA(
   //Same XML format. But only one is allowed.
   [in] BSTR UDAName,
   [in] enum MxDataType Datatype,
   [in] enum MxAttributeCategory Category,
   [in] enum MxSecurityClassification Security,
   [in] VARIANT_BOOL IsArray,
   [in] VARIANT ArrayElementCount
);
```

Area property

Gets or sets the area of this object. An AppObject area must be set using the SetContainer or SetHost methods. The object does not need to be checked out, and the operation is immediate.

Class

IInstance

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Syntax

[C#]

```
string Area { set; get; };
```

[Visual C++]

```
// Returns name of the Area object, or blank if the
// object doesn't have an Area.
[propget]
HRESULT Area(
   [out, retval] BSTR* theArea
);
// Set the Area of this object.
[propput]
HRESULT Area(
   // The name of the Area, or blank to unassign
   // from existing Area.
   [in] BSTR NewArea
);
```

Attributes property

Returns the collection of attributes of this object.

Class

IInstance

Syntax

[C#]

```
IAttributes Attributes { get; };
```

[Visual C++]

```
// The collection of Attributes of this object.
// Configuring attributes requires that the object
// be checked out.
HRESULT Attributes(
   // The collection of Attributes of this object.
   [out, retval] IAttributes** theAttributes
);
```

BasedOn property

Returns the name of the base template for this object.

Class

IInstance

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Syntax

[C#]

```
string basedOn { get; };
```

[Visual C++]

```
// The name of the base template (root ancestor) of this object.
HRESULT BasedOn(
   // The name of the base template (root ancestor)
   // of this object.
   [out, retval] BSTR* theBaseTemplate
);
```

Category property

Returns the category of this object.

Class

IInstance

Syntax

[C#]

```
ECATEGORY category { get; };
```

[Visual C++]

```
// The Category of the object.
HRESULT Category(
   // The Category of the object.
   [out, retval] enum ECATEGORY* theCategory
);
```

CategoryGUID property

Gets the category GUID of this object.

Class

IInstance

Syntax

[C#]

```
VBGUID CategoryGUID { get; };
```

[Visual C++]

```
// The Category GUID of the object.
HRESULT CategoryGUID(
```

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

```
   // The Category of the object.
   [out, retval] VBGUID *categoryGUID
);
```

CheckedOutBy property

Returns the name of the user who has checked out this object.

Class

IInstance

Syntax

[C#]

```
string checkedOutBy { get; };
```

[Visual C++]

```
// The name of the user that checked out this object.
HRESULT CheckedOutBy(
   // The name of the user that checked out this object.
   [out, retval] BSTR* CheckedOutByUserName
);
```

CheckIn method

Checks in this object.

Class

IInstance

Syntax

[C#]

```
void CheckIn(
   string CheckInComment
);
```

[Visual C++]

```
// Checks in an object.
// Called on a checked-out object after it is
// configured and saved.
HRESULT CheckIn(
   // Check in an object.
   [in, optional] BSTR CheckInComment
);
```

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

CheckOut method

Checks out an object.

Class

IInstance

Syntax

[C#]

```
void CheckOut();
```

[Visual C++]

```
// Checks out an object.
HRESULT CheckOut();
```

CheckoutStatus property

Returns the check out status of the object.

Class

IInstance

Syntax

[C#]

```
ECheckoutStatus CheckoutStatus { get; };
```

[Visual C++]

```
// Get the check-out status of the object.
HRESULT CheckoutStatus(
   [out, retval] enum ECheckoutStatus* theCheckoutStatus
);
```

CommandResult property

Returns the CommandResult object, which has the results from the last method call.

Class

IInstance

Syntax

[C#]

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

```
ICommandResult CommandResult { get; };
```

[Visual C++]

```
HRESULT CommandResult(
   [out, retval] ICommandResult** CommandResult
);
```

Parameters

CommandResult

Returns the CommandResult object.

ConfigurableAttributes property

Returns a collection of this object's attributes that can be configured.

Class

IInstance

Syntax

[C#]

```
IAttributes ConfigurableAttributes { get; };
```

[Visual C++]

```
// The collection of Attributes of this object that
// can be configured.
// Configuring attributes requires that the object
// be checked out.
HRESULT ConfigurableAttributes(
   [out, retval] IAttributes** theConfigurableAttributes
);
```

ConfigVersion property

Returns this object's configuration version number. The configuration version is incremented each time the object is checked in.

Class

IInstance

Syntax

[C#]

```
int ConfigVersion { get; };
```

[Visual C++]

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

```
// The configuration version of the object.
HRESULT ConfigVersion(
   [out, retval] LONG* theConfigVersion
);
```

ContainedName property

Sets or returns this object's contained name. When setting the contained name, the object does not need to be checked out, and the operation is immediate.

Class

IInstance

Syntax

[C#]

```
string ContainedName { set; get; };
```

[Visual C++]

```
//
// Returns Contained name of this object (eg. Inlet)
[propget]
HRESULT ContainedName(
   // The contained name of this object (eg. Inlet)
   [out, retval] BSTR* theContainedName
);
// Sets the contained name of this object (eg. Inlet)
// The object does not need to be checked out; the
// operation is immediate.
[propput]
HRESULT ContainedName(
   // The contained name of this object (eg. Inlet)
   [in] BSTR newContainedName
);
```

Container property

Returns or sets the container for this object. When setting the container, the object does not need to be checked out, and the change is immediate. For an ApplicationObject, the ApplicationObject's Area and Host area are also updated.

Class

IInstance

Syntax

[C#]

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

```
string Container { set; get; };
```

[Visual C++]

```
// Returns name of the Container object, or blank if
//object is not contained.
[propget]
HRESULT Container(
   // The name of the Container object, or blank if
   // the object is not contained.
   [out, retval] BSTR* theContainer
);
// Set the Container for this object.
[propput]
HRESULT Container(
   // The name of the container, or blank to uncontain.
   [in] BSTR NewContainer
);
```

DeleteExtensionPrimitive method

Deletes an extension.

Class

IInstance

Syntax

[C#]

```
void DeleteExtensionPrimitive(
   string ExtensionType,
   string ExtensionPrimitiveName
);
```

[Visual C++]

```
// Delete an extension primitive
// Returns S_OK
HRESULT DeleteExtensionPrimitive(
   // Type of extension
   [in] BSTR ExtensionType,
   // Name of Extension primitive to be deleted.
   [in] BSTR ExtensionPrimitiveName
);
```

DeleteInstance method

Delete an instance.

Class

IInstance

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Syntax

[C#]

```
void DeleteInstance(
   EForceDeleteInstanceOption ForceDeleteOption
);
```

[Visual C++]

```
HRESULT DeleteInstance(
   [in, optional, defaultvalue( dontForceInstanceDelete )] EForceDeleteInstanceOption
   ForceDeleteOption
);
```

Parameters

ForceDeleteOption

Various options to indicate whether deletion of the object should be forced.

DeleteUDA method

Deletes a UDA.

Class

IInstance

Syntax

[C#]

```
void DeleteUDA(
   string UDAName
);
```

[Visual C++]

```
// Delete a UDA
// Returns:
// S_OK - Successfully delete a UDA
HRESULT DeleteUDA(
   // Delete a UDA by UDA name.
   [in] BSTR UDAName
);
```

Deploy method

Deploys the instance.

Class

IInstance

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Syntax

[C#]

```
void Deploy(
   EActionForCurrentlyDeployedObjects actionForDeployedObjects,
   ESkipIfCurrentlyUndeployed skipIfCurrentlyUndeployed,
   EDeployOnScan deployOnScan,
   EForceOffScan forceOffScan,
   ECascade cascade,
   bool markAsDeployedOnStatusMismatch
);
```

[Visual C++]

```
HRESULT Deploy(
   [in, optional, defaultvalue( skipDeploy )] EActionForCurrentlyDeployedObjects
   actionForDeployedObjects,
   [in, optional, defaultvalue( dontSkipIfCurrentlyUndeployed )] ESkipIfCurrentlyUndeployed
   skipIfCurrentlyUndeployed,
   [in, optional, defaultvalue( doDeployOnScan )] EDeployOnScan deployOnScan,
   [in, optional, defaultvalue( doForceOffScan )] EForceOffScan forceOffScan,
   [in, optional, defaultvalue( doCascade )] ECascade cascade,
   [in, optional, defaultvalue( 1 )] VARIANT_BOOL markAsDeployedOnStatusMismatch
);
```

Parameters

actionForDeployedObjects

Action for currently deployed object.

skipIfCurrentlyUndeployed

Action for currently undeployed object.

deployOnScan

Deploy on scan.

forceOffScan

Force off scan.

cascade

Cascade.

markAsDeployedOnStatusMismatch

Deploy status mismatch.

DeployedVersion property

The configuration version of this instance at the time it was last deployed.

Class

IInstance

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Syntax

[C#]

```
int DeployedVersion { get; };
```

[Visual C++]

```
HRESULT DeployedVersion(
   [out, retval] LONG*
);
```

DeployEx method

Deploy this instance.

Class

IInstance

Syntax

[C#]

```
void DeployEx(
   EActionForCurrentlyDeployedObjects actionForDeployedObjects,
   ESkipIfCurrentlyUndeployed skipIfCurrentlyUndeployed,
   EDeployOnScan deployOnScan,
   EForceOffScan forceOffScan,
   ECascade cascade,
   bool markAsDeployedOnStatusMismatch,
   string strCustomData
);
```

[Visual C++]

```
// Deploy this Instance.
HRESULT DeployEx(
   // Action for currently deployed object
   [in, optional, defaultvalue( skipDeploy )] EActionForCurrentlyDeployedObjects
   actionForDeployedObjects,
   // Action for currently undeployed object
   [in, optional, defaultvalue( dontSkipIfCurrentlyUndeployed )] ESkipIfCurrentlyUndeployed
   skipIfCurrentlyUndeployed,
   // Deploy on scan
   [in, optional, defaultvalue( doDeployOnScan )] EDeployOnScan deployOnScan,
   // Force off scan
   [in, optional, defaultvalue( doForceOffScan )] EForceOffScan forceOffScan,
   // Cascade
   [in, optional, defaultvalue( doCascade )] ECascade cascade,
   // Deploy status mismatch
   [in, optional, defaultvalue( 1 )] VARIANT_BOOL markAsDeployedOnStatusMismatch,
   // Data used by custom category package
   [in, optional, defaultvalue("")] BSTR strCustomData
);
```

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

DeploymentStatus property

Indicates whether the instance is currently deployed and if so, if it has pending changes.

Class

IInstance

Syntax

[C#]

```
EDeploymentStatus DeploymentStatus { get; };
```

[Visual C++]

```
HRESULT DeploymentStatus(
   [out, retval] EDeploymentStatus* theDeploymentStatus
);
```

Parameters

theDeploymentStatus

Indicates whether the instance is currently deployed and if so, if it has pending changes.

DerivedFrom property

Returns the name of the object's template.

Class

IInstance

Syntax

[C#]

```
string DerivedFrom { get; };
```

[Visual C++]

```
// Returns the name of the template that this object
// was created from.
[propget]
HRESULT DerivedFrom(
   // The name of the template that this object was
   // created from.
   [out, retval] BSTR* theParentTemplate
);
```

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

EditStatus property

Edits the status of the object.

Class

IInstance

Syntax

[C#]

```
EEditStatus EditStatus { get; };
```

[Visual C++]

```
// The edit status of the object.
HRESULT EditStatus(
   // The edit status of the object.
   [out, retval] enum EEditStatus* theEditStatus
);
```

Errors property

Retrieves the list of errors generated when the object was last validated and saved.

Class

IInstance

Syntax

[C#]

```
string[] Errors { get; };
```

[Visual C++]

```
// The list of errors generated the last
// time the object was validated and saved.
[propget]
HRESULT Errors(
   [out, retval] SAFEARRAY(BSTR)* theErrors
);
```

GetExtendedAttributes method

Retrieves the extended attributes in the hierarchy.

Class

IInstance

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Syntax

[C#]

```
IAttributes GetExtendedAttributes(
   string AttributeName,
   int upto_level,
   MxAttributeCategory[] ReturnOnlyTheseCategories
);
```

[Visual C++]

```
//Get Extended Attributes in Hierarchy
HRESULT GetExtendedAttributes(
   // Attribute full name which is extended
   [in] BSTR AttributeName,
   // Up to how many level user needs extended
   // primitives
   // -1 will return all levels
   [in] int upto_level,
   // Return attributes which are only these
   // categories. If safearray is empty, returns all
   // attributes.
   [in]SAFEARRAY(MxAttributeCategory) ReturnOnlyTheseCategories,
   // Return attributes
   [out,retval]IAttributes** theAttributes
);
```

GetObjectHelpURL method

Returns the URL where the object help is stored on the Galaxy repository.

Class

IInstance

Syntax

[C#]

```
string GetObjectHelpURL();
```

[Visual C++]

```
// Returns the URL to where the Object Help is
// stored on the Galaxy Repository.
// The client can display the URL in
// InternetExplorer or use web publishing
// to change the data.
HRESULT GetObjectHelpURL(
   // Returns the URL for the Object Help
   // location on the GR.
   [out, retval] BSTR*
);
```

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

HierarchicalName property

Returns the hierarchical name of this object.

Class

IInstance

Syntax

[C#]

```
string HierarchicalName { get; };
```

[Visual C++]

```
// The Hierarchical name of this object. (eg.
// Reactor.Tank.Inlet)
HRESULT HierarchicalName(
   [out, retval] BSTR* theHierarchicalName
);
```

Host property

Returns the name of this object's host.

Class

IInstance

Syntax

[C#]

```
string Host { set; get; };
```

[Visual C++]

```
// Retrieve the name of the Host for this object, or
// blank if unassigned.
HRESULT Host(
   // The name of the Host for this object.
   [out, retval] BSTR* theHost
);
```

RenameExtensionPrimitive method

Renames the object extension primitive.

Class

IInstance

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Syntax

[C#]

```
void RenameExtensionPrimitive(
   string OldPrimitiveName,
   string NewPrimitiveName
);
```

[Visual C++]

```
// Rename an object extension primitive
// Returns S_OK
HRESULT RenameExtensionPrimitive(
   // Old name of primitive extension
   [in] BSTR OldPrimitiveName,
   // New name specified by user
   [in] BSTR NewPrimitiveName
);
```

RenameUDA method

Changes the name of a UDA.

Class

IInstance

Syntax

[C#]

```
void RenameUDA(
   string OldUDAName,
   string NewUDAName
);
```

[Visual C++]

```
// Rename a UDA
// Returns:
// S_OK - Successfully rename a UDA
HRESULT RenameUDA(
   // The old external name of the primitive
   [in] BSTR OldUDAName,
   // The new external name of the primitive
   [in] BSTR NewUDAName
);
```

Save method

Saves the object.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Class

IInstance

Syntax

[C#]

```
void Save();
```

[Visual C++]

```
// Saves the object after it's configured.
// Object must be checked out.
HRESULT Save();
```

Tagname property

Returns or sets this object’s tagname.

Class

IInstance

Syntax

[C#]

```
string Tagname { set; get; };
```

[Visual C++]

```
// Returns the Tagname of this object.
[propget]
HRESULT Tagname(
   // The Tagname of this object.
   [out, retval] BSTR* theTagname
);
// Sets the Tagname of this object.
// The object does not need to be checked out;
// the operation is immediate.
[propput]
HRESULT Tagname(
   // The Tagname of this object.
   [in] BSTR newTagName
);
```

Undeploy method

Undeploy this instance.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Class

IInstance

Syntax

[C#]

```
void Undeploy(
   EForceOffScan UndeployRule,
   ECascade cascade,
   bool markAsUndeployedOnFailure
);
```

[Visual C++]

```
HRESULT Undeploy(
   [in, optional, defaultvalue( dontForceOffScan )] EForceOffScan UndeployRule,
   [in, optional, defaultvalue( doCascade )] ECascade cascade,
   [in, optional, defaultvalue( 0 )] VARIANT_BOOL markAsUndeployedOnFailure
);
```

Parameters

UndeployRule

Indicates whether to force an object off-scan if it is running on-scan prior to undeployment.

cascade

Cascade.

markAsUndeployedOnFailure

On failure, set deploy status as undeployed.

UndeployEx method

Undeploy this instance.

Class

IInstance

Syntax

[C#]

```
void Undeploy(
   EForceOffScan UndeployRule,
   ECascade cascade,
   bool markAsUndeployedOnFailure
);
```

[Visual C++]

```
// Undeploy this Instance.
```

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

```
HRESULT UndeployEx(
   // Indicates whether to force an object off-scan
   // if it's running on-scan prior to undeployment.
   [in, optional, defaultvalue( dontForceOffScan )] EForceOffScan UndeployRule,
   // Cascade
   [in, optional, defaultvalue( doCascade )] ECascade cascade,
   // On Failure, set deploy status as undeployed
   [in, optional, defaultvalue( 0 )] VARIANT_BOOL markAsUndeployedOnFailure,
   // Data used by custom category package
   [in, optional, defaultvalue("")] BSTR strCustomData
);
```

UndoCheckOut method

Reverses the check out operation without making changes.

Class

IInstance

Syntax

[C#]

```
void UndoCheckOut();
```

[Visual C++]

```
// Undo check-out of this object.
HRESULT UndoCheckOut();
```

Unload method

Unloads the object cache, releasing internal resources obtained by the object during operations. After executing this method, the object is still available. Internal resources will be reloaded as necessary.

Class

IInstance

Syntax

[C#]

```
void Unload();
```

[Visual C++]

```
// Unloads the gObject cache. This method doesn't
// affect functionality.
HRESULT Unload();
```

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

UpdateUDA method

Saves the UDA information.

Class

IInstance

Syntax

[C#]

```
void UpdateUDA(
   string wszUDAInfo,
   MxDataType DataType,
   MxAttributeCategory category,
   MxSecurityClassification Security,
   bool IsArray,
   object ArrayElementCount
);
```

[Visual C++]

```
// Update a UDA
// Returns:
// S_OK - Successfully update a UDA
HRESULT UpdateUDA(
   // Same XML format. But only one is allowed. The
   // new info will overwrite the old info.
   [in] BSTR wszUDAInfo,
   [in] enum MxDataType Datatype,
   [in] enum MxAttributeCategory Category,
   [in] enum MxSecurityClassification Security,
   [in] VARIANT_BOOL IsArray,
   [in] VARIANT ArrayElementCount
);
```

Upload method

Uploads AutomationObject configuration changes made at run time.

Class

IInstance

Syntax

[C#]

```
void Upload(
   EAutomaticallyUndocheckout automaticallyUndocheckout,
   ESkipOtherUsersCheckedOutObjects skipOtherUsersCheckedOutObjects,
   ESkipObjectsWithPendingUpdates skipObjectsWithPendingUpdates
```

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

```
);
```

[Visual C++]

```
HRESULT Upload(
   [in, optional, defaultvalue( doAutomaticallyUndocheckout )] EAutomaticallyUndocheckout
   automaticallyUndocheckout,
   [in, optional, defaultvalue( doSkipOtherUsersCheckedOutObjects )]
   ESkipOtherUsersCheckedOutObjects skipOtherUsersCheckedOutObjects,
   [in, optional, defaultvalue( doSkipObjectsWithPendingUpdates )]
   ESkipObjectsWithPendingUpdates skipObjectsWithPendingUpdates
);
```

Parameters

automaticallyUndocheckout

Automatically undo the check out.

skipOtherUsersCheckedOutObjects

Skip other users' checked out objects.

skipObjectsWithPendingUpdates

Skip objects with pending updates.

UploadEx method

Saves the AutomationObject configuration changes.

Class

IInstance

Syntax

[C#]

```
void UploadEx(
   EAutomaticallyUndocheckout automaticallyUndocheckout,
   ESkipOtherUsersCheckedOutObjects skipOtherUsersCheckedOutObjects,
   ESkipObjectsWithPendingUpdates skipObjectsWithPendingUpdates,
   string strCustomData
);
```

[Visual C++]

```
// Uploads AutomationObject configuration changes
// made during runtime.
HRESULT UploadEx(
   // Automatically Undocheckout
   [in, optional, defaultvalue(doAutomaticallyUndocheckout )] EAutomaticallyUndocheckout
   automaticallyUndocheckout,
   // Skip other users CheckedOuts
   [in, optional, defaultvalue( doSkipOtherUsersCheckedOutObjects )]
   ESkipOtherUsersCheckedOutObjects skipOtherUsersCheckedOutObjects,
   // Skip objects with pending updates
```

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

```
   [in, optional, defaultvalue( doSkipObjectsWithPendingUpdates )]
   ESkipObjectsWithPendingUpdates skipObjectsWithPendingUpdates,
   // Data used by custom category package
   [in, optional, defaultvalue("")] BSTR strCustomData
);
```

ValidationStatus property

Returns this object's validation status.

Class

IInstance

Syntax

[C#]

```
EPACKAGESTATUS ValidationStatus { get; };
```

[Visual C++]

```
// The ValidationStatus of this object based on the
// last time the object was validated and saved.
HRESULT ValidationStatus(
   // The validation status of the object.
   [out, retval] enum EPACKAGESTATUS* theValidationStatus
);
```

Warnings property

Returns the list of warnings generated the last time the object was validated and saved.

Class

IInstance

Syntax

[C#]

```
string[] Warnings { get; };
```

[Visual C++]

```
// The list of warnings that got generated the last time the object was validated and
saved.
HRESULT Warnings(
   // The list of warnings that were generated the
   // last time the object was validated and saved.
   [out, retval] SAFEARRAY(BSTR)* theWarnings
);
```

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

IMxValue

Represents the fundamental variant object that contains a single data value or single array of data values of like type. The IMxValue is used to get and set data within the ArchestrA framework.

IMxValue class members

This class has the following members.

Operations

Clone method

Creates a duplicate MxValue object with a state identical to the current MxValue.

Empty method

Changes the MxValue's data type to MxNoData.

PutBoolean method

Stores a Boolean value in the MxValue.

PutInteger method

Stores a long value in the MxValue.

PutFloat method

Stores a float value in the MxValue.

PutDouble method

Stores a double value in the MxValue.

PutString method

Stores a Unicode string value in the MxValue.

PutTime method

Stores a VBFILETIME value in the MxValue.

PutElapsedTime method

Stores a VB_LARGE_INTEGER value in the MxValue.

PutMxReference method

Stores an MxReference in the MxValue.

PutMxStatus method

Stores an MxStatus in the MxValue.

PutMxDataType method

Stores an MxDataType in the MxValue.

PutMxSecurityClassification method

Stores an MxSecurityClassification in the MxValue.

PutMxDataQuality method

Stores an MxDataQuality in the MxValue.

PutCustomStruct method

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Stores a struct in the MxValue.

PutCustomEnum method

Stores an MxCustomEnum in the MxValue.

GetDataType method

Returns the data type of the value stored in the MxValue.

GetBoolean method

Returns the value stored in the MxValue as a VARIANT_BOOL.

GetInteger method

Returns the value stored in the MxValue as a long.

GetFloat method

Returns the value stored in the MxValue as a float.

GetDouble method

Returns the value stored in the MxValue as a double.

GetString method

Returns the value stored in the MxValue as a string.

GetTime method

Returns the value stored in the MxValue as a VBFILETIME.

GetElapsedTime method

Returns the value stored in the MxValue as a VB_LARGE_INTEGER.

GetMxReference method

Returns the value stored in the MxValue as an IMxReference *.

GetMxStatus method

Returns the MxStatus stored in the MxValue.

GetMxDataType method

Returns the value stored in the MxValue as an MxDataType.

GetMxSecurityClassification method

Returns the value stored in the MxValue as an MxSecurityClassification.

GetMxDataQuality method

Returns the value stored in the MxValue as an MxDataQuality.

GetCustomStruct method

Returns the qualified struct stored in the MxValue.

GetCustomEnum method

Returns the value stored in the MxValue as a qualified enum.

GetDimensionCount method

Indicates whether the MxValue holds an array.

PutElement method

Puts an element into an array held by an MxValue.

GetElement method

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Gets an element from an array held in an MxValue.

GetDimensionSize method

Gets the size of the array held by the MxValue.

PutCustomStructVB method

Stores a qualified struct in the MxValue for a VB client.

GetCustomStructVB method

Returns the qualified struct stored in the MxValue for a VB client.

PutInternationalStrings method

Initializes an MxValue with a set of internationalized strings.

PutInternationalStringsVB method

Initializes an MxValue with a set of internationalized strings for a VB client.

GetInternationalStrings method

Returns the internationalized strings stored by the MxValue.

GetInternationalStringsVB method

Returns the internationalized strings stored by the MxValue for a VB client.

GetInternationalString method

Returns the internationalized string associated with a particular local.

PutInternationalString method

Adds the internationalized string of a specific locale to the existing internationalized strings.

Clone method

Creates a duplicate MxValue object with a state identical to the current MxValue.

Class

IMxValue

Syntax

[C#]

