



AVEVA™ Gateway for SmartPlant 3D

6.0

© 2015-2025 AVEVA Group Limited or 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/>

Publication date: Tuesday, April 15, 2025

Publication ID: 1526537

Contact information

AVEVA Group Limited
High Cross
Maddingley Road
Cambridge
CB3 0HB. UK

<https://sw.aveva.com/>

For information on how to contact sales and customer training, see <https://sw.aveva.com/contact>.

For information on how to contact technical support, see <https://sw.aveva.com/support>.

To access the AVEVA Knowledge and Support center, visit <https://softwaresupport.aveva.com>.

Contents

- Chapter 1 What is new in AVEVA Gateway for SmartPlant 3D?..... 5**
 - Gateway for SmartPlant 3D 6.0.0..... 5**
 - Introduction..... 5
 - Content..... 5
 - Prerequisite for this release - Operating System..... 5
 - Prerequisite for this release - Products..... 6
 - Works (is compatible) with..... 6
 - Supersedes..... 6
 - To Install Product Release..... 6
 - List of Enhancements..... 6
 - List of Fault Corrections..... 7
 - Product Quality Statement..... 7
 - Product Quality..... 7
 - Exceptions – Significant Defects And Recommended Workarounds..... 7
 - Known Issues..... 7
- Gateway for SmartPlant 3D 6.0..... 8**
 - Get Started..... 10**
 - Hardware Requirements..... 10
 - Software Requirements..... 10
 - Security Guidelines..... 11
 - Install the Gateway..... 12
 - Access the Gateway..... 12
 - Uninstall the Gateway..... 13
 - Run the Gateway from the Project Explorer..... 14**
 - Create Projects..... 14
 - Open Projects..... 15
 - Define the Project Settings..... 16
 - General Settings..... 17
 - Extract Settings..... 21
 - Transform Settings..... 32
 - Load Settings..... 35
 - Execute Projects..... 51
 - Project Version Upgrade..... 52
 - Access an AWS S3 Bucket..... 53
 - Access AVEVA Cloud Storage..... 55
 - Run the Gateway from the Command Line..... 60**

Status Messages from the Gateway Processing	67
Appendix A: Mapping Configuration	70
LookupDataSource	70
Deriving Object ID and Class ID from Lookup	73
Default Value Mapping in Attribute	74
TemplateLookups	75
Transforms	76
Regular Expression Evaluation Timeout	77
Timestamp References	78
AutoNumber References	78
InternalId References	79
Manifest References	79
Migrate Oledb to ODBC	82
Base Mapping Types	85
Object Mapping	85
Attribute Mapping	87
Dataset Mapping	93
Association Mapping	94
ObjectID Mapping	100
ClassID Mapping	102
Context Mapping	103
ObjectName Mapping	105
RevisionID Mapping	106
Incidental Classification Mapping	107
Search Criteria Mapping	108
Expand Attribute Mapping	110
3D Model Hierarchies	111
Presentation Mapping	111
Appendix B: CSV Reporting	114
Appendix C: Tracking Changes in