```
void Clone(
   out MxValue ppMxValue
);
```

[Visual C++]

```
HRESULT Clone(
   [out] IMxValue **ppMxValue
);
```

Parameters

ppMxValue

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Receives the IMxValue interface pointer of the newly cloned object.

Returns

S_OK - Success

E_POINTER - An invalid pointer was passed in.

E_OUTOFMEMORY - Unable to allocate memory for the new MxValue object.

Empty method

Changes the MxValue's data type to MxNoData.

Class

IMxValue

Syntax

[C#]

```
void Empty();
```

[Visual C++]

```
HRESULT Empty();
```

Returns

S_OK - Success

PutBoolean method

Stores a Boolean value in the MxValue. The MxValue's data type will be MxBoolean.

Class

IMxValue

Syntax

[C#]

```
void PutBoolean(
   bool newVal
);
```

[Visual C++]

```
HRESULT PutBoolean(
   [in] VARIANT_BOOL newVal
);
```

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Parameters

newVal

The value that will be stored in the MxValue object.

Returns

S_OK - Success

PutInteger method

Stores a long value in the MxValue. The MxValue's data type will be MxInteger.

Class

IMxValue

Syntax

[C#]

```
void PutInteger(
   int newVal
);
```

[Visual C++]

```
HRESULT PutInteger(
   [in] long newVal
);
```

Parameters

newVal

The value that will be stored in the MxValue object.

Returns

S_OK - Success

PutFloat method

Stores a float value in the MxValue. The MxValue's data type will be MxFloat.

Class

IMxValue

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Syntax

[C#]

```
void PutFloat(
   float newVal
);
```

[Visual C++]

```
HRESULT PutFloat(
   [in] float newVal
);
```

Parameters

newVal

The value that will be stored in the MxValue object. A value that is NaN (not a number) is permitted.

Returns

S_OK - Success

PutDouble method

Stores a double value in the MxValue. The MxValue's data type will be MxDouble.

Class

IMxValue

Syntax

[C#]

```
void PutDouble(
   double newVal
);
```

[Visual C++]

```
HRESULT PutDouble(
   [in] double newVal
);
```

Parameters

newVal

The value that will be stored in the MxValue object. A value that is NaN (not a number) is permitted.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Returns

S_OK - Success

PutString method

Stores a Unicode string value in the MxValue. The MxValue's data type will be MxString.

Class

IMxValue

Syntax

[C#]

```
void PutString(
   string newVal
);
```

[Visual C++]

```
HRESULT PutString(
   [in, string] LPCWSTR newVal
);
```

Parameters

newVal

The value that will be stored in the MxValue object.

Returns

S_OK - Success

E_OUTOFMEMORY - Unable to allocate memory for the string.

PutTime method

Stores a VBFILETIME value in the MxValue. The MxValue's data type will be MxTime.

Class

IMxValue

Syntax

[C#]

```
void PutTime(
```

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

```
   ref VBFILETIME pNewVal
);
```

[Visual C++]

```
HRESULT PutTime(
   [in] VBFILETIME *pNewVal
);
```

Parameters

pNewVal

The value that will be stored in the MxValue object.

Returns

S_OK - Success

PutElapsedTime method

Stores a VB_LARGE_INTEGER value in the MxValue. The MxValue's data type will be MxElapsedTime.

Class

IMxValue

Syntax

[C#]

```
void PutElapsedTime(
   ref VB_LARGE_INTEGER pNewVal
);
```

[Visual C++]

```
HRESULT PutElapsedTime(
   [in] VB_LARGE_INTEGER *pNewVal
);
```

Parameters

pNewVal

The value that will be stored in the MxValue object.

Returns

S_OK - Success

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

PutMxReference method

Stores an MxReference in the MxValue. The MxValue's data type will be MxReferenceType.

Class

IMxValue

Syntax

[C#]

```
void PutMxReference(
   IMxReference newVal
);
```

[Visual C++]

```
HRESULT PutMxReference(
   [in] IMxReference *newVal
);
```

Parameters

newVal

The MxReference that will be stored in the MxValue object.

Returns

S_OK - Success

PutMxStatus method

Stores an MxStatus in the MxValue. The MxValue's data type will be MxStatusType.

Class

IMxValue

Syntax

[C#]

```
void PutMxStatus(
   ref MxStatus newVal
);
```

[Visual C++]

```
HRESULT PutMxStatus(
   [in] const MxStatus *newVal
);
```

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Parameters

newVal

The MxStatus that will be stored in the MxValue object.

Returns

S_OK - Success

PutMxDataType method

Stores an MxDataType in the MxValue. The MxValue's data type will be MxDataTypeEnum.

Class

IMxValue

Syntax

[C#]

```
void PutMxDataType(
   MxDataType newVal
);
```

[Visual C++]

```
HRESULT PutMxDataType(
   [in] MxDataType newVal
);
```

Parameters

newVal

The MxDataType that will be stored in the MxValue object.

Returns

S_OK - Success

PutMxSecurityClassification method

Stores an Mx SecurityClassification in the MxValue. The MxValue's data type will be MxSecurityClassificationEnum.

Class

IMxValue

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Syntax

[C#]

```
void PutMxSecurityClassification(
   MxSecurityClassification newVal
);
```

[Visual C++]

```
HRESULT PutMxSecurityClassification(
   [in] MxSecurityClassification newVal
);
```

Parameters

newVal

The MxSecurityClassification that will be stored in the MxValue object.

Returns

S_OK - Success

PutMxDataQuality method

Stores an MxData Quality in the MxValue. The MxValue's data type will be MxDataQualityEnum.

Class

IMxValue

Syntax

[C#]

```
void PutMxDataQuality(
   ref short newVal
);
```

[Visual C++]

```
HRESULT PutMxDataQuality(
   [in] const MxDataQuality *newVal
);
```

Parameters

newVal

The MxDataQuality that will be stored in the MxValue object.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Returns

S_OK - Success

PutCustomStruct method

Stores a struct in the MxValue. The MxValue's data type will be MxCustomStruct.

Class

IMxValue

Syntax

[C#]

```
void PutCustomStruct(
   int guid,
   int structSize,
   ref byte pStruct
);
```

[Visual C++]

```
HRESULT PutCustomStruct(
   [in] long guid,
   [in] long structSize,
   [in, size_is(structSize)] const unsigned char *pStruct
);
```

Parameters

guid

A quasi-GUID that uniquely identifies this struct. The BRO does not validate the GUID.

structSize

Size of the struct.

pStruct

Pointer to the struct. MxValue will create its own copy of the struct.

Returns

S_OK - Success

PutCustomEnum method

Stores an MxCustomEnum in the MxValue. The MxValue's data type will be MxCustomEnum.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Class

IMxValue

Syntax

[C#]

```
void PutCustomEnum(
   string value,
   short ordinal,
   short primitiveId,
   short attributeId
);
```

[Visual C++]

```
HRESULT PutCustomEnum(
   [in] LPCWSTR value,
   [in] short ordinal,
   [in] MxPrimitiveId primitiveId,
   [in] MxAttributeId attributeId
);
```

Parameters

value,

The enumeration's string value. This parameter is used by Message Exchange clients. BRO can pass in 0 for internal clients.

ordinal,

The enumeration's ordinal value. This parameter is used by primitives. Other clients can pass in 0.

primitiveId,

The primitive corresponding to the attribute which holds an array of all the possible enumeration string values, or just pass in 0. This parameter is ignored by the BRO on external sets. This parameter is used by the BRO to initialize an MxValue.

attributeId

The attribute holding an array of all the possible enumeration string values, or just pass in 0. This parameter is ignored by the BRO on external sets. This parameter is used by the BRO to initialize an MxValue.

Returns

S_OK - Success

GetDataType method

Returns the data type of the value stored in the MxValue.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Class

IMxValue

Syntax

[C#]

```
MxDataType GetDataType();
```

[Visual C++]

```
HRESULT GetDataType(
   [out, retval] MxDataType *pVal
);
```

Parameters

pVal

Receives the data type.

Returns

S_OK - Success

E_POINTER - An invalid pointer was passed in.

GetBoolean method

Returns the value stored in the MxValue as a VARIANT_BOOL.

Class

IMxValue

Syntax

[C#]

```
bool GetBoolean();
```

[Visual C++]

```
HRESULT GetBoolean(
   [out, retval] VARIANT_BOOL *pVal
);
```

Parameters

pVal

Receives the value.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Returns

S_OK - Success

MX_E_ConversionNotSupported - The stored value is a type that cannot be converted to VARIANT_BOOL.

MX_E_UnableToConvertString - The string held in the MxValue is not "true" or "false".

E_POINTER - An invalid pointer was passed in.

GetInteger method

Returns the value stored in the MxValue as a long.

Class

IMxValue

Syntax

[C#]

```
int GetInteger();
```

[Visual C++]

```
HRESULT GetInteger(
   [out, retval] long *pVal
);
```

Parameters

pVal

Receives the value.

Returns

S_OK - Success

MX_E_ConversionNotSupported - The stored value is a type that cannot be converted to long.

MX_E_UnableToConvertString - The string held in the MxValue cannot be converted to a long.

MX_E_Overflow - Conversion of the stored value results in overflow.

E_POINTER - An invalid pointer was passed in.

GetFloat method

Returns the value stored in the MxValue as a float.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Class

IMxValue

Syntax

[C#]

```
float GetFloat();
```

[Visual C++]

```
HRESULT GetFloat(
   [out, retval] float *pVal
);
```

Parameters

pVal

Receives the value.

Returns

S_OK - Success

MX_E_ConversionNotSupported - The stored value is a type that cannot be converted to float.

MX_E_UnableToConvertString - The string held in the MxValue cannot be converted to a float.

MX_E_Overflow - Conversion of the stored value results in overflow.

E_POINTER - An invalid pointer was passed in.

GetDouble method

Returns the value stored in the MxValue as a double.

Class

IMxValue

Syntax

[C#]

```
double GetDouble();
```

[Visual C++]

```
HRESULT GetDouble(
   [out, retval] double *pVal
);
```

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Parameters

pVal

Receives the value.

Returns

S_OK - Success

MX_E_ConversionNotSupported - The stored value is a type that cannot be converted to double.

MX_E_UnableToConvertString - The string held in the MxValue cannot be converted to a double.

E_POINTER - An invalid pointer was passed in.

GetString method

Returns the value stored in the MxValue as a string.

Class

IMxValue

Syntax

[C#]

```
string GetString();
```

[Visual C++]

```
HRESULT GetString(
   [out, retval] BSTR *pVal
);
```

Parameters

pVal

Receives the value. Use SysFreeString to free the BSTR when finished with it.

If the value stored in the MxValue is not of type MxString, a string representation of the value is returned as follows:

- MxNoData - "No Data"
- MxBoolean - "true" or "false"
- MxInteger - "1234"
- MxFloat - "3.5"
- MxDouble - "3.5"
- MxString - "Hello World"
- MxTime - "Sun May 01 20:27:01 1994"

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

- MxElapsedTime - "89384" (elapsed time in 100 nanosecond increments.)
- MxReferenceType - "Valve1.pv"
- MxStatusType - "Error detected by Requesting NMX: Request timed out."
- MxDataTypeEnum - "MxInteger"
- MxSecurityClassificationEnum - "MxSecurityFreeAccess"
- MxDataQualityType - "Uncertain"
- MxCustomEnum - "Open"
- MxCustomStruct - "478602" (guid for the struct)

Returns

S_OK - Success

E_OUTOFMEMORY - Unable to allocate memory for the string.

E_POINTER - An invalid pointer was passed in.

GetTime method

Returns the value stored in the MxValue as a VBFILETIME.

Class

IMxValue

Syntax

[C#]

```
void GetTime(
   out VBFILETIME pVal
);
```

[Visual C++]

```
HRESULT GetTime(
   [out] VBFILETIME *pVal
);
```

Parameters

pVal

Receives the value.

Returns

S_OK - Success

MX_E_ConversionNotSupported - The stored value is a type that cannot be converted to VBFILETIME.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

MX_E_UnableToConvertString - The string held in the MxValue cannot be converted to a VBFILETIME.

E_POINTER - An invalid pointer was passed in.

GetElapsedTime method

Returns the value stored in the MxValue as a VB_LARGE_INTEGER.

Class

IMxValue

Syntax

[C#]

```
VB_LARGE_INTEGER GetElapsedTime();
```

[Visual C++]

```
HRESULT GetElapsedTime(
   [out, retval] VB_LARGE_INTEGER *pVal
);
```

Parameters

pVal

Receives the value.

Returns

S_OK - Success.

MX_E_ConversionNotSupported - The stored value is not of type MxElapsedTime.

E_POINTER - An invalid pointer was passed in.

GetMxReference method

Returns the value stored in the MxValue as an IMxReference *.

Class

IMxValue

Syntax

[C#]

```
IMxReference GetMxReference();
```

[Visual C++]

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

```
HRESULT GetMxReference(
   [out, retval] IMxReference **pVal
);
```

Parameters

pVal

Receives the value.

Returns

S_OK - Success.

MX_E_ConversionNotSupported - The stored value is not an MxReference or MxString.

E_POINTER - An invalid pointer was passed in.

GetMxStatus method

Returns the MxStatus stored in the MxValue.

Class

IMxValue

Syntax

[C#]

```
MxStatus GetMxStatus();
```

[Visual C++]

```
HRESULT GetMxStatus(
   [out, retval] MxStatus *pVal
);
```

Parameters

pVal

Receives the value.

Returns

S_OK - Success.

MX_E_ConversionNotSupported - The stored value is not an MxStatus.

E_POINTER - An invalid pointer was passed in.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

GetMxDataType method

Returns the value stored in the MxValue as an MxDataType.

Class

IMxValue

Syntax

[C#]

```
MxDataType GetMxDataType();
```

[Visual C++]

```
HRESULT GetMxDataType(
   [out, retval] MxDataType *pVal
);
```

Parameters

pVal

Receives the value.

Returns

S_OK - Success

MX_E_ConversionNotSupported - The stored value is a type that cannot be converted to MxDataType.

MX_E_UnableToConvertString - The string held in the MxValue cannot be converted to an MxDataType.

MX_E_Overflow - Conversion of the stored value results in overflow.

E_POINTER - An invalid pointer was passed in.

GetMxSecurityClassification method

Returns the value stored in the MxValue as an MxSecurityClassification.

Class

IMxValue

Syntax

[C#]

```
MxSecurityClassification GetMxSecurityClassification();
```

[Visual C++]

```
HRESULT GetMxSecurityClassification(
```

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

```
   [out, retval] MxSecurityClassification *pVal
);
```

Parameters

pVal

Receives the value.

Returns

S_OK - Success

MX_E_ConversionNotSupported - The stored value is a type that cannot be converted to MxSecurityClassification.

MX_E_UnableToConvertString - The string held in the MxValue cannot be converted to an MxSecurityClassification.

MX_E_Overflow - Conversion of the stored value results in overflow.

E_POINTER - An invalid pointer was passed in.

GetMxDataQuality method

Returns the value stored in the MxValue as an MxDataQuality.

Class

IMxValue

Syntax

[C#]

```
short GetMxDataQuality();
```

[Visual C++]

```
HRESULT GetMxDataQuality(
   [out, retval] MxDataQuality *pVal
);
```

Parameters

pVal

Receives the value.

Returns

S_OK - Success.

MX_E_ConversionNotSupported - The stored value is a type that cannot be converted to

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

MxMxDataQualityataType.

MX_E_UnableToConvertString - The string held in the MxValue cannot be converted to an MxDataQuality.

MX_E_Overflow - Conversion of the stored value results in overflow.

E_POINTER - An invalid pointer was passed in.

GetCustomStruct method

Returns the qualified struct stored in the MxValue.

Class

IMxValue

Syntax

[C#]

```
void GetCustomStruct(
   out int pGuid,
   out int pStructSize,
   System.IntPtr pStruct
);
```

[Visual C++]

```
HRESULT GetCustomStruct(
   [out] long *pGuid,
   [out] long *pStructSize,
   [out, size_is(,*pStructSize)] unsigned char **pStruct
);
```

Parameters

pGuid,

Receives the quasi-GUID that uniquely identifies this struct. The BRO does not validate the GUID.

pStructSize,

Receives the size of the struct held by the MxValue.

pStruct

Address of the pointer which will receive the struct. The caller must call CoTaskMemFree on pStruct when it is finished using it.

Returns

S_OK - Success.

MX_E_ConversionNotSupported - The stored value is not a qualified struct.

E_POINTER - An invalid pointer was passed in.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

GetCustomEnum method

Returns the value stored in the MxValue as a qualified enum.

Class

IMxValue

Syntax

[C#]

```
void GetCustomEnum(
   out string pValue,
   out short pOrdinal,
   out short pPrimitiveId,
   out short pAttributeId
);
```

[Visual C++]

```
HRESULT GetCustomEnum(
   [out] BSTR *pValue,
   [out] short *pOrdinal,
   [out] MxPrimitiveId *pPrimitiveId,
   [out] MxAttributeId *pAttributeId
);
```

Parameters

pValue,

Receives the enumeration's string value.

pOrdinal,

Receives the enumeration's ordinal value.

pPrimitiveId,

Receives primitive ID of the primitive that contains the attribute holding an array of all the possible enumeration string values.

pAttributeId

Receives attribute ID of the attribute holding an array of all the possible enumeration string values.

Returns

S_OK - Success.

MX_E_ConversionNotSupported - The stored value is a type that cannot be converted to a qualified enum.

E_POINTER - An invalid pointer was passed in.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

GetDimensionCount method

Indicates whether the MxValue holds an array. MxValue holds an array if PutElement has been called on it and there have been no subsequent calls to Empty or PutXxx (where Xxx is an Integer, Boolean or any of the other data type).

Class

IMxValue

Syntax

[C#]

```
void GetDimensionCount(
   out short nDimensions
);
```

[Visual C++]

```
HRESULT GetDimensionCount(
   [out] short *nDimensions
);
```

Parameters

nDimensions

Returns 1 if the MxValue is an array or 0 if MxValue does not hold an array.

PutElement method

Puts an element into an array held by an MxValue. Call this method consecutively to populate an MxValue with an array of one dimension. The first call on this method will establish the MxDataType for the array. Subsequent calls to this method must supply MxValues of the same MxDataType.

Note: An array's dimension cannot be reduced. Therefore it is not a good idea to cache an MxValue. The array should be destroyed and recreated if its dimension needs to be reduced. Array elements must be written sequentially when initializing array values.

Syntax

[C#]

```
void PutElement(
   int index,
   MxValue pMxValue
);
```

[Visual C++]

```
HRESULT PutElement(
   [in] long index,
   [in] IMxValue *pMxValue
```

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

```
);
```

Parameters

index

Index into the array. Must be between 1 and 2147483647 inclusive.

pMxValue

The value to be put as an array element.

Returns

MX_E_InvalidIndex - Index is out of range.

MX_E_IndexOutOfOrder - Index is not sequential.

MX_E_WrongDataType - A prior call to PutElement supplied an MxValue of a different data type.

GetElement method

Gets an element from an array held in an MxValue.

Class

IMxValue

Syntax

[C#]

```
void GetElement(
   int index1,
   MxValue pMxValue
);
```

[Visual C++]

```
HRESULT GetElement(
   [in] long index1,
   [in] IMxValue *pMxValue
);
```

Parameters

index1

Index into the one-dimensional array.

pMxValue

Receives the specified element's value.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Returns

MX_E_InvalidIndex - Index is out of range.

MX_E_WrongNumberOfDimensions - This is not a one-dimensional array.

GetDimensionSize method

Gets the size of the array held by the MxValue. MxValue determines the size of the array based on the max index value of prior calls to PutElement.

Class

IMxValue

Syntax

[C#]

```
void GetDimensionSize(
   out int pSize
);
```

[Visual C++]

```
HRESULT GetDimensionSize(
   [out] long *pSize
);
```

Parameters

pSize

Receives the specified dimension's size.

PutCustomStructVB method

Stores a qualified struct in the MxValue for a VB client.

Class

IMxValue

Syntax

[C#]

```
void PutCustomStructVB(
   int guid,
   ref byte[] pStruct
);
```

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

[Visual C++]

```
HRESULT PutCustomStructVB(
   [in] long guid,
   [in,out] SAFEARRAY(BYTE) *pStruct
);
```

GetCustomStructVB method

Returns the qualified struct stored in the MxValue for a VB client.

Class

IMxValue

Syntax

[C#]

```
void GetCustomStructVB(
   out int pGuid,
   ref byte[] pStruct
);
```

[Visual C++]

```
HRESULT GetCustomStructVB(
   [out] long *pGuid,
   [in,out] SAFEARRAY(BYTE) *pStruct
);
```

PutInternationalStrings method

Initializes an MxValue with a set of internationalized strings.

Class

IMxValue

Syntax

[C#]

```
void PutInternationalStrings(
   int count,
   ref InternationalizedString strings
);
```

[Visual C++]

```
HRESULT PutInternationalStrings(
   [in] long count,
   [in, size_is(count)] struct InternationalizedString strings[]
);
```

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Parameters

count

The number of internationalized strings the MxValue is being initialized with.

strings[]

The internationalized strings the MxValue is being initialized with.

Returns

S_OK - operation succeeded.

E_FAIL - operation failed.

PutInternationalStringsVB method

Initializes an MxValue with a set of internationalized strings for a VB client.

Class

IMxValue

Syntax

[C#]

```
void PutInternationalStringsVB(
   ref InternationalizedString[] ppsa
);
```

[Visual C++]

```
HRESULT PutInternationalStringsVB(
   [in] SAFEARRAY(struct InternationalizedString) *ppsa
);
```

Returns

S_OK - operation succeeded.

E_FAIL - operation failed.

GetInternationalStrings method

Returns the internationalized strings stored by the MxValue.

Class

IMxValue

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Syntax

[C#]

```
void GetInternationalStrings(
   out int count,
   System.IntPtr locals
);
```

[Visual C++]

```
HRESULT GetInternationalStrings(
   [out] long *count,
   [out, size_is(,*count)] struct InternationalizedString **locals
);
```

Parameters

count

The number of internationalized strings that are stored in the MxValue.

locals

The internationalized strings that are stored in the MxValue.

Returns

S_OK - operation succeeded.

E_FAIL - operation failed.

MX_E_WrongDataType - the stored value isn't an internationalized string.

GetInternationalStringsVB method

Returns the internationalized strings stored by the MxValue for a VB client.

Class

IMxValue

Syntax

[C#]

```
void GetInternationalStringsVB(
   ref InternationalizedString[] ppsa
);
```

[Visual C++]

```
HRESULT GetInternationalStringsVB(
   [in,out] SAFEARRAY(struct InternationalizedString) *ppsa
);
```

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Returns

S_OK - operation succeeded.

E_FAIL - operation failed.

MX_E_WrongDataType- the stored value isn't an internationalized string.

GetInternationalString method

Returns the internationalized string associated with a particular locale. The internationalized string for the locale passed in must have been added to the MxValue via calls to PutInternationalString, PutInternationalStringsVB, or PutInternationalStrings prior to calling this method.

Class

IMxValue

Syntax

[C#]

```
string GetInternationalString(
   int locale
);
```

[Visual C++]

```
HRESULT GetInternationalString(
   [in] long locale,
   [out, retval] BSTR *internationalizedString
);
```

Parameters

locale,

The locale of the internationalized string that is to be retrieved from the MxValue.

internationalizedString

The internationalized string that corresponds to the locale passed in.

Returns

S_OK - operation succeeded.

E_FAIL - operation failed.

MX_E_WrongDataType - the stored value isn't an internationalized string.

MX_E_LocaleNotSupported - the internationalized text doesn't exist for the locale passed in.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

PutInternationalString method

Adds the internationalized string of a specific locale to the existing internationalized strings. Held by the MxValue. If the locale is already supported by the internationalized string then the string passed in overwrites the existing string for that locale.

Class

IMxValue

Syntax

[C#]

```
void PutInternationalString(
   int locale,
   string InternationalizedString
);
```

[Visual C++]

```
HRESULT PutInternationalString(
   [in] long locale,
   [in] BSTR internationalizedString
);
```

Parameters

locale,

The locale of the internationalized string that is being set into the MxValue.

internationalizedString,

The internationalized string that is to be set into the MxValue.

Returns

S_OK - operation succeeded.

E_FAIL - operation failed.

IPermission

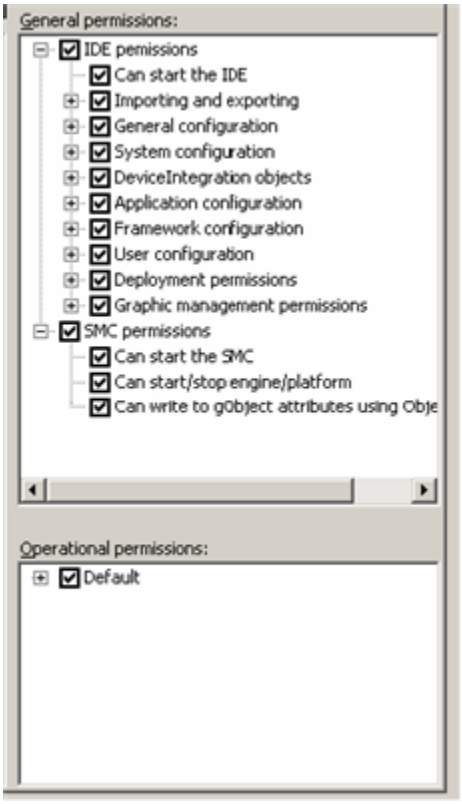
IPermission provides read-only access to the properties of a single permission, including name, parent name, whether the permission is configured, if it has any children permissions, collection children permissions, whether it is a security group (root nodes in operation permission are). If the permission is a security group, IPermission can access the security group object.

Each role has two collections of permissions, a collection of general permissions and a collection of operational permissions, from which the client program can drill down to each sub-permission.

For example, for a role configured with the options shown in the sample **Options** dialog box, IPermission returns the following:

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Operation	What it returns
PermissionName	General Permission
PermissionParentName	Empty string
IsConfigured	Returns VARIANT_TRUE, or VARIANT_FALSE if any of its permission in the tree is configured
HasChildren	Returns VARIANT_TRUE
ChildPermissions	Returns the collection of IPermissions, which in this case is IDE Permissions and SMC Permissions
IsASecurityGroup	Returns VARIANT_FALSE. For "Operational Permissions" children, this property is VARIANT_TRUE.
SecurityGroup	Returns the ISecurityGroup object interface through which the user can access security group related properties. This property is valid for Operational Permission children. In other cases, this returns a NULL pointer.



IPermission Class members

This class has the following members.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Operations

PermissionName property

Returns the permission name.

PermissionParentName property

Returns the name of the permission's parent permission.

IsConfigured property

Returns whether the permission is configured; that is, whether the permission is active.

HasChildren property

Returns true if the permission has child permissions.

ChildPermissions property

Returns the collection of child permissions.

IsSecurityGroup property

Returns true if the permission is a security group.

SecurityGroup property

Retrieves the security group object.

PermissionName property

Returns the name of the permission.

Class

IPermission

Syntax

[C#]

```
string PermissionName { get; };
```

[Visual C++]

```
HRESULT PermissionName(
   [out,retval]BSTR *permissionName
);
```

PermissionParentName property

Returns the name of the permission's parent, if any. (For top-level permissions, this is an empty string.)

Class

IPermission

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Syntax

[C#]

```
string PermissionParentName { get; };
```

[Visual C++]

```
HRESULT PermissionParentName(
   [out,retval]BSTR *permissionParentName
);
```

IsConfigured property

Returns true if the permission is configured; that is, if the permission is active.

Class

IPermission

Syntax

[C#]

```
bool IsConfigured { get; };
```

[Visual C++]

```
HRESULT IsConfigured(
   [out,retval]VARIANT_BOOL *isConfigured
);
```

HasChildren property

Returns true if the permission has child permissions.

Class

IPermission

Syntax

[C#]

```
bool HasChildren { get; };
```

[Visual C++]

```
HRESULT HasChildren(
   [out,retval]VARIANT_BOOL *hasChildren
);
```

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

ChildPermissions property

Returns the collection of child permissions.

Class

IPermission

Syntax

[C#]

```
IPermissions ChildPermissions { get; };
```

[Visual C++]

```
HRESULT ChildPermissions(
   [out,retval]IPermissions **childPermissions
);
```

IsSecurityGroup property

Returns true if the permission is a security group.

Class

IPermission

Syntax

[C#]

```
bool IsASecurityGroup { get; };
```

[Visual C++]

```
HRESULT IsASecurityGroup(
   [out,retval]VARIANT_BOOL *isSecurityGroup
);
```

SecurityGroup property

Retrieves the security group object.

Class

IPermission

Syntax

[C#]

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

```
ISecurityGroup SecurityGroup { get; };
```

[Visual C++]

```
HRESULT SecurityGroup(
   [out,retval]ISecurityGroup **secGroup
);
```

IPermissions

Represents a collection of security permissions.

IPermissions class members

This class has the following members.

Operations

Item Property

Returns a permission.

Count Property

Returns the number of permissions in the collection.

Item property

Returns a permission.

Class

IPermissions

Syntax

[C#]

```
IPermission this[object galaxyIdentifier] { get; };
```

[Visual C++]

```
HRESULT Item(
   [in] VARIANT galaxyIdentifier,
   [out, retval] IPermission**
);
```

Parameters

galaxyIdentifier

The index of a permission. This is a numeric value from 1 to Count, or the name of the permission.

IPermission

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

The returned permission.

Count property

Returns the number of permissions in the collection.

Class

IPermissions

Syntax

[C#]

```
int count { get; };
```

[Visual C++]

```
HRESULT Count(
   [out, retval] LONG*
);
```

IScriptLibrary

Represents a script library.

IScriptLibrary class members

This class has the following members.

Operations

CommandResult Property

Returns the CommandResult object, which has the last method call's result.

Export method

Export this script library.

Name Property

Returns the name of the script library.

CommandResult property

Returns the CommandResult object, which has the last method call's result.

Class

IScriptLibrary

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Syntax

[C#]

```
ICommandResult CommandResult { get; };
```

[Visual C++]

```
HRESULT CommandResult(
   [out, retval] ICommandResult** CommandResult
);
```

Parameters

CommandResult

Returns the CommandResult object.

Export method

Export this script library.

Class

IScriptLibrary

Syntax

[C#]

```
void Export(
   string path
);
```

[Visual C++]

```
HRESULT Export(
   [in] BSTR path
);
```

Parameters

path

Path to copy the .aaslib file.

Name property

Returns the name of the script library.

Class

IScriptLibrary

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Syntax

[C#]

```
string Name { get; };
```

[Visual C++]

```
HRESULT Name(
   [out, retval] BSTR* theScriptLibrary
);
```

Parameters

theScriptLibrary

The script library name.

IScriptLibraries

Represents a collection of script libraries.

IScriptLibraries class members

This class has the following members.

Operations

CommandResult Property

Returns the CommandResult object, which has the last method call's result.

Item Property

Return a script library.

Count Property

Returns the number of script libraries in the collection.

Add Method

Add a script library to the Galaxy.

CommandResult property

Returns the CommandResult object, which has the last method call's result.

Class

IScriptLibraries

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Syntax

[C#]

```
ICommandResult CommandResult { get; };
```

[Visual C++]

```
HRESULT CommandResult(
   [out, retval] ICommandResult** CommandResult
);
```

Parameters

CommandResult

Returns the CommandResult object.

Item property

Returns a script library.

Class

IScriptLibraries

Syntax

[C#]

```
IScriptLibrary this[object scriptLibraryIdentifier] { get; };
```

[Visual C++]

```
HRESULT Item(
   [in] VARIANT scriptLibraryIdentifier,
   [out, retval] IScriptLibrary**
);
```

Parameters

scriptLibraryIdentifier

The index of a script library. This is a numeric value from 1 to count, or the name of the script library.

IScriptLibrary

The returned script library object.

Count property

Returns the number of script library objects in the collection.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Class

IScriptLibraries

Syntax

[C#]

```
int count { get; };
```

[Visual C++]

```
HRESULT Count(
   [out, retval] LONG*
);
```

Add method

Add a new script library to the Galaxy.

Class

IScriptLibraries

Syntax

[C#]

```
void Add(
   string scriptLibraryPath
);
```

[Visual C++]

```
HRESULT Add(
   [in] BSTR scriptLibraryPath
);
```

Parameters

scriptLibraryPath

The name of the new script library.

ISecurityGroup

Accesses a security group object and its properties, including the security group name and the list of gObjects.

ISecurityGroup class members

This class has the following members.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Operations

GroupName property

Returns a security group name.

gObjects property

Returns the list of gObjects in this security group.

GroupName property

Returns the security group name.

Class

ISecurityGroup

Syntax

[C#]

```
string GroupName { get; };
```

[Visual C++]

```
HRESULT GroupName(
   [out, retval] BSTR *groupName
);
```

Parameters

groupName

The returned security group name.

gObjects property

Returns the list of gObjects in this security group.

Class

ISecurityGroup

Syntax

[C#]

```
IgObjects gObjects { get; };
```

[Visual C++]

```
HRESULT gObjects(
   [out, retval] IgObjects** *gObjects
```

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

```
);
```

Parameters

gObjects

The list of gObjects in this security group.

ISecurityGroups

Represents a collection of security groups.

ISecurityGroups class members

This class has the following members.

Operations

Item Property

Returns a security group.

Count Property

Returns the number of security groups in the collection.

Item property

Return a security group.

Class

ISecurityGroups

Syntax

[C#]

```
ISecurityGroup this[object galaxyIdentifier] { get; };
```

[Visual C++]

```
HRESULT Item(
   [in] VARIANT galaxyIdentifier,
   [out, retval] securityGroup**
);
```

Parameters

galaxyIdentifier

The index of a galaxy. This is a numeric value from 1 to count, or the name of the security group.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

securityGroup

The returned security group.

Count property

Returns the number of security groups in the collection.

Class

ISecurityGroups

Syntax

[C#]

```
int count { get; };
```

[Visual C++]

```
HRESULT Count(
   [out, retval] LONG* count
);
```

Parameters

count

Returns the number of security groups in the collection.

ISettings

Represents a settings object. Client configures the data for this settings object using the attributes of the instance property. After completed, the client calls Close to save and check-in the changes, or Cancel to undo the changes. While the settings object is being configured, no other object in the GR may be checked-out.

ISettings class members

This class has the following members.

Operations

CommandResult Property

Returns the CommandResult object, which has the last method call's result.

Cancel method

Cancels the configuration of this object.

Instance property

Gets the instance for configuring this object.

Close method

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Saves and checks in the settings object.

CommandResult property

Returns the CommandResult object, which has the last method call's result.

Class

ISettings

Syntax

[C#]

```
ICommandResult CommandResult { get; };
```

[Visual C++]

```
HRESULT CommandResult(
   [out, retval] ICommandResult** CommandResult
);
```

Parameters

CommandResult

Returns the CommandResult object.

Cancel method

Cancels the configuration of this object.

Class

ISettings

Syntax

[C#]

```
void Cancel();
```

[Visual C++]

```
HRESULT Cancel();
```

Instance property

Get the instance for configuring this object.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Class

ISettings

Syntax

[C#]

```
IInstance Instance { get; };
```

[Visual C++]

```
HRESULT Instance(
   [out, retval] IInstance** anInstance
);
```

Parameters

anInstance

The instance to be used to configure the object.

Close method

Saves and checks in the settings object.

Class

ISettings

Syntax

[C#]

```
void Close();
```

[Visual C++]

```
HRESULT Close();
```

ITemplate

Represents a Template object.

ITemplate class members

This class has the following members.

Operations

AddExtensionPrimitive method

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Adds an extension primitive.

AddUDA method

Adds a UDA.

Area property

Returns or sets the area for this object.

Attributes property

Returns the collection of the object's attributes.

BasedOn property

Returns the name of the object's base template.

Category property

Returns the category of this object.

CategoryGUID property

Returns the category GUID for this object.

CheckIn method

Checks in the object.

CheckedOutBy property

Returns the user who has checked out this object.

CheckOut Method

Checks out the object.

CheckoutStatus property

Returns the checkout status of this object.

CommandResult Property

Returns the CommandResult object, which has the last method call results.

CommandResults property

Returns the CommandResults object, which has the last method call's result. Used for CreateInstances.

ConfigurableAttributes property

Returns a collection of the configurable attributes for this object.

ConfigVersion property

Returns the configuration version number.

ContainedName property

Returns or sets the contained name for this object.

Container property

Returns or sets the container for this object.

CreateInstance method

Create a new instance from this template.

CreateInstances method

Creates a collection of new instances from a template.

CreateTemplate method

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Derive a new template from this template.

DeleteExtensionPrimitive method

Delete the extension primitive.

DeleteTemplate method

Deletes the template.

DeleteUDA method

Deletes a user defined attribute.

DerivedFrom property

Returns the name of the template this object was derived from.

EditStatus property

Returns the status of this object.

Errors property

Returns a list of errors generated the last time this object was validated and saved.

GetExtendedAttributes method

Returns a list of attributes for this object.

GetObjectHelpURL method

Returns the URL where the object's help is stored in the Galaxy repository.

HierarchicalName property

Returns the hierarchical name of this object.

Host property

Returns the name of this object's host.

RenameExtensionPrimitive method

Change the name of an extension primitive..

RenameUDA method

Changes the name of a user defined attribute.

Save method

Saves an object.

Tagname property

Returns the tagname of this object.

Toolset property

Returns the Toolset that this template belongs to.

UndoCheckOut method

Changes the object's checked out status without saving changes.

Unload method

Unloads the object's cache. The object remains usable.

UpdateUDA method

Change a user defined attribute.

ValidationStatus property

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Returns the validation status information from the last time the object was validated and saved.

Warnings property

Returns a list of warnings from the last time the object was validated and saved.

AddExtensionPrimitive method

Adds an extension primitive to this object.

Class

ITemplate

Syntax

[C#]

```
void AddExtensionPrimitive(
   string ExtensionType,
   string ExtensionPrimitiveName,
   bool IsObjectExtension
);
```

[Visual C++]

```
HRESULT AddExtensionPrimitive(
   // Type of extension
   [in] BSTR ExtensionType,
   // Name of extension primitive to be added -
   // For attribute extension primitive - Fully
   // qualified name of attribute being extended
   // For object extension primitive - New name
   // specified by user for object extension
   // primitive
   [in] BSTR ExtensionPrimitiveName,
   // If wszName is the new name specified by user,
   // this field will be set to true. For example,
   //when add script pass true
   [in, optional, defaultvalue( -1 )]VARIANT_BOOL IsObjectExtension
);
```

AddUDA method

Adds a UDA.

Class

ITemplate

Syntax

[C#]

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

```
void AddUDA(
   string UDAName,
   MxDataType DataType,
   MxAttributeCategory category,
   MxSecurityClassification Security,
   bool IsArray,
   object ArrayElementCount
);
```

[Visual C++]

```
// Add a UDA
//
// Returns:
// S_OK - Successfully add a UDA
HRESULT AddUDA(
   // Same XML format. But only one is allowed.
   [in] BSTR UDAName,
   [in] enum MxDataType Datatype,
   [in] enum MxAttributeCategory Category,
   [in] enum MxSecurityClassification Security,
   [in] VARIANT_BOOL IsArray,
   [in] VARIANT ArrayElementCount
);
```

Area property

Get or set the area of this object. An AppObject area must be set using the SetContainer or SetHost methods. The object does not need to be checked out, and the operation is immediate.

Class

ITemplate

Syntax

[C#]

```
string Area { set; get; };
```

[Visual C++]

```
// Returns name of the Area object, or blank if the
// object doesn't have an Area.
[propget]
HRESULT Area(
   [out, retval] BSTR* theArea
);
// Set the Area of this object.
[propput]
HRESULT Area(
   // The name of the Area, or blank to unassign
   // from existing Area.
   [in] BSTR NewArea
);
```

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Attributes property

Returns the collection of attributes of this object.

Class

ITemplate

Syntax

[C#]

```
IAttributes Attributes { get; };
```

[Visual C++]

```
// The collection of Attributes of this object.
// Configuring attributes requires that the object
// be checked out.
HRESULT Attributes(
   // The collection of Attributes of this object.
   [out, retval] IAttributes** theAttributes
);
```

BasedOn property

Returns the name of the base template for this object.

Class

ITemplate

Syntax

[C#]

```
string basedOn { get; };
```

[Visual C++]

```
// The name of the base template (root ancestor) of
// this object.
HRESULT BasedOn(
   // The name of the base template (root ancestor)
   // of this object.
   [out, retval] BSTR* theBaseTemplate
);
```

Category property

Returns the category of this object.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Class

ITemplate

Syntax

[C#]

```
ECATEGORY category { get; };
```

[Visual C++]

```
// The Category of the object.
HRESULT Category(
   // The Category of the object.
   [out, retval] enum ECATEGORY* theCategory
);
```

CategoryGUID property

Gets the category GUID of this object.

Class

ITemplate

Syntax

[C#]

```
VBGUID CategoryGUID { get; };
```

[Visual C++]

```
// The Category GUID of the object.
HRESULT CategoryGUID(
   // The Category of the object.
   [out, retval] VBGUID *categoryGUID
);
```

CheckIn method

Check in this object.

Class

ITemplate

Syntax

[C#]

```
void CheckIn(
```

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

```
   string CheckInComment
);
```

[Visual C++]

```
// Checks in an object.
// Called on a checked-out object after it is
// configured and saved.
HRESULT CheckIn(
   // Check in an object.
   [in, optional] BSTR CheckInComment
);
```

CheckOut method

Checks out an object.

Class

ITemplate

Syntax

[C#]

```
void CheckOut();
```

[Visual C++]

```
// Checks out an object.
HRESULT CheckOut();
```

CheckedOutBy property

Returns the name of the user who has checked out this object.

Class

ITemplate

Syntax

[C#]

```
string checkedOutBy { get; };
```

[Visual C++]

```
// The name of the user that this object is checked
// out by.
HRESULT CheckedOutBy(
   // The name of the user that this object is
   // checked out by.
   [out, retval] BSTR* CheckedOutByUserName
);
```

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

CheckoutStatus property

Returns the check out status of the object.

Class

ITemplate

Syntax

[C#]

```
ECheckoutStatus CheckoutStatus { get; };
```

[Visual C++]

```
// The check-out status of the object.
HRESULT CheckoutStatus(
   [out, retval] enum ECheckoutStatus* theCheckoutStatus
);
```

CommandResult property

Returns the CommandResult object, which has the results from the last method call.

Class

ITemplate

Syntax

[C#]

```
ICommandResult CommandResult { get; };
```

[Visual C++]

```
// Returns the CommandResult Object, which has last
// method call result
HRESULT CommandResult(
   // Returns the CommandResult
   [out, retval] ICommandResult** CommandResult
);
```

CommandResults property

Returns the CommandResults object, which has the last method call's result. Used for CreateInstances

Class

ITemplate

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Syntax

[C#]

```
ICommandResults CommandResults { get; };
```

[Visual C++]

```
HRESULT CommandResults(
   [out, retval] ICommandResults** CommandResults
);
```

Parameters

CommandResults

Returns the CommandResults object.

ConfigurableAttributes property

Returns a collection of this object's attributes that can be configured.

Class

ITemplate

Syntax

[C#]

```
IAttributes ConfigurableAttributes { get; };
```

[Visual C++]

```
// The collection of Attributes of this object that
// can be configured.
// Configuring attributes requires that the object
// be checked out.
HRESULT ConfigurableAttributes(
   // The collection of Attributes of this object
   // that can be configured.
   [out, retval] IAttributes** theConfigurableAttributes
);
```

ConfigVersion property

Returns this object's configuration version number. The configuration version is incremented each time the object is checked in.

Class

ITemplate

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Syntax

[C#]

```
int ConfigVersion { get; };
```

[Visual C++]

```
// The configuration version of the object.
HRESULT ConfigVersion(
   // The configuration version of the object.
   [out, retval] LONG* theConfigVersion
);
```

ContainedName property

Sets or returns this object's contained name. When setting the contained name, the object does not need to be checked out, and the operation is immediate.

Class

ITemplate

Syntax

[C#]

```
string ContainedName { set; get; };
```

[Visual C++]

```
//
// Returns Contained name of this object (eg. Inlet)
[propget]
HRESULT ContainedName(
   // The contained name of this object (eg. Inlet)
   [out, retval] BSTR* theContainedName
);
// Sets the contained name of this object (eg. Inlet)
// The object does not need to be checked out; the
// operation is immediate.
[propput]
HRESULT ContainedName(
   // The contained name of this object (eg. Inlet)
   [in] BSTR newContainedName
);
```

Container property

Returns or sets the container for this object. When setting the container, the object does not need to be checked out, and the change is immediate. For an ApplicationObject, the ApplicationObject's Area and Host area are also updated.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Class

ITemplate

Syntax

[C#]

```
string Container { set; get; };
```

[Visual C++]

```
// Returns name of the Container object, or blank if
//object is not contained.
[propget]
HRESULT Container(
   // The name of the Container object, or blank if
   // the object is not contained.
   [out, retval] BSTR* theContainer
);
// Set the Container for this object.
[propput]
HRESULT Container(
   // The name of the container, or blank to uncontain.
   [in] BSTR NewContainer
);
```

CreateInstance method

Create a new Instance from this template.

Class

ITemplate

Syntax

[C#]

```
IInstance CreateInstance(
   string Name,
   bool createContainedObjects
);
```

[Visual C++]

```
HRESULT CreateInstance(
   [in] BSTR Name,
   [in, optional, defaultvalue( 1 )] VARIANT_BOOL createContainedObjects,
   [out, retval] IInstance**
);
```

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Parameters

Name

The name of the newly created instance.

createContainedObjects

Set to True to create contained objects.

IInstance

The instance that was created.

CreateInstances method

Creates a collection of new instances from a template.

Class

ITemplate

Syntax

[C#]

```
IgObjects CreateInstances(
   ref string[] tagnames,
   bool createContainedObjects
);
```

[Visual C++]

```
HRESULT CreateInstances(
   [in, out] SAFEARRAY(BSTR)* tagnames,
   [in, optional, defaultvalue( 1 )] VARIANT_BOOL createContainedObjects,
   [out, retval] IgObjects**
);
```

Parameters

tagnames

An array of unique tagnames for the newly created instances.

createContainedObjects

Set to true to create contained objects.

IgObjects

The instances that were created.

CreateTemplate method

Derive an a new template from this template.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Class

ITemplate

Syntax

[C#]

```
ITemplate CreateTemplate(
   string Name,
   bool createContainedObjects
);
```

[Visual C++]

```
HRESULT CreateTemplate(
   [in] BSTR Name,
   [in, optional, defaultvalue( 1 )] VARIANT_BOOL createContainedObjects,
   [out, retval] ITemplate**
);
```

Parameters

Name

The name of the newly created template.

createContainedObjects

Set to true to create contained objects.

ITemplate

The template that was created.

DeleteExtensionPrimitive method

Deletes an extension.

Class

ITemplate

Syntax

[C#]

```
void DeleteExtensionPrimitive(
   string ExtensionType,
   string ExtensionPrimitiveName
);
```

[Visual C++]

```
// Delete an extension primitive
//
// Returns S_OK
```

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

```
HRESULT DeleteExtensionPrimitive(
   // Type of extension
   [in] BSTR ExtensionType,
   // Name of Extension primitive to be deleted.
   [in] BSTR ExtensionPrimitiveName
);
```

DeleteTemplate method

Deletes the template.

Class

ITemplate

Syntax

[C#]

```
void DeleteTemplate(
   EForceDeleteTemplateOption ForceDeleteTemplateOption
);
```

[Visual C++]

```
HRESULT DeleteTemplate(
   [in, optional, defaultvalue( dontForceTemplateDelete )] EForceDeleteTemplateOption
   ForceDeleteTemplateOption
);
```

Parameters

ForceDeleteTemplateOption

Indicates whether descendant instances should be undeployed prior to deletion.

DeleteUDA method

Deletes a UDA.

Class

ITemplate

Syntax

[C#]

```
void DeleteUDA(
   string UDAName
);
```

[Visual C++]

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

```
// Delete a UDA
//
// Returns:
// S_OK - Successfully delete a UDA
HRESULT DeleteUDA(
   // Delete an UDA by UDA name.
   [in] BSTR UDAName
);
```

DerivedFrom property

Returns the name of the object's template.

Class

ITemplate

Syntax

[C#]

```
string DerivedFrom { get; };
```

[Visual C++]

```
// Returns the name of the template that this object
// was created from.
[propget]
HRESULT DerivedFrom(
   // The name of the template that this object was
   // created from.
   [out, retval] BSTR* theParentTemplate
);
```

EditStatus property

Edit the status of the object.

Class

ITemplate

Syntax

[C#]

```
EEditStatus EditStatus { get; };
```

[Visual C++]

```
// The edit status of the object.
HRESULT EditStatus(
   // The edit status of the object.
   [out, retval] enum EEditStatus* theEditStatus
```

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

```
);
```

Errors property

Retrieves the list of errors generated when the object was last validated and saved.

Class

ITemplate

Syntax

[C#]

```
string[] Errors { get; };
```

[Visual C++]

```
// The list of errors that got generated the last
// time the object was validated and saved.
HRESULT Errors(
   // The list of errors that was generated the last
   // time the object was validated and saved.
   [out, retval] SAFEARRAY(BSTR)* theErrors
);
```

GetExtendedAttributes method

Get the extended attributes in the hierarchy.

Class

ITemplate

Syntax

[C#]

```
IAttributes GetExtendedAttributes(
   string AttributeName,
   int upto_level,
   MxAttributeCategory[] ReturnOnlyTheseCategories
);
```

[Visual C++]

```
//Get Extended Attributes in Hierarchy
HRESULT GetExtendedAttributes(
   // Attribute full name which is extended
   [in] BSTR AttributeName,
   //Up to how many level user needs extended
   //primitives
   //-1 will return all levels
   [in] int upto_level,
```

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

```
   //Return attributes which are only these
   //categories. If safearray is empty returns all
   //attributes.
   [in]SAFEARRAY(MxAttributeCategory) ReturnOnlyTheseCategories,
   //Return attributes
   [out,retval]IAttributes** theAttributes
);
```

GetObjectHelpURL method

Returns the URL where the object help is stored on the Galaxy repository.

Class

ITemplate

Syntax

[C#]

```
string GetObjectHelpURL();
```

[Visual C++]

```
// Returns the URL to where Object Help is
// stored on the Galaxy Repository.
// The client can display the URL in
// InternetExplorer or use web publishing
// to change the data.
HRESULT GetObjectHelpURL(
   // Returns the URL to where the Object Help is
   //stored on the GR.
   [out, retval] BSTR*
);
```

HierarchicalName property

Returns the hierarchical name of this object.

Class

ITemplate

Syntax

[C#]

```
string HierarchicalName { get; };
```

[Visual C++]

```
// The Hierarchical name of this object. (eg.
// Reactor.Tank.Inlet)
HRESULT HierarchicalName(
```

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

```
   // The hierarchical name of this object. (eg.
   // Reactor.Tank.Inlet)
   [out, retval] BSTR* theHierarchicalName
);
```

Host property

Returns or sets the name of this object's host.

Class

ITemplate

Syntax

[C#]

```
string Host { set; get; };
```

[Visual C++]

```
// Retrieve the name of the Host for this object, or
// blank if unassigned.
[propget]
HRESULT Host(
   // The name of the Host for this object.
   [out, retval] BSTR* theHost
);
// Set the Host for this object.
// If this object is an ApplicationObject, its Area
// and container area also updated.
// The object does not need to be checked out; the
// operation is immediate.
[propput]
HRESULT Host(
   // The name of the Host, or blank to unassign
   // from existing Host.
   [in] BSTR NewHost
);
```

RenameExtensionPrimitive method

Rename the object extension primitive.

Class

ITemplate

Syntax

[C#]

```
void RenameExtensionPrimitive(
```

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

```
   string OldPrimitiveName,
   string NewPrimitiveName
);
```

[Visual C++]

```
// Rename an object extension primitive
//
// Returns S_OK
HRESULT RenameExtensionPrimitive(
   // Old name of primitive extension
   [in] BSTR OldPrimitiveName,
   // New name specified by user
   [in] BSTR NewPrimitiveName
);
```

RenameUDA method

Change the name of a UDA.

Class

ITemplate

Syntax

[C#]

```
void RenameUDA(
   string OldUDAName,
   string NewUDAName
);
```

[Visual C++]

```
// Rename a UDA
//
// Returns:
// S_OK - Successfully rename a UDA
HRESULT RenameUDA(
   // The old external name of the primitive
   [in] BSTR OldUDAName,
   // The new external name of the primitive
   [in] BSTR NewUDAName
);
```

Save method

Save the object.

Class

ITemplate

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Syntax

[C#]

```
void Save();
```

[Visual C++]

```
// Saves the object after it is configured.
// Object must be checked out.
HRESULT Save();
```

Tagname property

Returns or sets this object's tagname.

Class

ITemplate

Syntax

[C#]

```
string Tagname { set; get; };
```

[Visual C++]

```
// Returns the Tagname of this object.
[propget]
HRESULT Tagname(
   // The Tagname of this object.
   [out, retval] BSTR* theTagname
);
// Sets the Tagname of this object.
// The object does not need to be checked out; the
// operation is immediate.
[propput]
HRESULT Tagname(
   // The Tagname of this object.
   [in] BSTR newTagName
);
```

Toolset property

Returns the Toolset that this template belongs to.

Class

ITemplate

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Syntax

[C#]

```
string Toolset { set; get; };
```

[Visual C++]

```
[propget]
HRESULT Toolset(
   [out, retval] BSTR* theToolset
);
[propput]
HRESULT Toolset(
   [in] BSTR theToolset
);
```

Parameters

theToolset

The Toolset that this template belongs to.

UndoCheckOut method

Reverses the check out operation without making changes.

Class

ITemplate

Syntax

[C#]

```
void UndoCheckOut();
```

[Visual C++]

```
// Undo check-out this object.
HRESULT UndoCheckOut();
```

Unload method

Unload the object cache, releasing internal resources obtained by the object during operations. After executing this method, the object is still available. Internal resources will be reloaded as necessary.

Class

ITemplate

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Syntax

[C#]

```
void Unload();
```

[Visual C++]

```
// Unloads the gObject cache. This method does not
// affect functionality. It is used to release
// internal resources obtained by the object during
// operations.
// After this method is called, the object is still
// usable and will reload the internal resources as
// needed.
HRESULT Unload();
```

UpdateUDA method

Save the UDA information.

Class

ITemplate

Syntax

[C#]

```
void UpdateUDA(
   string wszUDAInfo,
   MxDataType DataType,
   MxAttributeCategory category,
   MxSecurityClassification Security,
   bool IsArray,
   object ArrayElementCount
);
```

[Visual C++]

```
// Update a UDA
// Returns:
// S_OK - Successfully update a UDA
HRESULT UpdateUDA(
   // Same XML format. But only one is allowed. The
   // new info will overwrite the old info.
   [in] BSTR wszUDAInfo,
   [in] enum MxDataType Datatype,
   [in] enum MxAttributeCategory Category,
   [in] enum MxSecurityClassification Security,
   [in] VARIANT_BOOL IsArray,
   [in] VARIANT ArrayElementCount
);
```

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

ValidationStatus property

Returns this object's validation status.

Class

ITemplate

Syntax

[C#]

```
EPACKAGESTATUS ValidationStatus { get; };
```

[Visual C++]

```
// The ValidationStatus of this object based on the
// last time the object was validated and saved.
HRESULT ValidationStatus(
   // The validation status of the object.
   [out, retval] enum EPACKAGESTATUS* theValidationStatus
);
```

Warnings property

Returns the list of warnings generated the last time the object was validated and saved.

Class

ITemplate

Syntax

[C#]

```
string[] Warnings { get; };
```

[Visual C++]

```
// The list of warnings that was generated the last
// time the object was validated and saved.
HRESULT Warnings(
   // The list of warnings that was generated the
   //last time the object was validated and saved.
   [out, retval] SAFEARRAY(BSTR)* theWarnings
);
```

IToolset

Represents a toolset.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

IToolset class members

This class has the following members.

Operations

CommandResult property

Returns the CommandResult object, which has the last method call's result.

GetChildToolsets method

Retrieves the list of child toolsets.

MoveToToolset method

Moves the toolset to a new parent toolset.

Name property

Returns the name of the toolset.

Rename method

Renames this toolset.

CommandResult property

Returns the CommandResult object, which has the last method call's result.

Class

IToolset

Syntax

[C#]

```
ICommandResult CommandResult { get; };
```

[Visual C++]

```
HRESULT CommandResult(
   [out, retval] ICommandResult** CommandResult
);
```

Parameters

CommandResult

Returns the CommandResult object.

GetChildToolsets method

Retrieves the list of child toolsets.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Class

IToolset

Syntax

[C#]

```
IToolsets GetChildToolsets(
   int depth
);
```

[Visual C++]

```
// Retrieves the list of child Toolsets.
HRESULT GetChildToolsets(
// the level of children to retrieve: pass -1 for N.
   [in] int depth,
   [out, retval] IToolsets** toolsets
);
```

MoveToToolset method

Moves the toolset to a new parent toolset.

Class

IToolset

Syntax

[C#]

```
void MoveToToolset(
   string parentToolset
);
```

[Visual C++]

```
// Moves toolset to a new parent toolset
HRESULT MoveToToolset(
   [in] BSTR parentToolset
);
```

Name property

Returns the name of the toolset.

Class

IToolset

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Syntax

[C#]

```
string Name { get; };
```

[Visual C++]

```
HRESULT Name(
   [out, retval] BSTR* theToolset
);
```

Parameters

theToolset

The toolset name.

Rename method

Renames this toolset.

Class

IToolset

Syntax

[C#]

```
void Rename(
   string newName
);
```

[Visual C++]

```
HRESULT Rename(
   [in] BSTR newName
);
```

Parameters

newName

The new name of the toolset.

IToolsets

Represents a collection of toolsets.

IToolsets class members

This class has the following members.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Operations

Add method

Add a new toolset to the Galaxy.

AddToolSet method

Add a new toolset to the Galaxy.

DeleteToolSet method

Delete the toolset from the Galaxy.

Item property

Returns a toolset.

Count property

Returns the number of toolsets in the collection.

Add method

Adds a new toolset to the Galaxy.

Class

IToolsets

Syntax

[C#]

```
void Add(
   string newToolset
);
```

[Visual C++]

```
HRESULT Add(
   [in] BSTR newToolset
);
```

Parameters

newToolset

The name of the new toolset.

AddToolSet method

Adds the toolset to the Galaxy.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Class

IToolsets

Syntax

[C#]

```
void AddToolSet(
   string parentToolset,
   string newToolset
);
```

[Visual C++]

```
// Add a new toolset to the Galaxy.
HRESULT AddToolSet(
   [in] BSTR parentToolset,
   [in] BSTR newToolset
);
```

DeleteToolSet method

Deletes the toolset from the Galaxy.

Class

IToolsets

Syntax

[C#]

```
void DeleteToolSet(
   string toolsetToDelete
);
```

[Visual C++]

```
// Delete an existing toolset from the Galaxy.
HRESULT DeleteToolSet(
   [in] BSTR toolsetToDelete
);
```

Item property

Returns a toolset.

Class

IToolsets

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Syntax

[C#]

```
IToolset this[object toolsetIdentifier] { get; };
```

[Visual C++]

```
HRESULT Item(
   [in] VARIANT toolsetIdentifier,
   [out, retval] IToolset**
);
```

Parameters

toolsetIdentifier

The index of a toolset. This is a numeric value from 1 to Count. Or it is the name of the toolset.

IToolset

The returned toolset.

Count property

Returns the number of toolsets in the collection.

Class

IToolsets

Syntax

[C#]

```
int count { get; };
```

[Visual C++]

```
HRESULT Count(
   [out, retval] LONG*
);
```

Type definitions

This section describes the enumerations defined and exposed through GRAccess. These include:

EATTRIBUTEPROPERTY

EActionForCurrentlyDeployedObjects

EAuthenticationMode

EAutomaticallyUndocheckout

EBASERUNTIMEOBJECT

ECATEGORY

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

ECOMMONATTRIBUTES
ECOMMONPRIMITIVE
ECascade
ECheckoutStatus
EConditionType
EDeployOnScan
EDeploymentStatus
EEXECUTIONGROUP
EEditStatus
EExecutionOrder
EExportType
EFileType
EForceDeleteInstanceOption
EForceDeleteTemplateOption
EForceOffScan
EGRCommandResult
EMatch
EPACKAGESTATUS
ERESERVEDPRIMITIVEIDS
ESkipIfCurrentlyUndeployed
ESkipObjectsWithPendingUpdates
ESkipOtherUsersCheckedOutObjects
ETestOnly
EUserDefault
EgObjectIsTemplateOrInstance
MxAttributeCategory
MxPropertyLockedEnum

EATTRIBUTEPROPERTY

Lists property IDs of all attributes.

enum **EATTRIBUTEPROPERTY**

```
{idxAttribPropUndefined = -1
idxAttribPropName = 0,
idxAttribPropAttributeCategory = 1,
idxAttribPropCfgSethandler = 2,
idxAttribPropRtSethandler = 3,
idxAttribPropType = 4,
idxAttribPropUpperBoundDim1 = 5,
idxAttribPropValue = 10,
idxAttribPropSecurityClassification = 11,
idxAttribPropLocked = 12,
```

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

```
idxAttribPropQuality = 13,
idxAttribPropBit00 = 14,
idxAttribPropBit01 = 15,
idxAttribPropBit02 = 16,
idxAttribPropBit03 = 17,
idxAttribPropBit04 = 18,
idxAttribPropBit05 = 19,
idxAttribPropBit06 = 20,
idxAttribPropBit07 = 21,
idxAttribPropBit08 = 22,
idxAttribPropBit09 = 23,
idxAttribPropBit10 = 24,
idxAttribPropBit11 = 25,
idxAttribPropBit12 = 26,
idxAttribPropBit13 = 27,
idxAttribPropBit14 = 28,
idxAttribPropBit15 = 29,
idxAttribPropBit16 = 30,
idxAttribPropBit17 = 31,
idxAttribPropBit18 = 32,
idxAttribPropBit19 = 33,
idxAttribPropBit20 = 34,
idxAttribPropBit21 = 35,
idxAttribPropBit22 = 36,
idxAttribPropBit23 = 37,
idxAttribPropBit24 = 38,
idxAttribPropBit25 = 39,
idxAttribPropBit26 = 40,
idxAttribPropBit27 = 41,
idxAttribPropBit28 = 42,
idxAttribPropBit29 = 43,
idxAttribPropBit30 = 44,
idxAttribPropBit31 = 45,
idxAttribPropValueReadOnly = 46,
idxAttribPropSecurityClassificationReadOnly = 47,
idxAttribPropLockedReadOnly = 48,
idxAttribPropTime = 49,
idxAttribPropBuffer = 50,
idxAttribPropHasBuffer = 51,
idxAttribPropEND = 46
}
```

Members

idxAttribPropType

MxDataTypeEnum

idxAttribPropUndefined

idxAttribPropUpperBoundDim1

MxInteger

idxAttribPropQuality

MxDataQualityEnum

idxAttribPropAttributeCategory

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

MxInteger
idxAttribPropLocked
MxInteger
idxAttribPropBit05
idxAttribPropBit02
idxAttribPropBit00
idxAttribPropBit01
idxAttribPropBit03
idxAttribPropBit04
idxAttribPropBit06
idxAttribPropBit07
idxAttribPropBit08
idxAttribPropBit09
idxAttribPropBit10
idxAttribPropBit11
idxAttribPropBit12
idxAttribPropBit13
idxAttribPropBit14
idxAttribPropBit15
idxAttribPropBit16
idxAttribPropBit17
idxAttribPropBit18
idxAttribPropBit19
idxAttribPropBit20
idxAttribPropBit21
idxAttribPropBit22
idxAttribPropBit23
idxAttribPropSecurityClassification
MxSecurityClassificationEnum
idxAttribPropBit24
idxAttribPropBit25
idxAttribPropBit26
idxAttribPropBit27
idxAttribPropBit28
idxAttribPropBit29
idxAttribPropBit30
idxAttribPropRtSethandler
MxBoolean

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

idxAttribPropBit31

idxAttribPropValueReadOnly

MxBoolean

idxAttribPropSecurityClassificationReadOnly

MxBoolean

idxAttribPropLockedReadOnly

MxBoolean

idxAttribPropTime

MxDouble

idxAttribPropBuffer

MxBoolean

idxAttribPropHasBuffer

MxCustomStruct

idxAttribPropEND

idxAttribPropValue

Depends on datatype

idxAttribPropName

MxString

idxAttribPropCfgSethandler

MxBoolean

EActionForCurrentlyDeployedObjects

Action for currently deployed objects.

enum **EActionForCurrentlyDeployedObjects**

{

```
skipDeploy = 0,
redeployOriginal = 1,
deployChanges = 2
}
```

Members

deployChanges

Deploy updated versions.

skipDeploy

Skip if currently deployed.

redeployOriginal

redeploy orignal version.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

EAuthenticationMode

Indicates whether operating system or Galaxy authentication should be used.

enum **EAuthenticationMode**

{

```
osAuthenticationMode = 0,
galaxyAuthenticationMode = 1,
}
```

Members

galaxyAuthenticationMode

Galaxy Authentication Mode

osAuthenticationMode

OS Authentication Mode

EAutomaticallyUndocheckout

Automatically undoes checkout.

enum **EAutomaticallyUndocheckout**

{

```
dontAutomaticallyUndocheckout = 0,
doAutomaticallyUndocheckout = 1
```

}

Members

dontAutomaticallyUndocheckout

Fail if any object is checked out to the current user.

doAutomaticallyUndocheckout

Automatically undoes checkout of any check-outs to the current user before processing.

EBASERUNTIMEOBJECT

Specifies the IDs of the attributes of the base run time object to be used for deployment.

enum **EBASERUNTIMEOBJECT**

```
{
idxBROAttributeUndefined = -1,
idxBROPrimitiveList = 100,
idxBRODeployComplete = 101,
idxBROCreateDynamicAttributes = 102,
idxBRODumpState = 103
```

}

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Members

idxBROAttributeUndefined

idxBROPrimitiveList

Array of MxQualifiedStruct - primitiveId, parentId, primitiveGUID.

idxBRODeployComplete

MxBoolean

idxBROCreateDynamicAttributes

idxBRODumpState

ECATEGORY

Lists categories of attributes.

enum **ECATEGORY**

{

```
idxCategoryUndefined = 0,
idxCategoryPlatformEngine = 1,
idxCategoryClusterEngine = 2,
idxCategoryApplicationEngine = 3,
idxCategoryViewEngine = 4,
idxCategoryProductEngine = 5,
idxCategoryHistoryEngine = 6,
idxCategoryPrintEngine = 7,
idxCategoryOutpost = 8,
idxCategoryQueryEngine = 9,
idxCategoryApplicationObject = 10,
idxCategoryIONetwork = 11,
idxCategoryIODevice = 12,
idxCategoryArea = 13,
idxCategoryUserProfile = 14,
idxCategoryDisplay = 15,
idxCategorySymbol = 16,
idxCategoryViewApp = 17,
idxCategoryProductionObject = 18,
idxCategoryReport = 19,
idxCategorySharedProcedure = 20,
idxCategoryInsertablePrimitive = 21,
idxCategoryIDEMacro = 22,
idxCategoryGalaxy = 23
}
```

Members

idxCategoryViewApp

idxCategoryViewEngine

idxCategoryOutpost

idxCategoryApplicationObject

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

idxCategoryUserProfile
idxCategoryReport
idxCategoryGalaxy
idxCategoryPlatformEngine
idxCategoryIDEMacro
idxCategoryUndefined
idxCategoryHistoryEngine
idxCategoryPrintEngine
idxCategoryClusterEngine
idxCategoryDisplay
idxCategoryProductEngine
idxCategoryApplicationEngine
idxCategorySymbol
idxCategoryProductionObject
idxCategoryInsertablePrimitive
idxCategoryQueryEngine
idxCategoryIODevice
idxCategoryArea
idxCategorySharedProcedure
idxCategoryIONetwork

ECOMMONATTRIBUTES

Lists the attributes common to all primitives.

enum **ECOMMONATTRIBUTES**

{

```
idxComAttUnknown = -1,
idxComAttExternalName = 1,
idxComAttInternalName = 2
}
```

Members

idxComAttExternalName
idxComAttInternalName
idxComAttUnknown

ECOMMONPRIMITIVE

Specifies the IDs of the attributes of the common primitive.

enum **ECOMMONPRIMITIVE**

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

```
{
idxCommonUndefined = -1,
idxCommonExternalName = 1,
idxCommonInternalName = 2,
idxCommonTagname = 100,
idxCommonShortDescription = 101,
idxCommonCategoryIDEnum = 102,
idxCommonAttributes = 103,
idxCommonCmdScanState = 104,
idxCommonScanState = 105,
idxCommonSecurityGroup = 106,
idxCommonError = 107,
idxCommonRelativeOrderEnum = 108,
idxCommonArea = 109,
idxCommonContainer = 110,
idxCommonHost = 111,
idxCommonAlarmModeEnum = 112,
idxCommonAlarmMode = 113,
idxCommonAlarmModeCmd = 114,
idxCommonAlarmInhibit = 115,
idxCommonInAlarm = 116,
idxCommonBasedOn = 117,
idxCommonIsTemplate = 118,
idxCommonCodeBase = 119,
idxCommonUDAs = 120,
idxCommon_InheritedUDAs = 121,
idxCommonExtensions = 122,
idxCommon_InheritedExtensions = 123,
idxCommon_Category = 124,
idxCommon_ConfigVersion = 125,
idxCommon_Warnings = 126,
idxCommonContainedName = 127,
idxCommonExecutionRelatedObject = 128,
idxCommonExecutionRelativeOrder = 129,
idxCommonHierachicalName = 130,
idxCommonMinorVersion = 131
}
```

Members

idxCommonAlarmMode

idxCommonBasedOn

idxCommonExecutionRelatedObject

idxCommonContainedName

idxCommonInAlarm

idxCommonCategoryIDEnum

idxCommonCategoryEnum

idxCommonExecutionRelativeOrder

idxCommonUndefined

idxCommonSecurityGroup

idxCommon_Warnings

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

idxCommonAlarmModeEnum
idxCommonInternalName
idxCommon_InheritedUDAs
idxCommon_InheritedExtensions
idxCommonShortDescription
idxCommonExtensions
idxCommonHierachicalName
idxCommonExternalName
idxCommonCodeBase
idxCommonRelativeOrderEnum
idxCommonExecutionRelativeOrderEnum
idxCommonUDAs
idxCommonContainer
idxCommonAlarmInhibit
idxCommonArea
idxCommonHost
idxCommonTagname
idxCommonError
idxCommonErrors
idxCommonAttributes
idxCommon_Category
idxCommonCmdScanState
idxCommonScanStateCmd
idxCommon_ConfigVersion
this is for ConfigVersion
idxCommonScanState
idxCommonScanStateActual
idxCommonMinorVersion
idxCommonAlarmModeCmd
idxCommonIsTemplate

ECascade

Cascade operation.

enum **ECascade**

{

```
dontCascade = 0,
doCascade = 1,
```

}

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Members

dontCascade

Don't cascade

doCascade

Do force off scan

ECheckoutStatus

Indicates the check-out status of an object.

enum ECheckoutStatus

{

```
notCheckedOut = 0,
checkedOutToMe = 1,
checkedOutToSomeoneElse = 2
```

}

Members

notCheckedOut

The object is not checked-out.

checkedOutToSomeoneElse

The object is checked out to someone else.

checkedOutToMe

The object is checked out the currently logged-in user.

EConditionType

Used by the querying methods to get a list of objects from the GR. Indicates the type of condition used by the query.

enum **EConditionType**

{

```
derivedOrInstantiatedFrom = 1,
basedOn = 2,
containedBy = 3,
hostEngineIs = 4,
belongsToArea = 5,
assignedTo = 6,
withinSecurityGroup = 7,
createdBy = 8,
lastModifiedBy = 9,
checkedOutBy = 10,
namedLike = 11,
validationStatusIs = 12,
deploymentStatusIs = 13,
```

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

```
checkoutStatusIs = 14,
objectCategoryIs = 15,
hierarchicalNameLike = 16,
NameEquals = 17
NameSpaceIdIs = 18
}
```

Members

derivedOrInstantiatedFrom

Search for objects that are derived or instantiated from the given template (first level children only.)

objectCategoryIs

Search for objects with the check-out status. Value parameter should be the numeric value represented by ECATEGORY.

deploymentStatusIs

Search for objects with the given deployment status. Value parameter should be the numeric value represented by EDeploymentStatus.

hierarchicalNameLike

Search for objects whose hierarchical name is like the given text.

belongsToArea

Search for objects whose Area is the given object.

withinSecurityGroup

Search for objects that match the given security group.

lastModifiedBy

Search for objects that were modified by the given user.

checkedOutBy

Search for objects that are checked out by the given user.

validationStatusIs

Search for objects with the given validation status. Value parameter should be the numeric value represented by EPACKAGESTATUS.

hostEngineIs

Search for objects hosted by the given engine (Domain objects of an AppEngine or Engines of a Platform.)

namedLike

Search for objects whose name is like the given text.

NameEquals

Search for objects whose name is equal to the given text.

NameSpaceIdIs

Search for NameSpaceId equal to the value of AutomationObject = 1, BackupObject = 2 or VisualElement = 3

containedBy

Search for objects contained by the given object.

checkoutStatusIs

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Search for objects with the check-out status. Value parameter should be the numeric value represented by ECheckoutStatus.

createdBy

Search for objects that were created by the given user.

assignedTo

Search for objects hosted by the given object (ApplicationObjects or IONetworks of an Area, IODevices of an IONetwork, Domain objects of an AppEngine, or Engines of a Platform)

basedOn

Search for objects that are based on or instantiated from the given base template (all descendents).

ShowInArchestraBrowser

Search for Objects that "show_ in_archestra_browser" = = given value

objectCategoryGUIDIs

Search Objects with "object category guid" = = given value.

HasExtensionType

Search automation objects for extension type. Possible values are:

"alarmextension"	"analogextension"
"AttributeExtension"	"badvaluealarmextension"
"booleanextension"	"DisplayExtension"
"historyextension"	"inputextension"
"inputoutputextension"	"logdatachangeeventextension"
"ObjectExtension"	"outputextension"
"ScriptExtension"	"SymbolExtension"

EDeployOnScan

Deploy on scan.

enum **EDeployOnScan**

{

```
dontDeployOnScan = 0,
doDeployOnScan = 1,
```

}

Members

doDeployOnScan

Do deploy on scan.

dontDeployOnScan

Don't deploy on scan.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

EDeploymentStatus

Indicates the whether the object is deployed.

enum **EDeploymentStatus**

{

```
   notDeployed = 0,
   deployed = 1,
   deployedWithPendingChanges = 2
```

}

Members

notDeployed

The object is not deployed.

deployed

The object is deployed.

deployedWithPendingChanges

The object is deployed and it has pending changes.

EEXECUTIONGROUP

Lists ExecutionGroup options.

enum **EEXECUTIONGROUP**

{

```
idxExeGroupUnknown = -1,
idxExeGroupCommon = 0,
idxExeGroupInput = 1,
idxExeGroupPreScript = 2,
idxExeGroupCustom = 3,
idxExeGroupCustom1 = 4,
idxExeGroupCustom2 = 5,
idxExeGroupCustom3 = 6,
idxExeGroupCustom4 = 7,
idxExeGroupCustom5 = 8,
idxExeGroupCustom6 = 9,
idxExeGroupCustom7 = 10,
idxExeGroupCustom8 = 11,
idxExeGroupCustom9 = 12,
idxExeGroupCustom10 = 13,
idxExeGroupExpression = 14,
idxExeGroupScript = 15,
idxExeGroupPostScript = 16,
idxExeGroupOutput = 17,
idxExeGroupHistory = 18,
idxExeGroupAlarm = 19,
```

}

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Members

idxExeGroupCustom6
idxExeGroupCustom4
idxExeGroupCustom5
idxExeGroupCustom2
idxExeGroupCustom3
idxExeGroupCustom1
New Default
idxExeGroupCommon
idxExeGroupCustom7
idxExeGroupCustom8
idxExeGroupCustom9
idxExeGroupScript
idxExeGroupOutput
idxExeGroupExpression
idxExeGroupInput
idxExeGroupPostScript
idxExeGroupCustom10
idxExeGroupCustom
idxExeGroupAlarm
idxExeGroupPreScript
idxExeGroupHistory
idxExeGroupUnknown

EEditStatus

Indicates the edit status of an object.

enum **EEditStatus**

{

```
notBeingEdited = 0,
editedByThisSession = 1,
editedByYouInOtherSession = 2,
editedByAnotherUser = 3,
```

}

Members

editedByThisSession

The object is being edited by this session.

editedByYouInOtherSession

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

The object is being edited by you in another session.

notBeingEdited

The object is not being edited.

editedByAnotherUser

The object is being edited by another user.

EExecutionOrder

Indicates the execution order of an object as related to the object specified by ExecutionRelatedObject.

enum EExecutionOrder

{

```
The execution order doesn't matter and is determined by the Engine hosting the object.
NONE = 1,
The object should run before the object referenced by ExecutionRelatedObject.
BEFORE,
The object should run after the object referenced by ExecutionRelatedObject.
AFTER
```

}

Members

NONE

BEFORE

AFTER

EExportType

Indicates the type of export: aaPKG.

enum **EExportType**

{

```
exportAsPDF = 0,
exportAsCSV = 1,
```

}

Members

exportAsPDF

Export to a .PDF file. The resulting cab file contains a list of all templates and instances (along with required templates) plus any code modules.

exportAsCSV

Export instances to a text .CSV which can be opened in Excel.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

EFileType

Describes the type of file that is in the PIM file repository. PIM uses this to determine how to install the file.

enum **EFileType**

{

```
eUndefinedFileType = 0,
eNormal = 1,
eComDLL = 2,
eComEXE = 3,
eNTService = 4,
eMergeRegistryScript = 5,
eForHTMLEditor = 6,
eDictionary = 7,
eMsiMergeModule = 8,
eNETFrameworkAssembly
```

}

Members

eForHTMLEditor

File is an HTML editor code module file. Store in Vendorname\Primitive GUID subdirectory instead of Vendorname subdirectory.

eNormal

Normal file. Do nothing extra except install file.

eMsiMergeModule

File is an MSI merge module, which will be merged into an installable MSI file.

eComEXE

File is a COM local server .EXE. Run "<filename> /RegServer".

eUndefinedFileType

eComDLL

File is a COM in-proc server .DLL. Run regsvr32 on the file.

eNETFrameworkAssembly

File is a .NET Framework assembly.

eDictionary

File is an ArchestrA Dictionary file.

eNTService

File is an NT Service .EXE. Run "<filename> -Service".

eMergeRegistryScript

File is a .REG registry script. Merge it into system registry.

EForceDeleteInstanceOption

Indicates whether a deployed instance should be undeployed prior to deletion.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

enum **EForceDeleteInstanceOption**

{

```
dontForceInstanceDelete = 0,
undeployIfNotRunningOnScan = 1,
undeployIfDeployed = 2
```

}

Members

undeployIfDeployed

If object is deployed, set it off-scan if needed, and undeploy prior to deletion.

dontForceInstanceDelete

Don't force the deletion of the instance.

undeployIfNotRunningOnScan

If object is deployed, undeploy it prior to deletion. Object must be off-scan.

EForceDeleteTemplateOption

Indicates how to handle descendents prior to deletion.

enum **EForceDeleteTemplateOption**

{

```
dontForceTemplateDelete = 0,
cascadeDeleteDontUndeploy = 1,
cascadeDeleteUndeployDescendentsIfRunningOffScan = 2,
cascadeDeleteUndeployDescendentsIfRunning = 3
```

}

Members

cascadeDeleteUndeployDescendentsIfRunning

Prior to deletion, decendent instances are first undeployed (and forced off-scan if needed) and then deleted.

cascadeDeleteUndeployDescendentsIfRunningOffScan

Prior to deletion, decendent instances are first undeployed and then deleted. Undeployed objects must be off-scan.

cascadeDeleteDontUndeploy

Do delete descendent objects first, but only if they're not deployed.

dontForceTemplateDelete

Don't force the deletion of the template. The template cannot have any derived descendents.

EForceOffScan

Force off-scan.

enum **EForceOffScan**

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

{

```
dontForceOffScan = 0,
doForceOffScan = 1,
```

}

Members

dontForceOffScan

Don't force off-scan.

doForceOffScan

Do force off-scan.

EGRCommandResult

Indicates the result of a command issued to the GR.

enum **EGRCommandResult**

{

```
cmdUnknownError = -1,
cmdSuccess = 0,
cmdInsufficientPermissions = 101,
cmdNoSuchGRNode = 102,
cmdNoSuchUser = 103,
cmdPasswordIncorrect = 104,
cmdLicenseUnavailable = 105,
cmdNoSuchFile = 201,
cmdCouldntCreateFile = 202,
cmdObjectIsCheckedOut = 301,
cmdObjectIsCheckedOutToSomeoneElse = 302,
cmdObjectNotCheckedOutToMe = 303,
cmdObjectCannotBeOverwritten = 401,
cmdTemplateInUse = 402,
cmdObjectIsAContainer = 403,
cmdObjectHostNotFound = 404,
cmdInstanceIsHost = 405,
cmdObjectIsRequired = 406,
cmdObjectNotAnAutomationObject = 407,
cmdObjectInBadState = 408,
cmdCustomConfigurationError = 409,
cmdObjectInReadOnlyMode = 410,
cmdObjectHostNotDeployed = 501,
cmdInstanceIsDeployed = 502,
cmdInvalidGRLoadMode = 503,
cmdOEMVersionIncompatible = 504,
cmdRetryClientSync = 505,
cmdSyncSemaphore = 506,
cmdOutOfSync = 507
```

}

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Members

cmdLicenseUnavailable
Operation failed because a license problem occurred.
cmdCustomConfigurationError
The operation failed because of configuration error.
cmdObjectHostNotFound
The operation failed because the given instance does not have a host.
cmdInsufficientPermissions
Operation failed because the user has insufficient permission.
cmdNoSuchGRNode
Operation failed because the given GR Node is invalid.
cmdObjectIsAContainer
The operation failed because the object is a container of other objects.
cmdObjectHostNotDeployed
The operation failed because the host of the given instance is not deployed.
cmdNoSuchFile
The specified file was not found.
cmdObjectNotAnAutomationObject
The operation failed because the object is not an AutomationObject.
cmdObjectIsRequired
The object could not be deleted because it is a required object.
cmdSuccess
Success code.
cmdUnknownError
Unknown error.
cmdObjectInBadState
The operation failed because the object is in a bad state.
cmdObjectCannotBeOverwritten
The specified object already exists and cannot be overwritten.
cmdObjectIsCheckedOutToSomeoneElse
The operation failed because the object is currently checked out to someone else.
cmdObjectInReadOnlyMode
The operation failed because the object is in read-only mode.
cmdTemplateInUse
The operation failed because the template has one or more objects derived from it.
cmdInstanceIsHost
The operation failed because the instance is a host of other objects.
cmdCouldntCreateFile

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

The operation failed because the file could not be created.

cmdInstanceIsDeployed

The operation failed because the instance is currently deployed.

cmdObjectNotCheckedOutToMe

The operation failed because the object is not checked out to the current user.

cmdObjectIsCheckedOut

The operation failed because the object is currently checked out.

cmdNoSuchUser

Operation failed because the given user name is invalid.

cmdPasswordIncorrect

Operation failed because the given password is invalid.

cmdInvalidGRLoadMode

The operation failed because enum GRLoad mode is invalid.

cmdOEMVersionIncompatible

The operation failed because client and galaxy OEMVersions are incompatible.

cmdRetryClientSync

The operation failed because enum cmdRetryClientSync mode is invalid.

cmdSyncSemaphore

The operation failed because enum cmdForceSynchronize mode is invalid.

cmdOutOfSync

The operation failed because enum cmdOutOfSync mode is invalid.

EMatch

Indicates which results should be returned: condition match or negate result.

enum **EMatch**

{

```
NotMatchCondition = 0,
MatchCondition = 1
```

}

Members

MatchCondition

Condition match results.

NotMatchCondition

Condition match negate results.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

EPACKAGESTATUS

Indicates the status of an object after it is validated.

enum **EPACKAGESTATUS**

{

```
Package configured properly.
ePackageUnknownStatus = -1,
The object is properly configured and can be deployed.
ePackageGood = 0,
The object is not properly configured, but it can be deployed.
ePackageBad = 1,
The object is not properly configured, and cannot be deployed.
ePackageWarning = 2
```

}

Members

ePackageBad

ePackageGood

ePackageWarning

ePackageUnknownStatus

ERESERVEDPRIMITIVEIDS

Specifies the primitive IDs of the common primitives and the topmost custom primitive.

enum **ERESERVEDPRIMITIVEIDS**

{

```
eReservedPrimitiveIdUndefined = -1,
eBaseRuntimeObjectId = 1,
eCommonPrimitiveId = 2,
eTopLevelCustomPrimitiveId = 100
```

}

Members

eCommonPrimitiveId

eTopLevelCustomPrimitiveId

eReservedPrimitiveIdUndefined

eBaseRuntimeObjectId

Primitive ID reserved for BRO use during deployment.

ESkipIfCurrentlyUndeployed

Skip if currently undeployed.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

enum **ESkipIfCurrentlyUndeployed**

{

```
dontSkipIfCurrentlyUndeployed = 0,
doSkipIfCurrentlyUndeployed = 1,
```

}

Members

doSkipIfCurrentlyUndeployed

Do skip if currently undeployed.

dontSkipIfCurrentlyUndeployed

Don't skip if currently undeployed.

ESkipObjectsWithPendingUpdates

Skip objects with pending updates.

enum **ESkipObjectsWithPendingUpdates**

{

```
dontSkipObjectsWithPendingUpdates = 0,
doSkipObjectsWithPendingUpdates = 1
```

}

Members

dontSkipObjectsWithPendingUpdates

Upload will fail if any object has pending updates.

doSkipObjectsWithPendingUpdates

Automatically skip objects that have pending updates.

ESkipOtherUsersCheckedOutObjects

Skip other users' CheckedOuts.

enum **ESkipOtherUsersCheckedOutObjects**

{

```
dontSkipOtherUsersCheckedOutObjects = 0,
doSkipOtherUsersCheckedOutObjects = 1
```

}

Members

doSkipOtherUsersCheckedOutObjects

Automatically skip objects that are checked out to another user.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

dontSkipOtherUsersCheckedOutObjects

Upload will fail if any objects are checked out to other users.

ETestOnly

Indicates whether the operation is to be performed or just tested to determine if it can be performed.

enum **ETestOnly**

{

```
performOperation = 0,
testOnly = 1,
```

}

Members

performOperation

Perform the operation.

testOnly

Only test to determine if the operation can be performed.

EUserDefault

The user default.

enum **EUserDefault**

{

```
defaultPlatform = 1,
defaultAppEngine = 2,
defaultHistorianNode = 3,
defaultArea = 4,
defaultSecurityGroup = 5
```

}

Members

defaultSecurityGroup	The name of the default security group of the user.
defaultHistorianNode	The name of the historian node name of the user.
defaultAppEngine	The name of the default AppEngine of the user.
defaultArea	The name of the default Area of the user.
defaultPlatform	The name of the default Platform of the user.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

EgObjectIsTemplateOrInstance

Indicates whether a gObject is a template or an instance.

enum **EgObjectIsTemplateOrInstance**

{

```
gObjectIsTemplate = 0,
gObjectIsInstance = 1,
```

}

Members

gObjectIsTemplate

GObject is a Template.

gObjectIsInstance

GObject is a Instance.

MxAttributeCategory

Indicates the category of an attribute.

enum **MxAttributeCategory**

{

```
MxCategoryUndefined = -1,
Can only be configured at IDE time and not Lockable. Attribute doesn't exist at Runtime.
MxCategoryPackageOnly = 0,
Can only be configured at IDE time and is Lockable. Attribute doesn't exist at Runtime.
(eg. CmdDisagreeAlarm.Available)
MxCategoryPackageOnly_Lockable = 1,
Not configurable at IDE time. Calculated and readable at runtime, and not Checkpointed.
(eg. Valve1.PVAuto)
MxCategoryCalculated = 2,
Not configurable at IDE time. Calculated and readable at runtime and Checkpointed. (eg.
Timer or Counter Accumulator)
MxCategoryCalculatedRetentive = 3,
Only the Security Classification is configurable at IDE time. User writable but not
Supervisory writable. Checkpointed. (eg. Valve1.CmdTImeoutAlarm.Acknowledged)
MxCategoryWriteable_U = 4,
Not configurable at IDE time. Supervisory writable but not User writable. Checkpointed.
MxCategoryWriteable_S = 5,
Only the Security Classification is configurable at IDE time. User and Supervisory writable
and Checkpointed. (eg. Valve1.CMD, PID1.SP)
MxCategoryWriteable_US = 6,
Can be configured at IDE time but not lockable. User writable but not Supervisory writable.
Checkpointed. (eg. Valve1.PVOverrideEnabled)
MxCategoryWriteable_UC = 7,
Can be configured at IDE time but not lockable. User and Supervisory writable and
Checkpointed. (eg. Valve1.PVMode)
MxCategoryWriteable_USC = 8,
Can be configured at IDE time and is Lockable. User writable but not Supervisory writable.
Checkpointed. (eg. Valve1.FieldInputToPVMap)
```

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

```
MxCategoryWriteable_UC_Lockable = 9,
Can be configured at IDE time and is Lockable. User and Supervisory writable and
Checkpointed. (eg. Valve1.CmdTImeoutAlarm.Limit)
MxCategoryWriteable_USC_Lockable = 10,
Can be configured at IDE time and is Lockable. Not writable at runtime or checkpointed.
(eg. Valve1.NumInputs)
MxCategoryWriteable_C_Lockable = 11,
Not configurable or Lockable at IDE time. User or Supervisory writable, but System
writable. Not checkpointed. (eg. GREngine._BindReference)
MxCategory_SystemSetsOnly = 12,
Not configurable or Lockable at IDE time. User or Supervisory writable, but System
writable. Not readable at Runtime. Not checkpointed. (eg. _InternalName, _ExternalName)
MxCategory_SystemInternal = 13,
Not configurable or Lockable, but readable at IDE time. Not writable but readable at
Runtime. Not checkpointed. (eg. Tagname)
MxCategory_SystemWriteable = 14,
Defined at SDK time and not modified after that.
MxCategory_Constant = 15,
Defined at Config time. This attribute does not exist at runtime
MxCategory_SystemInternal_Browsable = 16,
Defined at SDK time. This attribute does not exist at runtime
MxCategory_CalculatedPackage = 17,
Defined at SDK time. This attribute is used to send data to runtime.
MxCategory_DeleteAfterStartup = 18,
MxAttributeCategoryEND = 19
```

}

Members

MxCategoryCalculated

MxCategoryWriteable_USC

MxCategoryWriteable_UC_Lockable

MxCategoryWriteable_USC_Lockable

MxCategoryWriteable_C_Lockable

MxCategoryCalculatedRetentive

MxCategoryUndefined

MxCategory_SystemWriteable

MxCategory_Constant

MxCategory_SystemInternal_Browsable

MxCategoryWriteable_US

MxCategoryWriteable_UC

MxCategory_SystemSetsOnly

MxCategory_CalculatedPackage

MxCategory_DeleteAfterStartup

MxAttributeCategoryEND

MxCategoryPackageOnly_Lockable

MxCategory_SystemInternal

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

MxCategoryPackageOnly

MxCategoryWriteable_U

MxCategoryWriteable_S

MxPropertyLockedEnum

Indicates the lock status of an attribute.

enum **MxPropertyLockedEnum**

{

```
MxUndefinedLockedStatus = -1,
The Attribute is Unlocked. Data can be modified.
MxUnLocked = 0,
The Attribute is Locked in this object and propagated to all descendents. Data can be
modified.
MxLockedInMe = 1,
The Attribute is LockedByMe in an ancestor object. Data cannot be modified.
MxLockedInParent = 2,
MxPropertyLockedEnumEND = 3
```

}

Members

MxUndefinedLockedStatus

MxLockedInParent

MxPropertyLockedEnumEND

MxLockedInMe

MxUnLocked

GRAccess code examples

This section provides examples related to performance and other best practices.

GRAccess code examples are located in the following folder:

C:\Program Files (x86)\ArchestrA\Samples

The code examples are:

- CreateGalaxyCPP.

  See Example: define the entry point for a console application
- CreateGalaxyCSharp

  See Example: create and populate a galaxy
- GUISampleApp

  The GUISampleApp implements all aspects of the GRAccess API, and is intended only to demonstrate the functions that can be used in a GRAccess application. It allows you, as a customer or system integrator, to examine the source code in Visual Studio, and to map the results of UI interaction to specific GRAccess API calls.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

WARNING! The GUISampleApp is not intended as a production-ready application. **DO NOT** use the GUISampleApp with production galaxies. It is intended only to give you insight into the GRAccess source code, and let you see how user actions within the GUI affect GRAccess API calls.

Programming tips

Follow these programming tips:

- GRAccessApp m_gr = new GRAccessApp(). m_gr object must be active when you want to use any of GRAccess Objects. This means that it should not go out of scope from your code path.
- CommandResult and CommandResults are error checking methods like GetLastError in C++ and The Err Object in VB. After each GRAccess object API call, you can check that the API succeeded or not. APIs that deal with multiple objects return CommandResults.
- If you are using GRAccess from a remote machine, make sure you have run the OSConfigUtility.
- If needed, modify the import statement in stdafx.h to include (x86) in the file path, as shown in the following example:

```
#import "C:\Program Files (x86)\ArchestrA\Framework\Bin\ArchestrA.GRAccess.dll"
no_namespace, raw_interfaces_only
```

- The import path may vary depending on variables such as the InTouch installation directory. Modify the import statement in stdafx.h to import from the correct library path, as shown in the following example:

```
#import "C:\Program Files (x86)\Wonderware\InTouch\AppServerSecurity.dll"
no_namespace, raw_interfaces_only
```

Example: define the entry point for a console application

The following example shows the entry point for the console application with ATL support:

Note: This code can be found in C:\Program Files (x86)\ArchestrA\Samples\CreateGalaxyCPP.

```
// ----------------------------------------------------
// <copyright company="AVEVA Software, LLC" file="CreateGalaxyCPP.cpp">
// © 2022 AVEVA Software, LLC. All rights reserved.
//
// THIS CODE AND INFORMATION ARE PROVIDED "AS IS" WITHOUT WARRANTY OF ANY
// KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE
// IMPLIED WARRANTIES OF MERCHANTABILITY AND/OR FITNESS FOR A
// PARTICULAR PURPOSE.
// </copyright>
// </summary>
//----------------------------------------------------
// CreateGalaxyCPP.cpp : Defines the entry point for the console application with ATL
support.
//
#include "stdafx.h"
#include <time.h>
#import "GRAccess111.tlb" no_namespace
class COMINIT
{
public:
```

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

```
COMINIT()
   {
      // Initialize COM.
      comInit = CoInitializeEx(
            0,
         //COINIT_APARTMENTTHREADED
            COINIT_APARTMENTTHREADED | COINIT_DISABLE_OLE1DDE
            );
   }
   ~COMINIT()
   {
      // Uninitialize COM.
      if (SUCCEEDED(comInit))
      {
         CoUninitialize();
      }
   }
private:
   HRESULT comInit;
};
int _tmain(int argc, _TCHAR* argv[])
{
   COMINIT SmartComInitialize;
   CComPtr<IGRAccess> spGRApp;
   HRESULT hr = spGRApp.CoCreateInstance ( __uuidof(GRAccessApp));
   if (spGRApp == NULL)
      return -1;
   CComPtr<IGalaxies> spGalaxies;
   DWORD dwsize = MAX_COMPUTERNAME_LENGTH + 1;
   TCHAR szGRNodeComputerName[MAX_COMPUTERNAME_LENGTH + 1];
   ::GetComputerName(szGRNodeComputerName, &dwsize);
   CComBSTR bstrGRNodeComputerName = szGRNodeComputerName ;
   spGalaxies = spGRApp->QueryGalaxies (bstrGRNodeComputerName.m_str);
   CComPtr<ICommandResult> spCommandResult;
   spGRApp->get_CommandResult (&spCommandResult);
   if(spCommandResult->Successful == VARIANT_FALSE || spGalaxies == NULL)
   {
      wprintf(L"%s : %s \n",spCommandResult->CustomMessage.GetBSTR (), spCommandResult-
      >Text.GetBSTR ());
      return 1;
   }
   CComPtr<IGalaxy> spGalaxy;
   _variant_t vtGalName = L"Example1";
   //OR you can use index if you have only one galaxy use 1. All collections are 1 base
   //_variant_t vtGalName = L"1";
   spGalaxies->get_Item (vtGalName,&spGalaxy);
   if(spGalaxy == NULL)
   {
         spGRApp->CreateGalaxy (vtGalName.bstrVal
         ,bstrGRNodeComputerName.m_str,FALSE,galaxyAuthenticationMode,L"");
            CComPtr<ICommandResult> spCommandResult;
            spGRApp->get_CommandResult (&spCommandResult);
            if(spCommandResult->Successful == VARIANT_FALSE || spGalaxies == NULL)
            {
            wprintf(L"Create Galaxy Named Example1 Failed: %s : %s \n",spCommandResult-
            >CustomMessage.GetBSTR (), spCommandResult->Text.GetBSTR ());
```

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

```
            return 1;
         }
         spGalaxies = spGRApp->QueryGalaxies (bstrGRNodeComputerName.m_str);
         spGalaxies->get_Item (vtGalName,&spGalaxy);
      }
      spGalaxy->Login (L"",L"");
      CComPtr<IgObjects> spGObjects;
      _variant_t vtTemplateName = L"$UserDefined";
      spGObjects = spGalaxy->QueryObjects
      (gObjectIsTemplate,NameEquals,&vtTemplateName,MatchCondition );
      CComPtr<IgObject> spGObject;
      spGObjects->get_Item (vtTemplateName,&spGObject);
      CComPtr<IInstance> spInstance;
      CComQIPtr<ITemplate> spTemplate = spGObject;
      time_t ltime;
      time( &ltime );
      WCHAR UTCStr [32];
      wsprintf(UTCStr,L"%ld",ltime);
      _bstr_t instanceName(L"UD_");
      instanceName += UTCStr;
      spInstance = spTemplate->CreateInstance (instanceName,VARIANT_FALSE);
      spInstance->CheckOut ();
      _variant_t vtCount = 5;
      spInstance->AddUDA
      (L"Names",MxString,MxCategoryWriteable_USC_Lockable,MxSecurityOperate,VARIANT_TRUE,vt
      Count);
      CComPtr<IAttributes> spAttributes;
      spInstance->get_ConfigurableAttributes (&spAttributes);
      CComPtr<IAttribute> spAttrib;
      _variant_t vtAttrName = L"Names";
      spAttributes->get_Item (vtAttrName,&spAttrib);
      //Diplay first 5 attribute names from collection
      for (int i = 1; i <= 5; i++)
      {
         CComPtr<IAttribute> spAttrib1 = spAttributes->GetItem(i);
         wprintf(L"%s\n",spAttrib1->GetName().GetBSTR());
      }
      CComPtr<IMxValue> spNamesValue;
      spNamesValue.CoCreateInstance (__uuidof(MxValue));
      for(int i=1;i<=5;i++)
      {
         CComPtr<IMxValue> mxv;
         mxv.CoCreateInstance (__uuidof(MxValue));
         mxv->PutInteger (i);
         // Attribute Data type is string Why Integer ??
         // The reason is MxValue (Just Like Variant) is smart which contains logic
         // to convert from one type to another.
         spNamesValue->PutElement (i,mxv);
      }
      spAttrib->SetValue (spNamesValue);
      spInstance->Save ();
      spInstance->CheckIn (L"");
      spGalaxy->Logout ();
      WCHAR dummy [81];
      wprintf (L"\nPress ENTER to quit: ");
      _getws_s (dummy, 80);
```

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

```
      //char ch = getchar();
      return 0;
   }
```

Example: create and populate a galaxy

The steps below demonstrate how to use GRAccess through C# and Visual Studio.

This procedure creates an ArchestrA Galaxy and populates it with a single instance.

Build and execute the program below. It creates a Galaxy called "Example1" and a single instance of a $UserDefined template with a name based on the current date and time.

Note: The C# console application should work with ATL support.

1. Using Visual Studio, create a new C# console application "CreateGalaxyExample".
2. On the **Main** menu, click **Project / Add Reference** and add a reference to the GRAccess primary interop assembly. By default, this is: C:\Windows\assembly\GAC_MSIL\ArchestrA.GRAccess\2.0.0.0__23106a86e706d0ae\ArchestrA.GRAccess.dll
3. Replace the contents of the generated C# source file with the following text:

   Note: This code can be found in C:\Program Files (x86)\ArchestrA\Samples\CreateGalaxyCSharp.

```
//
-----------------------------------------------------------------------------------------
-----------------
// <copyright company="AVEVA Software, LLC" file="Main.cs">
// © 2022 AVEVA Software, LLC. All rights reserved.
//
// THIS CODE AND INFORMATION ARE PROVIDED "AS IS" WITHOUT WARRANTY OF ANY
// KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE
// IMPLIED WARRANTIES OF MERCHANTABILITY AND/OR FITNESS FOR A
// PARTICULAR PURPOSE.
// </copyright>
// <summary>
//
// </summary>
//
-----------------------------------------------------------------------------------------
-----------------
using System;
using ArchestrA.GRAccess;
class CreateGalaxyExample
{
   [STAThread]
   static void Main()
   {
      string nodeName = Environment.MachineName;
      string galaxyName = "Example1";
      // create GRAccessAppClass object
      GRAccessApp grAccess = new GRAccessAppClass();
      // try to get galaxy
      IGalaxies gals = grAccess.QueryGalaxies(nodeName);
      if (gals == null || grAccess.CommandResult.Successful == false)
      {
   Console.WriteLine(grAccess.CommandResult.CustomMessage + grAccess.CommandResult.Text);
```

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

```
        return;
    }
    IGalaxy galaxy = gals[galaxyName];
    ICommandResult cmd;
    // create galaxy if it doesn't already exist
    if( galaxy == null )
    {
        grAccess.CreateGalaxy(
            galaxyName,
            nodeName,
            false, // no security
            EAuthenticationMode.galaxyAuthenticationMode,
            "" );
        cmd = grAccess.CommandResult;
        if (!cmd.Successful)
        {
            Console.WriteLine("Create Galaxy Named Example1 Failed: " +
            cmd.Text + " : " +
            cmd.CustomMessage);
            return;
    }
    galaxy = grAccess.QueryGalaxies( nodeName )[ galaxyName ];
}
// log in
galaxy.Login( "", "" );
cmd = galaxy.CommandResult;
if (!cmd.Successful)
{
    Console.WriteLine("Login to galaxy Example1 Failed :" +
           cmd.Text + " : " +
           cmd.CustomMessage);
    return;
}
// get the $UserDefined template
string [] tagnames = { "$UserDefined" };
IgObjects queryResult = galaxy.QueryObjectsByName(
    EgObjectIsTemplateOrInstance.gObjectIsTemplate,
    ref tagnames );
cmd = galaxy.CommandResult;
if (!cmd.Successful)
{
    Console.WriteLine("QueryObjectsByName Failed for $UserDefined Template :" +
           cmd.Text + " : " +
           cmd.CustomMessage);
    return;
}
ITemplate userDefinedTemplate = (ITemplate) queryResult[1];
// create an instance of $UserDefined, named with current time
DateTime now = DateTime.Now;
string instanceName = String.Format( "sample_object_{0}_{1}_{2}"
       , now.Hour.ToString( "00" )
       , now.Minute.ToString( "00" )
       , now.Second.ToString( "00" ));
IInstance sampleinst = userDefinedTemplate.CreateInstance(instanceName, true );
//How to edit the object ?
sampleinst.CheckOut();
```

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

```
   sampleinst.AddUDA("Names",
          MxDataType.MxString,
          MxAttributeCategory.MxCategoryWriteable_USC_Lockable,
          MxSecurityClassification.MxSecurityOperate,
          true,
          5);
   IAttributes attrs = sampleinst.ConfigurableAttributes;
   //Diplay first 5 attribute names from collection
   for (int i = 1; i <= 5; i++)
   {
      IAttribute attrb = attrs[i];
      Console.WriteLine(attrb.Name);
   }
   IAttribute attr1 = attrs["Names"];
   MxValue mxv = new MxValueClass();
   // we don't need to check that attribute is array type or not
   // because we set it as array type when we addUDA.
   // I am just showing example, you can do like this.
   if (attr1.UpperBoundDim1 > 0)
   {
      for (int i = 1; i <= attr1.UpperBoundDim1; i++)
      {
         MxValue mxvelement = new MxValueClass();
         mxvelement.PutString("string element number " + i.ToString());
         mxv.PutElement(i, mxvelement);
      }
      attr1.SetValue(mxv);
   }
   sampleinst.Save();
   sampleinst.CheckIn("Check in after addUDA");
   galaxy.Logout();
   Console.WriteLine();
   Console.Write("Press ENTER to quit: ");
   string dummy;
   dummy = Console.ReadLine();
   }
}
```

IDE Extensions development guide

This guide explains how to extend the functionality of the System Platform IDE by adding new commands into the IDE command pool, or by modifying existing IDE commands.

Introduction

The purpose of this document is to explain System Platform IDE extensibility features and help Development, QA and OEM partners develop extensions that can integrate seamlessly with the IDE by providing:

- The contract IDE extensions should follow for seamless integration with the IDE.
- Services provided by IDE to facilitate extension development.
- Pointers to example extensions with and without user interface.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

- Instructions on how to test and install IDE extensions.
- Information on how to leverage IDE extensibility features to develop an IDE extension.
- Tips to avoid making common mistakes when developing IDE extensions.

It is assumed that the reader of this document is familiar with the basic features of ArchestrA from configuration viewpoint, and has good working knowledge of Microsoft .Net programming, especially C# language.

Organization

This document is divided into seven main sections:

- The first section, “Terms and Definitions”, covers glossary of terminology used in this document to help reader understand the concept before going into details.
- The second section, "Overview," provides an overview of extensibility features provided by the System Platform IDE, its contributing subsystems and depicts high level picture of how all components run together.
- The third section, “Creating an IDE Extension”, covers information about creating an UI and non-UI IDE extension.
- The fourth section, “Packaging and Deployment”, provides information on how to package, deploy and test an IDE extension.
- The fifth section, “Extending IDE Specifics”, describes how exactly various aspects of the IDE can be extended to add new features or change behavior of existing ones.
- The sixth section, “Tips on Developing IDE Extensions”, points out common pitfalls that an IDE extension developer should avoid and recommends some optimizations.
- The last section, “Overriding Predefined IDE Commands”, lists all commands defined by the IDE, segregated based on whether they can be overridden by its extensions.

Terms and definitions

This section describes terminology used in the document for easier understanding of the ArchestrA IDE extensibility features.

- Galaxy Explorer – The main Integrated Development Environment (IDE) window that allows users to browse, review and operate on galaxy objects within a Galaxy. When the IDE is started, after connecting to a galaxy the Galaxy Explorer is displayed.
- Galaxy Browser – A convenient tool in the IDE that allows users to search for galaxy objects object attributes, attribute properties and so on.
- Command – A galaxy object specific or global operation, pertaining to a Galaxy from configuration viewpoint. Deploying or checking out an AutomationObject instance could be examples of commands.
- Command UI vector – User interface element through which user can initiate an operation by invoking a command. This includes menus, menu items, toolbars, toolbar items, contextual menus, contextual menus items and keyboard shortcuts.
- Extensible IDE UI elements – All of the IDE command UI vectors and status bar panels constitute extensible IDE user interface (UI) elements.
- Command Override – Conditionally or unconditionally changing behavior of a command declared by the IDE

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

or other extension module. Remember, overriding a command changes the command behavior, not the actual definition of the command, attributes such as command text or command icon will remain unchanged.

- IDE extension – A .Net plug-in module loaded by the IDE at runtime, which seamlessly integrates and interacts with the IDE.
- Command Extension – An IDE extension usually without user interface, which can add new commands, add new command UI vector to IDE or modify behavior of existing commands.
- View Extension – An IDE extension with user interface displayed inside IDE, which can be docked just like default IDE views. A view extension can also be a command extension.
- Command Framework – The IDE framework that manages the application's command set and command UI vector, provides command extensibility and handles command routing.
- Docking Framework – The IDE framework that manages the application's window layout including creation and positioning of docking windows.
- Service – Any reusable functionality provided by the IDE or an extension module, which can be shared between extension modules.
- Service Pool - The application's global service pool that provides functionality for registering and consuming services offered by the IDE and other extension modules.
- IDEExtensions Folder – The predefined IDE folder, a subfolder under IDE application path, where all IDE extensions are installed and IDE loads all of its extensions from.

Overview

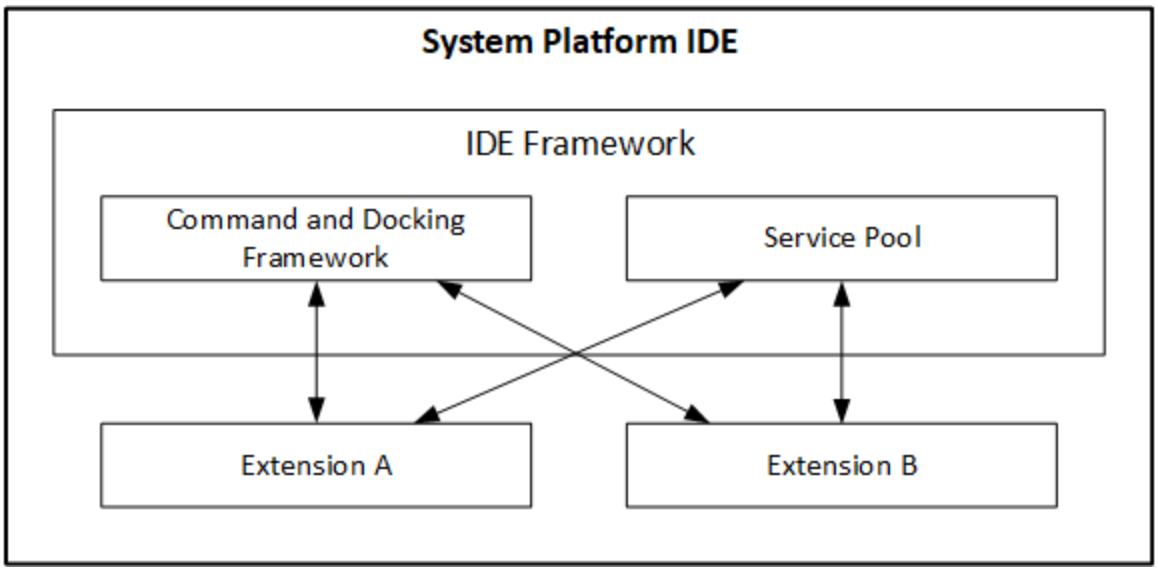
New extensible System Platform IDE comprises of a collection of subsystems that provide the basic functionalities such as command extensibility, view extensibility and docking behavior. It consists of the following sub-systems:

- Docking and Command Framework – Provides window layout and command management support to the extension modules.
- Service Pool – Manages application's global service pool and provides functionality for registering and consuming services offered by the IDE and other extension modules.
- IDE Extensions – Various user interface (UI) and non-UI extension modules.

The System Platform IDE provides a programmatic interface to develop extension modules to extend its default functionality. IDE extensions are the plug-in modules hosted by the IDE at runtime. When the IDE is started, it loads all of its extensions at runtime and interacts with them via predefined .Net interfaces. The Extension modules in turn can interact with the IDE using its services provided in the form of .NET interfaces.

By following the contract established by the IDE, extensions can extend various UI elements of the IDE including docking views, menus, toolbars, and keyboard shortcuts. With the help of IDE services, extension modules can subscribe to commands and handle commands invoked by the user. Thus, the running IDE is composed of its framework, all extension modules from different sources interacting with the IDE framework and with each other to provide the end-user with the necessary tools and operations to configure the Galaxy. The end result is monolithic user experience with the IDE and is depicted in the following diagram.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.



The IDE extensions can change IDE functionality or add new ones by:

- Adding new commands or operations, which user can invoke from within the IDE.
- Overriding behavior of existing IDE commands.
- Adding new Galaxy views that can be hosted inside the IDE.

Get started with IDE Extensions

This section describes how to create an IDE extension. The IDE supports two types of extensions, extensions without user interface called command extensions and ones with user interface called view extensions. They are generically referred to as IDE extensions in this document.

A command extension introduces new command(s) into the global IDE command pool, which will be available to the IDE itself and all other IDE extensions. It can also modify behavior of commands defined by the IDE or other extensions, and add new command vectors like, menu item, toolbar button etc. A command extension is usually a service and does not have user interface, but can bring up a dialog box when a command is invoked and is handled by it.

A view extension has user interface and when displayed inside the Galaxy Explorer can be docked just like native IDE views. A view extension can also implement command extension: a view extension can also introduce new commands into global pool of IDE commands, which will be available to the IDE itself and any other IDE extensions. In short, a view extension is a specialized command extension with user interface. A single IDE extension assembly can have multiple extensions with mix of command and view extensions.

In order to extend IDE by developing a command or view extension, and to achieve this to have galaxy database access, a typical developer would need the following three .Net assembly references added to their .Net project:

- ArchestrA.IDE.Extensibility.dll
- ArchestrA.Configuration.dll
- ArchestrA.Core.dll

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

IDE Extensions manifest

In order for the IDE to recognize its extensions, an extension should provide all information related to views and commands it needs to introduce in the form of embedded .Net manifest resource file. The manifest file is an xml, which follows predefined xml schema. Please refer to the IDE Extensions Xml Schema.xsd for the actual xml schema defined by the IDE for its extensions.

The following is a sample xml that a typical IDE extension would provide in the form of embedded manifest and is used thoughout this document for various examples:

```
<?xml version="1.0" encoding="utf-8"?>
<Root>
<Snapins>
<View TypeId="540C08D5-803A-41f5-8900-86B11A510193" Priority="650">
<Title>Flat Filtered View</Title>
<DefaultPosition>Top</DefaultPosition>
</View>
<View TypeId="2564EEA9-96FD-453d-907E-91FF9CA7CB68" Priority="600">
<Title>Network View</Title>
<DefaultPosition>Bottom</DefaultPosition>
</View>
<Snapin TypeId="907162B9-F0AE-40d0-92C9-79E3DAB04D43" Priority="550">
<Resources>
<Command Id="cmdTestCheckin">
<Name>Test Check-in</Name>
<IdleUpdate>true</IdleUpdate>
<Private>false</Private>
<ContextSensitive>true</ContextSensitive>
<Description>Checks-in the object</Description>
</Command>
<Command Id="cmdTestCheckout">
<Name>Test Check-out</Name>
<ContextSensitive>true</ContextSensitive>
<Description>Checks-out the object</Description>
</Command>
<MenuItem Id = "Object/TestCheckin" ParentId="Object"
ReferencedTo="Object/Checkin" ReferencePosition="After">
<CommandId>cmdTestCheckin</CommandId>
<Text>Test Check-in</Text>
</MenuItem>
<MenuItem Id = "Object/TestCheckout" ParentId="Object"
DefaultPosition="Bottom">
<CommandId>cmdTestCheckout</CommandId>
<Text>Test Check-out</Text>
</MenuItem>
<MenuItem Id = "Context/TestCheckout" ParentId="Context">
<CommandId>cmdTestCheckout</CommandId>
<Text>Test Check-out</Text>
</MenuItem>
<Toolbar Id="tbarTest">
<Name>Test</Name>
</Toolbar>
<ToolbarButton Id="tbarTestCI" ParentId="tbarTest"
DefaultPosition="Top">
<CommandId>cmdTestCheckin</CommandId>
<Image>TestCI_16x16.png</Image>
```

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

```
</ToolbarButton>
<Separator ParentId="tbarTestCO" ReferencedTo="tbarTestCI"
ReferencePosition="Before"/>
<ToolbarButton Id="tbtnHelp" ParentId="tbarTest" Group="Start">
<CommandId> cmdTestCheckout</CommandId>
<Image>TestCO_16x16.png</Image>
</ToolbarButton>
<Accelerator Id = "accelTestCI">
<CommandId>cmdTestCheckin</CommandId>
<Shortcut>CtrlI</Shortcut>
</Accelerator>
</Resources>
</Snapin>
</Snapins>
</Root>
```

Discover IDE Extensions

IDE extensions can be added to the IDE without rebuilding the core IDE. It just requires IDE to be restarted. When the IDE is started, it browses through the IDEExtensions folder and loads all valid IDE extension assemblies, which have required embedded meta-data. The extensions' manifest information is validated against fore mentioned "IDE Extensions Xml Schema". The IDE also verifies that the assemblies are strongly named and are signed using the AVEVA-provided public and private token key-pair.

Load IDE Extensions

The IDE framework provides a mechanism by which a specific sequence of loading among its extensions can be enforced by the developer of the extensions. IDE extensions are loaded in the order of their "Priority", which can be specified in their manifest file. Priority ranges from 1 through 1000, with 1 being the lowest and 999 being the highest priority available for IDE extensions. Priority 1000 is reserved for the IDE host itself for providing very basic functionality needed for all extensions like, galaxy database access. All Wonderware developed extensions have their priority properly set to ensure that they are loaded before other extensions. In the following two entries from the sample manifest, the priority of a view and a command extension is highlighted.

```
<View TypeId="540C08D5-803A-41f5-8900-86B11A510193" Priority="650">
<Snapin TypeId="907162B9-F0AE-40d0-92C9-79E3DAB04D43" Priority="550">
```

Instantiate IDE Extensions

In order for the IDE to instantiate an extension public class at runtime, the extension implementation must be adorned with the custom attribute "`ArchestrA.IDE.Extensibility.aaPluginAttribute`". The following example depicts tagging a public class with the "`aaPluginAttribute`" custom attribute.

```
[aaPlugin("2564EEA9-96FD-453d-907E-91FF9CA7CB68")]
public class NetworkView : System.Windows.Forms.UserControl, IaaIDEView
{
// TODO: Actual Implementation
}
```

Once the IDE discovers all of its extensions and determines their loading order, it tries to create instances of the extensions one by one. In order to find out class to instantiate from the extension module, IDE tries to match the "**TypeId**" value provided in the manifest with the value of the custom attribute "`aaPluginAttribute`" that the

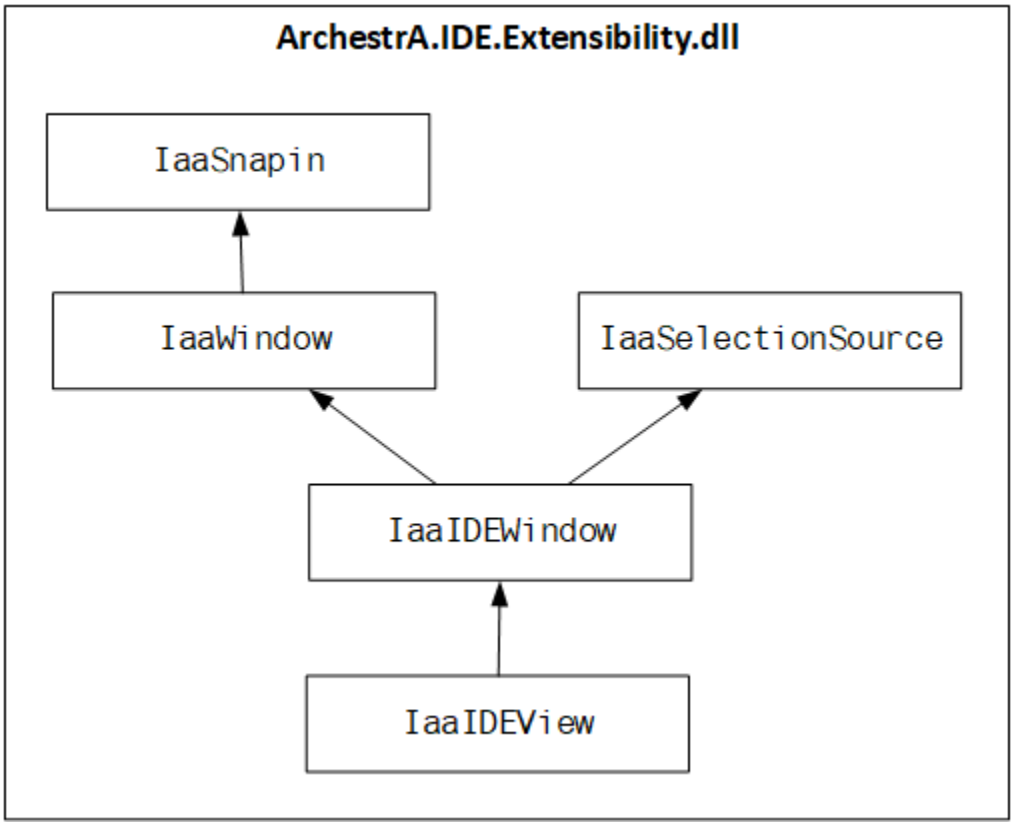
© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

extension public classes are tagged with. This is an **AllowMultiple** type of custom attribute and so, it allows user to adorn single public class with multiple TypeId values. The end result is user can create multiple instances of the same class as IDE extensions. **TypeId** value is typically a GUID. Following is an example extract from the sample manifest, emphasizing on **TypeId** value match with the value of "aaPluginAttribute" custom attribute.

```
< NetworkView TypeId="2564EEA9-96FD-453d-907E-91FF9CA7CB68" Priority="600">
[aaPlugin("2564EEA9-96FD-453d-907E-91FF9CA7CB68")]
public class NetworkView : System.Windows.Forms.UserControl, IaaIDEView
{
// TODO: Actual Implementation
}
```

Create an IDE Extension

In order to support an extensible architecture, the IDE provides various interfaces, which IDE extensions implement. Through these interfaces, the IDE communicates with its extensions. Depending on the type of extension being developed, extension developer must implement various interface(s) listed below. Please see the sample extensions project provided with this document for detailed usage of these interfaces. The following diagram shows inheritance of these IDE provided interfaces.



A typical IDE extension developer would go through the following steps when developing an IDE extension, which are common for any kind of extension:

- Adding references to the three .Net assemblies, as explained in the section "Getting Started with IDE Extensions".
- Adding assembly manifest file, as explained in the section "IDE Extensions Manifest".
- Adorning extension public classes with "aaPluginAttribute" custom attribute, as explained in the section

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

"Instantiating IDE Extensions".

Create a command extension

To develop a command extension, in addition to the steps needed for all IDE extensions, the extension public class must implement "IaaSnapin" and "IaaCommandTarget" interfaces from the "Extensibility" assembly.

When the extension is loaded, IDE calls extension's "Initialize" method in the "IaaSnapin" interface and sets its service provider site, which can be used by the extension to access global IDE service pool, register its own services and consume registered by others. IDE calls extension's "Uninitialize" method on the same interface while unloading the extension, when extensions should un-register their services.

Create a view extension

To develop a view extension, in addition to the steps needed for all IDE extensions, the extension public class must implement "IaaIDEView" interface from the "Extensibility" assembly. If the view needs to handle commands, it must implement "IaaCommandTarget" interface also from the same assembly.

The IDE uses "IaaIDEView" interface to communicate with the view extension, which has four methods and an event as described below:

- When requesting a view to select a specific object in it, IDE calls extension's "SelectObject" method. This is used for example, when synchronizing views.
- To find out whether the active view extension supports in-place renaming for the selected object, the IDE calls extension's "CanInPlaceRename" method.
- If the extension supports in-place renaming then the IDE calls its "BeginInPlaceRename" method to start renaming.
- When any command is invoked, the IDE call "EndInPlaceRename" method on the active view, if it supports in-place renaming to give it a chance to come out of rename.
- The extension is supposed to fire "RenameInPlaceEnd" event when in-place rename ends.

A view extension should provide some details on how and where it should be displayed in the Galaxy Explorer. Following is an example extract from the sample manifest, emphasizing on view details.

```
<Root>
<Snapins>
<View TypeId="540C08D5-803A-41f5-8900-86B11A510193" Priority="650">
<Title>Flat Filtered View</Title>
<DefaultPosition>Top</DefaultPosition>
<Visible>true</Visible>
</View>
</Snapins>
</Root>
```

- The Title tag indicates the view caption for the dockable window in the Galaxy Explorer for the view extension.
- The DefaultPosition tag indicates the initial dock position for the view extension when the IDE loads it for the first time. Since IDE persists its layout information, next time IDE loads the view layout from the persisted layout information. Valid values for the DefaultPosition are Top, Bottom, Left, Right and Floating.
- The Visible tag indicates whether the view extension should be displayed right away when it is loaded and

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

the valid values are “true” and “false”.

The “`IaaIDEView`” derives from “`IaaSelectionSource`” interface, which needs to be implemented by the view extensions. This allows the view extensions to publish its selection information to the IDE. The IDE then passes this information from the active view to the command extensions so that the command states can be appropriately updated. The selection information is also passed on to the command extensions when the command is invoked by the user. The “`IaaSelectionSource`” interface has one property and three events as described below:

- IDE uses the “`SelectedObjects`” property in this interface to get the current user selection in the view whenever user switches from one view to another.
- Whenever the selection changes in the view, the view should fire one of the three following events so that IDE’s selection state stays current. This is needed to update the commands so that all command UI vectors like, menu items and toolbar buttons are in sync with the user selection.
  - If possible, when the user adds a new item to the selection, the event “`SelectionAdd`” should be fired providing the newly selected object’s information in the event arguments.
  - If possible, when the user removes an item from the existing selection, the event “`SelectionRemove`” should be fired providing the information on the object getting de-selected in the event arguments.
  - At a minimum, whenever the selection changes, the view should fire the “`SelectionReset`” event, and provide the new selection in the view as the arguments.

IDE object type system

Since the running IDE consists of different extensions from different sources, possibly developed at different points of time, it is necessary that a standard and extensible system is available to exchange selection information between the view extensions which the user interacts with and the command extensions that perform operations for the user.

IDE provides a basic set of object types that cover the commonly used galaxy objects. All the view extensions should publish their selection information using these standard types so that command extensions can use this information to distinguish between the different objects that are in the user’s selection.

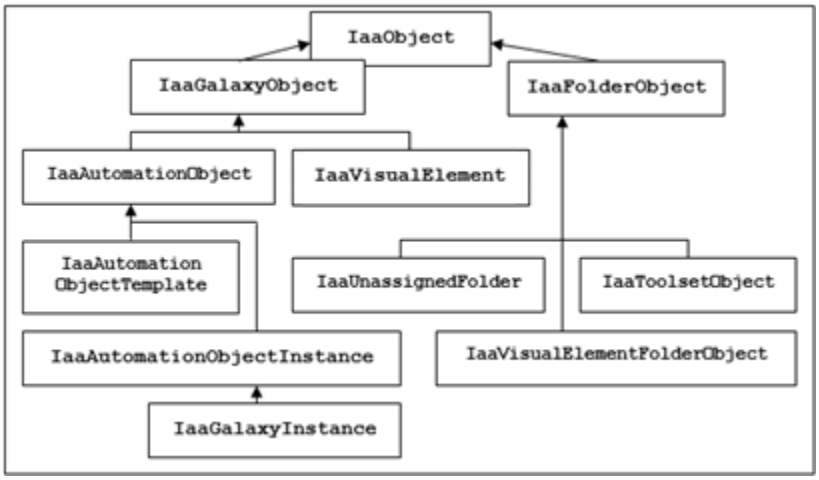
The object types provided by IDE are explained below:

- IaaObject – This is the most basic type for any kind of object that can be selected in the IDE. This is an abstract type.
- IaaGalaxyObject – This is the base type for a galaxy object (gObject). This type provides the basic information about the object via the “Id” property. This is used to distinguish between gObjects and folders (template toolsets etc).
- IaaAutomationObject – This is the base type for an automation object. This is used to distinguish between automation objects and visual elements.
- IaaAutomationObjectTemplate – This represents an automation object template.
- IaaAutomationObjectInstance – This represents an automation object instance.
- IaaGalaxyInstance – This represents the top-level galaxy object instance, which contains all other objects in the application.
- IaaFolderObject – This represents a generic container object which could be toolsets, unassigned area/host and library folders. This is an abstract type.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

- IaaUnassignedFolder – This represents the unassigned sections of the application views, which is just an UI presentation.
- IaaToolsetObject – This represents a template toolset.

It is is important that when publishing selection using these types, the type that matches the object as closely as possible is used so that the menu items and toolbar states are as accurate as possible. Following is the pictorial presentation of the interface hierarchy.



Validate, package, and deploy IDE Extensions

This section details on validation, packaging and deployment mechanism IDE extension modules or IDE extensions should use in order for them to work with the IDE seamlessly.

Validate the extension manifest

The contents of the extension manifest file (.aaPLUGIN) can be validated from inside the Visual Studio.NET IDE using the steps explained below:

1. Copy the IDE Extensions Xml Schema.xsd file to schemas folder under the Visual Studio.NET IDE installation path. Typically, this will be C:\Program Files\Microsoft Visual Studio .NET 2003\Common7\Packages\schemas\xml.
2. Select the aaPLUGIN manifest file in the project tree and right click.
3. Select “Open with…” command.
4. In the window that is shown, select HTML/XML editor.
5. Click on 'Set As Default'. This will make it open the aaPLUGIN file in XML editor by default.
6. Click on 'Open'.
7. Modify the <Root> tag to the following:

   ```
   <Root xmlns="http://wonderware.com/aaide.xsd">
   ```
8. From the XML top-level menu, select the ‘Validate XML data’ command to validate the contents. Fix errors if any.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Package IDE Extensions

IDE extensions must be packaged as strongly named .Net assemblies, built with “Microsoft Visual Studio 2003”, using “.Net framework 1.1” and signed using Wonderware provided key-pair. A .Net assembly can have one or more IDE extensions and can provide multiple manifests. On the other hand a IDE extension may be composed of multiple .Net assemblies.

Deploy the IDE Extension assemblies

IDE extensions must be deployed to the following predefined IDE folder. “[App Server Installation Path]\ArchestrA\Framework\Bin\IDEExtensions”

Remember IDE extension deployment is for the IDE node, not the GR node. Meaning, if two different IDE’s are connected to the same GR from two different machines and one of them has a newly added IDE extension then the IDE sessions from that machine would see this new extension, but the IDE sessions from the other machines would continue to see the old extensions.

When an IDE extension is composed of multiple .Net assemblies, the main assembly containing the embedded manifest (explained in the section “Getting Started with IDE Extensions”), must go to the “IDEExtensions” folder. Other assemblies belonging to the IDE extension can be installed in any folder under IDE application path, “[App Server Installation Path]\ArchestrA\Framework\Bin” or in the Global Assembly Cache (GAC).

When a developer is ready to test his IDE extension, he can just copy the extension .Net assembly to the “IDEExtensions” folder and restart the IDE. This could be iterative process to get the exact expected behavior. When the IDE extension is ready for delivery, it should be packaged into a MSI merge module, which in turn can be absorbed either into the App Server MSI install or another MSI install that checks whether IDE is installed on the machine at install time.

Test IDE Extensions

When your are ready to test your IDE extension, just copy the extension .Net assembly to the “IDEExtensions” folder and restart the IDE. This could be iterative process to get the exact expected behavior. When the IDE extension is ready for delivery, it should be packaged into a MSI merge module, which in turn can be absorbed either into the App Server MSI install or another MSI install that checks whether IDE is installed on the machine at install time.

Extend IDE specifics

This section describes IDE Extension How-To’s. It elaborates on how to extend various aspects of the IDE. This includes command extensibility, IDE view extensibility and using service pool for common functionalities. A reader can directly jump to this section if he is interested in only a specific aspect of IDE extensibility features.

Command extensibility

IDE extensions can expand the default command set of the IDE by introducing new commands and providing new command user interface vectors (menus, toolbars, keyboard accelerators and context menus), by which the end user can invoke these extended commands. The new commands and command UI vectors are integrated

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

with those provided by the IDE itself.

Add a new command

Using IDE extensibility infrastructure, IDE extensions can introduce new commands to the command set provided by the IDE. Predefined IDE commands and all new commands introduced by all of its extensions make the global pool of commands. The global command pool is available to the IDE itself and to all of the extensions for command updates and handling.

To add a new command into IDE global command pool, extension should provide command definition in its manifest resource. Remember, view extensions can introduce new commands and act as a command extension too. Following is an example extract from the sample manifest, emphasizing on how to add a new command.

```
<Snapins>
<Snapin TypeId="907162B9-F0AE-40d0-92C9-79E3DAB04D43" Priority="550">
<Resources>
<Command Id="cmdTestCheckin">
<Name>Test Check-in</Name>
<IdleUpdate>true</IdleUpdate>
<Private>false</Private>
<ContextSensitive>true</ContextSensitive>
<Description>Checks-in the object</Description>
</Command>
</Resources>
</Snapin>
</Snapins>
```

- Resources tag is where all command and all IDE user interface elements are defined.
- Command tag provides command definition. The "Id" attribute is the command identifier. Using this Id the IDE and all IDE extensions recognize this command. The command identifier must be unique amongst all commands.
- Name tag is reserved for future use when we support command UI customization.
- IdleUpdate tag dictates whether the command needs idle updates. Valid values are true and false with false being default.
- Private tag indicates whether the command can be overridden by other extensions. Valid values are true and false with false being default.
- Description tag provides status bar prompt when the command is selected.
- ContextSensitive dictates whether the command needs to be updated whenever selection in the active view changes. Valid values are true and false with false being default.

In addition to providing command definitions in the form of manifest, if it is interested in handling commands, it must register its command target with the IDE command framework, update command UI vectors and process the commands upon invocation. Please see “Handling Commands” section for detailed information.

Extend the IDE user interface

Using the IDE extensibility infrastructure, IDE extensions can introduce new command UI vectors like, menus, toolbars, context menus and keyboard shortcuts. New UI vectors can be added for the IDE commands or the commands defined by any IDE extension. They can also add new status bar panels. Newly introduced IDE UI elements merge with the ones defined by the IDE.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Extend IDE menus and menu items

IDE extensions can extend IDE menu structure by adding a new menu item, a new main menu or a menu separator with respect to an existing IDE menu/submenu. The menu definition is provided in the embedded manifest resource.

In 2023 IDE user interface is modernized to display Ribbonbar like the one used in Microsoft Office. The existing manifest file is compatible to achieve basic appearnace of a Ribbonbar though it does not fully support the customization of the Ribbonbar because the manifest was initially designed for a menu-like interface.

Adding Menu items – The following is an example extract from the sample manifest, emphasizing on how to extend IDE menus by adding new menu items. The first MenuItem tag tries to add a new item just below “Object->Check In” IDE menu and is attached to the “cmdTestCheckin” command whereas, the second one tries to add new item at the bottom of “Object” IDE menu and is attached to the “cmdTestCheckout” command.

```
<Resources>
<MenuItem Id = "Object/TestCheckin" ParentId="Object"
ReferencedTo="Object/Checkin" ReferencePosition="After">
<CommandId>cmdTestCheckin</CommandId>
<Text>Test Check-in</Text>
</MenuItem>
<MenuItem Id = "Object/TestCheckout" ParentId="Object"
DefaultPosition="Bottom">
<CommandId>cmdTestCheckout</CommandId>
<Text>Test Check-out</Text>
</MenuItem>
</Resources>
```

In 2023, the menu item is displayed as a button element inside the ribbonbar.

- MenuItem tag is where new menu item should be provided.
  - Id attribute is the identifier of the menu item. It can be any unique string, but it is recommended to use hierarchical location of the menu in IDE menu structure that would make it more readable.
  - ParentId attribute is the ID of the parent menu this menu item belongs to.
  - ReferencedTo attribute lets user dictate relative to which other menu item this item should be shown. ReferencePosition provides relative position, “Before” or “After”, defaulting to “After”.
  - DefaultPosition attribute in the second MenuItem example dictates whether the menu item should be displayed at the top or bottom amongst its siblings. DefaultPosition attribute can coexist with ReferencedTo and ReferencePosition attributes combination, in which case it will be used as fall back value when the IDE can not resolve references.
- CommandId tag is the identifier for the command the menu should be attached to. A menu item, which pops up sub-menus, should not be attached to a command. For example, “Import” menu has popup submenus and is enabled if at least one of the sub-menus is enabled.
- Text tag provides actual text of the menu item should display.

Adding Menus – IDE extensions can add new menus, which display other menu items underneath. In order to achieve this, one should add a Menu tag in the manifest resource. Menu tag does not have any associated command, since a menu is used to pop up its children menu items and not for command invocation. User just provides text for the menu. In order to add a top-level menu, the value of ParentId must be “menubarMain”. The following example shows how to add a menu entry to the Main menu bar.

```
<Resources>
<Menu Id = "SpecialImport" ParentId=" menubarMain ">
```

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

```
<Text>Special Import</Text>
</Menu>
</Resources>
```

The following example shows how to add a menu entry to the Main menu bar, and a menu item with a child menu.

```
<Resources>
<Menu Id = "SpecialImport" ParentId=" menubarMain ">
<Text>Special Import</Text>
</Menu>
<Menu Id = "ParentMenuItem" ParentId="SpecialImport">
<Text>ParentMenuItem</Text>
</Menu>
<MenuItem Id = "ParentMenuItem/ChildMenuItem"
ParentId="ParentMenuItem">
<CommandId>cmdGalaxyConfiguration</CommandId>
<Text>Galaxy Configuration</Text>
</MenuItem>
</Resources>
```

The following example shows how to add a ribbontab, a buttongroup and a buttonelement to a ribbontab respectively. Note that the buttonelement should always be inside a buttongroup.

```
<Resources>
<Menu Id = "Home" ParentId=" menubarMain ">
<Text>Home</Text>
</Menu>
<Menu Id = "Home/Edit" ParentId="Home">
<Text>Edit</Text>
</Menu>
<MenuItem Id = "Home/Edit/Tagname" ParentId="Home/Edit">
<CommandId>cmdRename</CommandId>
<Text>Rename</Text>
</MenuItem>
</Resources>
```

Adding Menu Separators – IDE extensions can also add a separator in the menu structure by adding a Separator tag in the manifest resource. Its ParentId, ReferencedTo and ReferencePosition attributes carry the same meaning as one for MenuItem tag. Separators don't need any Id attribute. Following is an example of a menu separator.

```
<Resources>
<Separator ParentId="Object" ReferencedTo="Object/ViewObjViewer"
ReferencePosition="Before"/>
</Resources>
```

Native IDE Menus – In order to add a menu item to the IDE Menus, the Parent ID must be known. The table below summarizes the ids (for programmatic reference in the ParentId field) of the menus (as viewed in the IDE):

Menu Name	Menu ID
Galaxy	Galaxy
Edit	Edit
View	View
Object	Object

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Help	Help

Extend context menus

IDE extensions can add new menu items to the IDE context menus shown in any of the IDE views including extension views. IDE has two predefined context menus; “cntxtmenuObjects” for galaxy objects and “cntxtmenuToolsets” for toolsets. If needed, extension developer can add a new context menu also. An IDE extension can add a new menu item to these context menus. Adding a new context menu item is exactly the same as explained in the previous section, the only difference is the ParentId of the context menu item must be the identifier of the context menu instead of main menu.

For example, to add a new menu item after “Open Read-Only” in the object context menu, surrounded by separators, extension needs to add the following information in the manifest resource.

```
<Resources>
<MenuItem Id="cntxtmenuObjects/TestCI"
ParentId="cntxtmenuObjects"
ReferencedTo="cntxtmenuObjects/OpenReadOnly"
ReferencePosition="After">
<CommandId>cmdTestCheckin</CommandId>
<Text>Test Check-&amp;amp;in</Text>
</MenuItem>
<Separator ParentId="cntxtmenuObjects"
ReferencedTo="cntxtmenuObjects/TestCI"
ReferencePosition="Before"/>
</Resources>
```

TODO: Need a table that lists the potential parentIds

Extend the IDE toolbar

IDE extensions can extend the IDE toolbar structure by creating a new toolbar that is displayed in the IDE toolbar, by adding a new toolbar button at the beginning or end of the existing toolbar, or by inserting a toolbar button before and after an existing button. The toolbar and toolbar definitions are provided in the embedded manifest resource. Following are two example extracts from the sample manifest, emphasizing on how to add a new IDE toolbar and toolbar button respectively.

```
<Resources>
<Toolbar Id="tbarTest">
<Name>Test</Name>
</Toolbar>
</Resources>
```

Toolbar tag is where new toolbar definition is provided by the extension. Id attribute is the unique identifier of the toolbar, which is referenced by the toolbar buttons placed in it. Name tag is used do display the toolbar list when user tries to hide/show toolbars.

```
<Resources>
<ToolbarButton Id="tbarTestCI" ParentId="tbarTest"
DefaultPosition="Top">
<CommandId>cmdTestCheckin</CommandId>
<Image>TestCI_16x16.png</Image>
</ToolbarButton>
<Separator ParentId="tbarTestCO" ReferencedTo="tbarTestCI"
ReferencePosition="Before"/>
```

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

```
<ToolbarButton Id="tbarTestCO" ParentId="tbarTest">
<CommandId>cmdTestCheckout</CommandId>
<Image>TestCO_16x16.png</Image>
</ToolbarButton>
</Resources>
```

- ToolbarButton tag is where new toolbar button definition should be provided.
  - Id attribute is the identifier of the toolbar button. It can be any unique string, but it is recommended to use identifier close to the command name is attached to for more readability.
  - ParentId is the unique identifier of the toolbar this button will be displayed in. ParentId determines whether the button will be displayed in the standard IDE toolbar or a newly added toolbar. To add a toolbar button to the standard toolbar, extension should provide “tbarStandard” as ParentId.
  - ReferencedTo attribute lets user dictate relative to which other toolbar button this button should be shown. ReferencePosition provides relative position, “Before” or “After”, defaulting to “After”.
  - DefaultPosition attribute in the first toolbar button example dictates whether the button should be displayed at the beginning or end in the toolbar. It defaults to top. DefaultPosition attribute can coexist with ReferencedTo and ReferencePosition attributes combination, in which case it will be used as fall back value when the IDE can not resolve references.
- CommandId tag is the identifier of the command the toolbar button is associated with.
- Image tag is the embedded resource image file name for the toolbar button image.
- Separator tag follows the same pattern as menu separators. The only difference is unlike menu separators, toolbar separator’s ParentId is an identifier of a toolbar.

TODO: Need a table that lists the potential parentIds

In 2023 toolbar functionality is obsolete.

Add command accelerators

The following example adds “Control + I” keyboard accelerator to the “cmdTestCheckin” command.

```
<Resources>
<Accelerator Id = "accelTestCI">
<CommandId>cmdTestCheckin</CommandId>
<Shortcut>CtrlI</Shortcut>
</Accelerator>
</Resources>
```

- Accelerator tag is where new keyboard accelerator definition should be provided. Id attribute is the unique identifier of the accelerator.
- CommandId tag is the identifier of the command the keyboard accelerator is associated with.
- Shortcut tag is the actual short cut key associated with the command. For combination with Control+Shift key use CtrlShift followed by the character as shown in the example.
- For a detailed list of supported short cuts keys, refer to the IDE Extension schema: IDE Extensions Xml Schema.xsd. In the schema document, look for the node named: ShortcutType.

Extend the IDE status bar

IDE extensions can extend IDE status bar by adding new panels to it. IDE has a predefined status bar with the

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

identifier “statusbardefault”. The following example adds a new panel to the default IDE status bar.

```
<Resources>
<StatusbarPanel Id="testSBPane" ParentId="statusbardefault">
<Sizing>Fixed</Sizing>
<Width>150</Width>
</StatusbarPanel>
</Resources>
```

- StatusbarPanel tag is where new status bar panel definition should be provided. Id attribute is the unique identifier of the status bar panel. ParentId attribute is the ID of the parent status bar, which must be “statusbardefault”, for adding a panel to the IDE status bar.
- Sizing tag dictates status bar style. Valid values are Fixed, Contents and Spring. Fixed indicates fixed sized status bar panel and it considers Width tag value for its size in terms of pixels. Contents style indicates that the status bar panel acquires size as much as its contents. Spring style indicates it occupies rest of the area in the status bar. Contents and Spring styles ignore Width tag value.

Override a command

The loading order of IDE extensions is determined by the “Priority” attribute value provided by the extensions in its manifest. Priority ranges from 1 through 999 and higher the priority number, earlier the extension is loaded by the IDE. Following is an example extract from the sample manifest, emphasizing the priority of an IDE extension.

```
<Snapin TypeId="907162B9-F0AE-40d0-92C9-79E3DAB04D43" Priority="550">
```

In addition to introducing new commands into the IDE, extensions can also choose to override the behavior of the commands that are defined by the IDE itself and those by other extensions. Overriding a command means changing the behavior of a command and not changing the definition of the actual command. In order to override a command, the extension should make sure that it is loaded after other extensions that are interested in the same command, by providing lower priority. Extensions loaded later have higher preference than the ones loaded before, when it comes to getting a chance at handling commands.

Now, in order to actually override a command behavior (defined by the IDE or other extension) the extension should register the command target, handle command updates and command invocations just the same way as it handles its own commands. The only difference is that it registers the command target for the command, which is defined by the IDE or some other extension instead of its own command. Please see “Handling Commands” section for detailed information on how to handle a command, irrespective of whether the command is defined by the same extension, by the IDE or some other extension.

View extensibility

Please see section “Creating a View Extension” for detailed information on how to provide view extensions and the sample IDE extension project provided with this document for actual implementation of a sample view. Remember, a view extension can also act as a command extension and supply its own commands, command UI vectors and so on.

IDE services

A service is any reusable functionality provided by the IDE or an extension module, which can be shared by all extension modules. The IDE provides some default services to facilitate extension development. An IDE

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

extension should register its service with the service pool in order to make it available to the IDE and its extensions. Once a service is registered, it will be available for consumption.

As explained in the section “Creating a Command Extension”, all IDE extensions must implement “IaaSnapin” interface. IDE calls the extension’s “Initialize” method implementation in this interface and sets its “`IaaServiceProvider`” site, when the extension is loaded. The extension can use the site to access global service pool. The “`IaaServiceProvider`” interface has three methods for registering, un-registering and accessing services, two events to subscribers when a service is added or removed; and one indexer for ease of usage as explained below.

- **RegisterService –** The method allows extension register its service to the IDE and makes it available in the global service pool.
- **UnregisterService –** The method allows extension to un-register its service and pulls it out of the global service pool.
- **GetService –** The method allows extension query for a specific service from the IDE global service pool.
- **ServiceAdded –** The event to notify subscribers when a service is added to the service pool.
- **ServiceRemoved –** The event to notify subscribers when a service is removed from the service pool.
- **Services –** The indexer is alternative to “GetService” method with ease of usage.

The following is an example of how a typical extension would register its service with the IDE and make it available in the global service pool.

```
[aaPlugin("907162B9-F0AE-40d0-92C9-79E3DAB04D43")]
public class MyClass : IaaSnapin, IaaCommandTarget, INewInterface
{
// TODO: Class, IaaCommandTarget and INewInterface implementation.
void IaaSnapin.Initialize(IaaServiceProvider site)
{
if(site != null)
{
m_site = site;
m_site.RegisterService(typeof(INewInterface), this);
}
}
}
```

The following is an example of how another extension would consume a service registered by the previous extension.

```
INewInterface mi = m_site.Services[typeof(INewInterface)] as INewInterface;
if(mi != null)
{
// TODO: Use INewInterface service.
}
```

ArchestrA/AVEVA configuration services

IDE extensions can utilize IDE published galaxy configuration database related services to browse and configure galaxy, find out information about individual galaxy objects, persist user preferences and so on. IDE itself and all System Platform extensions also use these services internally.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Galaxy configuration

IDE publishes a database related service called “IGalaxyConfiguration” to allow extensions query and configure galaxy database programmatically. It can be accessed from the service pool. Using this service one can perform various galaxy operations like, creating object instances, deriving templates, deleting template/ instances, renaming template/instances, deploying instances, getting updates upon galaxy changes and so on.

The service also provides functionality for the extensions to store and retrieve information data per galaxy and per user basis. Extensions can leverage this mechanism to store/retrieve its own specific data like, last user selected object, view status and so on.

Following is an example of how a typical extension would save/retrieve user preferences to/from the galaxy.

```
IGalaxyConfiguration gc=m_site.Services[typeof(IGalaxyConfiguration)]
as IGalaxyConfiguration;
if(gc != null)
{
// Retrieve user preferences for this extension.
byte [] buffer = gc.GetUserPreference(this.ToString());
// TODO: Make some changes to the user preferences.
// Store user preferences for this extension.
gc.SetUserPreference(this.ToString(), buffer);
}
```

Galaxy object information

IDE provides another useful functionality in the form of a light weight interface called "IGObjectInfo". For a given galaxy object this interface allows read-only access to its often used attributes like, its TagName, whether it is a template, whether it is checked out, toolset id of a template, whether instance is deployed, its category id, whether instance has any pending updates and so on.

In order to get to this interface, the extension should first query service pool for "IGalaxyConfiguration" service first. Then it should call "GetGObjectInfoFromID" method on the galaxy configuration interface. Following is an example of a typical "IGObjectInfo" interface usage.

```
private string GetTagNameForID(int id)
{
int goId = id;
string strTagName = "";
IGalaxyConfiguration gc =
m_site.Services[typeof(IGalaxyConfiguration)]as IGalaxyConfiguration;
if(gc != null)
{
// Get IGObjectInfo for the id.
IGObjectInfo goInfo = gc.GetGObjectInfoFromID(ref goId);
// Get the TagName for this object.
if(goInfo != null)
strTagName = goInfo.TagName;
}
return strTagName;
}
```

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

User interface services

IDE provides various predefined user interface related services, which are being used by the native IDE and Wonderware extensions. They should be used by all IDE extensions to avoid duplication of work, to provide uniform look and feel and for seamless integration with the IDE. This includes command framework related services, selection service, progress notification service, object icon service and so on.

Register and invoke commands

In order for an IDE extension to handle a command, the extension public class must implement "IaaCommandTarget" interface from the "Extensibility" assembly.

The first thing extensions need to do is to register the command target for the command it is interested in. Using the command provider site the extension can query for "IaaCommands" service and register its command target by calling "Register" method on this interface. In the same manner, extensions can un-register command target for a command by calling "Unregister" method on the same interface. Typically extensions register their command targets in "IaaSnapin.Initialize" method, whereas un-register them in "IaaSnapin.Uninitialize" method. The following is how the sample extension registers its commands.

```
void IaaSnapin.Initialize(IaaServiceProvider site)
{
m_site = site;
IaaCommands cmds = m_site.Services[typeof(IaaCommands)] as
IaaCommands;
// This class itself is the command target
cmds["cmdTestCheckin"].Register(this);
cmds["cmdTestCheckout"].Register(this);
}
```

After the command target is registered, the IDE uses this interface to communicate with the extensions to update command UI vectors and to process invoked commands. The "IaaCommandTarget" interface has two methods as explained below.

- **OnUpdate –** The IDE calls this method to give command target a chance to update command and command UI vectors. If the command is context sensitive, the method is called every time a selection in the active view changes.
- **OnInvoke –** The IDE calls this method to give command target chance to process the command when it is invoked.

When calling "OnUpdate" or "OnInvoke" methods, IDE passes an instance of command being processed in the form of "IaaCommandInstance" interface. The IDE Extension can probe this interface to get handle to the actual command, command arguments, what command UI this command is invoked through, and states of the command. Following is a sample "OnUpdate" implementation, which changes command and various command UI states.

```
public void OnUpdate(IaaCommandInstance cmd)
{
// Applicable command state property indicates, whether it should
// be shown/hidden when context menu is invoked.
cmd.State.Applicable = true;
cmd.State.Enabled = true; // Indicates command is enabled.
cmd.State.Checked = true; // Indicates command is checked.
```

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

```
}
```

If there is more than one extension that registers for a command, IDE framework uses the following order to determine the extension, which will be notified for command update and invocation.

- The active view extension the user is currently working with.
- Any extension that overrides the command.
- The extension that defined the command.

Please see the sample example extension for command handling command updates and invocation in detail.

Invoke/update other commands

The IDE provides “IaaCommands” and “IaaCommand” interfaces to facilitate extensions get access to any command available in its command set. Extension can query service pool to get “IaaCommands” and call its indexer to get actual command. This way extension can update/invoke any command by calling “Invoke” and “Update” helper methods respectively.

The following is an example of how to invoke another command. This is a simple command invocation and it does not pass any context. For mode detailed sample example extension provided. Updating another command works the same way.

```
private void InvokeAnotherCommand()
{
IaaCommands cmds = m_site.Services[typeof(IaaCommands)] as
IaaCommands;
IaaCommand cmdNew = cmds["cmdNew"];
if(cmdNew != null)
{
// IaaCommandContext concept is explained in the next section.
IaaCommandContext context = null;
cmdNew.Invoke(context, null); // Invoke cmdNew command.
}
}
```

Status bar service

The IDE provides a service called “IaaStatusbar” to allow its extensions share the same status bar. Extensions can use this service enumerate status bar panels, to show text/images in the panels and subscribe to panel’s event to get notification when it is double clicked. IDE extension can add its own panels into the status bar and access them using the status bar service. Following is an example of how extensions would use this functionality.

```
private void WriteStatusBarText(string strText)
{
IaaStatusbar statusbar = m_site.Services[typeof(IaaStatusbar)]
as IaaStatusbar;
if(statusbar != null)
{
IaaStatusPanel pane = statusbar.Panels["usernamepane"];
pane.Text = strText;
}
```

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

```
}
```

General UI service

The IDE provides a convenient and uniform means for all extensions to get the IDE font and caption, in the form of a service called “IaaGeneralUIService”. It has two properties as explained below.

- **IDEFont –** All IDE extensions should use this property to have uniform font all cross IDE, which gives seamless look and feel.
- **IDECaption –** All IDE extensions use this caption for all message boxes and dialogs, which provides “ArchestrA IDE” as the title. This gives IDE centralized control over caption of all message boxes and dialogs.

Progress notification service

User may perform various time consuming bulk operations from within the IDE like, importing objects, deploying instances, deleting objects and so on. While these operations are performed, IDE extensions should show the progress of operations in uniform fashion. IDE provides service called “IProgressControl” to facilitate uniform progress notification that provides operation progress in messages and percentage. It also allows user to cancel bulk operation in progress.

Please see the sample example extension for usage of progress notification service.

Object icon service

The IDE provides a convenient and uniform means for all extensions to get AutomationObject icon and its superimposed object status icons for a given galaxy object, in the form of a service called “IaaImageService”. Superimposed object status icons include, error/warning and various deploy related statuses. Extensions can also get object checked out status icons separately. Following is an example of how extensions would use this functionality.

```
private Image GetObjectImage(IGObjectInfo goInfo, ref Image imgExt)
{
Image imgObject = null;
IaaImageService imgsvc =
m_site.Services[typeof(IaaImageService)] as IaaImageService;
if(imgsvc != null && goInfo != null)
{
string strImageId;
imgObject = imgsvc.GetImage(goInfo, out strImageId, true);
if(imgExt != null)
imgExt =
imgsvc.GetExtendedEffect(ExtendedEffect.CheckedOutByMe);
}
return imgObject;
}
```

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Show context menus

When user tries to show context menu with right mouse click in an extension view, depending on context it might show object context menu or its own context menu (defined by itself in the manifest). IDE facilitates this functionality for the extensions. Following are two examples, one showing object context menu and other showing extension defined context menu. These methods should be invoked in the MouseUp event handler when right mouse button is clicked.

```
private void OnShowContextMenu()
{
IaaMenu menu = m_site.Services[typeof(IaaMenu)] as IaaMenu;
// Display object context menu
if(menu != null)
menu.ShowPopup ();
}
private void OnShowContextMenu()
{
IaaMenus menus = m_site.Services[typeof(IaaMenus)] as IaaMenus;
if(menus != null)
{
IaaMenu menu = menus[“MyContextMenu”];
// Display extension’s context menu
if(menu != null)
menu.ShowPopup();
}
}
```

Enumerate and activate IDE views

Because of extensible architecture of the IDE, its extensions do not really know about each other and ideally they should be able to work without being tightly coupled with each other. This might necessitate extensions to know what other IDE views are available and to activate a specific one of its interest programmatically. The IDE provides functionality in the form of a service to enumerate all IDE views and activate a specific one. Wonderware extensions use this functionality internally for various operations like create instance, create derived template, rename AutomationObject and so on. Following is an example of how an extension can leverage this service and find out a view that support in-place renaming and activate it.

```
private bool ActivateRightView()
{
bool bCanInPlaceRename = false;
IaaPluginWindows windows =
m_site.Services[typeof(IaaPluginWindows)] as
IaaPluginWindows;
if(windows != null)
{
IaaPluginWindow activeWindow = windows.ActiveWindow;
if(activeWindow != null)
{
IaaIDEView ideView = activeWindow.Window as IaaIDEView;
if(ideView != null)
{
bCanInPlaceRename = ideView.CanInPlaceRename();
```

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

```
if(bCanInPlaceRename)
{
activeWindow.Activate();
}
else
{
foreach(IaaPluginWindow wnd in windows)
{
IaaIDEView view = wnd.Window as IaaIDEView;
if(view != null)
{
bCanInPlaceRename = view.CanInPlaceRename();
if(bCanInPlaceRename)
{
wnd.Activate();
break;
}
}
}
}
}
}
}
return bCanInPlaceRename;
}
```

Tips for IDE Extension development

This section points out common pitfalls an IDE extension developer should avoid while developing an extension and recommends some optimizations.

- The extension class (adorned with "`aaPluginAttribute`" custom attribute) must be public, otherwise the IDE won't be able to instantiate it.
- The command target class being registered for a command (the class, which implements "`IaaCommandTarget`" interface) should be extension public class, otherwise command routing will be indeterminist.
- Ensure "`IaaIDEView.SelectObject`" method implementation checks for "`NameSpace`" property of the object when a galaxy object is requested.
- If an extension does not register to "`ServiceAdded`" and "`ServiceRemoved`" events from the "`IaaServiceProvider`" interface then it must not cache queried service pointer, since service could un-register itself from the pool in the meantime.
- Service pointer must be checked for non-null value after being queried from the service pool.
- Minimize processing (especially any UI processing) done in the "`IaaSnapin.Uninitialize`" method implementation, since the IDE might be shutting down during the call.
- Ensure "`IaaSelectionSource.SelectedObjects`" property getter implementation is as optimized as possible, since it is called upon every single selection change.
- When publishing objects in the selection service, providing common base type closest to the selected objects would optimize performance.
- When updating context sensitive commands when multiple objects are selected, check the common base

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

type and not the actual objects for better performance.

- General rules for command and command UI vector updates mentioned in "*General Configuration and IDE DFS.doc*" (FR89), related to enabling, disabling, showing and hiding UI vectors should be followed by all IDE extensions for uniformity.
- In order to add a new command and its corresponding command UI vectors like a menu item or a toolbar button, developer should follow steps mentioned below.
  - Add a new command in the extension's manifest file as described in the section "How to Add a New Command".
  - Add a new menu item in the extension's manifest file as described in the section "How to Extend IDE Menus". Please refer to the sections "How to Extend IDE Toolbar" to add a new toolbar button and the section "How to Extend Context Menus" to add a new context menu item.
  - Add code to register/un-register command target. Typically extensions register their command targets in "`IaaSnapin.Initialize`", whereas un-register them in "`IaaSnapin.Uninitialize`" method implementations.
  - Add code for the command target by implementing "`IaaCommandTarget`" interface, which has two methods, "`OnUpdate`" and "`OnInvoke`". These methods are called by the IDE to give extension chance to update and handle command invocation respectively. Please see section "Handling Commands" for more detailed information on how to register and handle commands.
- Adding a menu separator – Extension developer can add a separator to main menu or to the context menu by adding "Separator" tag in the manifest resource file. Following is an example of a menu separator.

```
<Resources>
<Separator ParentId="Object" ReferencedTo="Object/ViewObjViewer"
ReferencePosition="Before"/>
</Resources>
```

Its ParentId attribute indicates the menu the separator should be added to and for context menu it should be context menu id. ReferencedTo and ReferencePosition attributes dictate relative position of the separator with respect to some other menu item.

- Adding a top-level menu – In order to add a top-level menu, extensions developer should add "Menu" tag in the manifest resource file, wherein the "ParentId" attribute value must be "menubarMain" as shown in the following example.

```
<Resources>
<Menu Id = "SpecialImport" ParentId="Galaxy">
<Text>Special I&amp;mport</Text>
</Menu>
</Resources>
```

Override predefined IDE commands

Following is the list of IDE commands segregated based on whether they can be overridden by its extensions. When overriding an existing command, the command does not have to be redefined in the xml manifesto. The command override has only to be added to source code.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

Commands that can be overridden

The following predefined commands can be overridden by the IDE extensions.

- cmdNewInstance – Create an instance from the selected template.
- cmdNewTemplate – Create a derived template from the selected template.
- cmdNewToolset – Creates a new template toolset.
- cmdOpen – Open the editor for currently selected galaxy object.
- cmdOpenReadOnly – Open the editor for currently selected galaxy object in read-only de.
- cmdImportObject – Import Automation Object package(s) containing object configuration data and code modules.
- cmdImportScriptLib – Import a .NET code module or an .aaSLIB file for use in Automation Object Scripting.
- cmdGalaxyDump – Dump Instances to a (.csv) file.
- cmdExportObject – Export Automation Object package(s) containing object configuration data and code modules.
- cmdExportAllObjects – Export All Automation Objects in the Galaxy to a package containing object configuration data and code modules.
- cmdExportScriptLib – Export a previously imported Script Function Library to an .aaSLIB file.
- cmdGalaxyLoad – Load a Galaxy Dump (.csv) file.
- cmdEditSecurity – Configure the Galaxy security model.
- cmdViewSecurity – View the Galaxy security model.
- cmdConfigureTimeMaster – Configure the Time Master for the Galaxy.
- cmdCustomizeToolsets – Customize which toolsets are shown and hidden.
- cmdGalaxyStatus – Displays the status of the current Galaxy.
- cmdProperties - Show the Properties dialog for the currently selected galaxy object.
- cmdChangeGalaxy – Allow user to switch Galaxies.
- cmdChangeUser – Allow user to switch to a different user login.
- cmdCopy – Copy the current selection.
- cmdRename – Rename the current selection.
- cmdRenameContainedName – Rename the contained name of the object.
- cmdDelete – Delete the current selection.
- cmdFind – Find objects in the Galaxy.
- cmdUserInfo – Enables the user to view and modify their user information.
- cmdCheckout – Checks out the currently selected object(s).
- cmdCheckin – Checks in the currently selected object(s).
- cmdUnCheckout – Undo the check out of the currently selected object(s).
- cmdOverrideCheckout – Override the Check Out of another user for the currently selected object(s).
- cmdValidate – Validate the selected object(s).
- cmdViewObjViewer – View the object in the Runtime Object Viewer.
- cmdDeploy – Deploy the selected object(s).

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.

- cmdUndeploy – Undeploy the selected object(s).
- cmdAssignTo – Assign the selected object(s) to another object.
- cmdUnassign – Unassign the selected object(s).
- cmdSetDefault – Set the current object as the user's default for assignment, containment, etc.
- cmdUploadRTChanges – Upload the changes from the runtime to the Galaxy.
- cmdHideToolset – Hide template toolsets.
- cmdLicenseStatus – Display the license status of the current Galaxy.
- cmdHelpTopics – Launch the IDE Help.
- cmdEditorObjectHelp – Launches object-specific Help.
- cmdHelpAbout – Display information about the ArchestrA IDE.

Commands that cannot be overriden

The following predefined commands can not be overridden by the IDE extensions and are marked as private.

- cmdValidateUser – Requires the user to log-in again. This command does not have any UI vector and is used internally when security model is changed.
- cmdExit – Signs off and exits IDE application.
- cmdViewStatusbar – Toggle the visibility state of the status bar.
- cmdSyncViews – Synchronize object selection of all visible views.
- cmdGotoPartner – Switches to the redundant Application Engine partner.

© 2015-2024 AVEVA Group Limited and its subsidiaries. All rights reserved.



AVEVA Group plc
High Cross
Madingley Road
Cambridge
CB3 0HB
UK

Tel +44 (0)1223 556655

www.aveva.com
To find your local AVEVA office, visit **www.aveva.com/offices**

AVEVA believes the information in this publication is correct as of its publication date. As part of continued product development, such information is subject to change without prior notice and is related to the current software release. AVEVA is not responsible for any inadvertent errors. All product names mentioned are the trademarks of their respective holders.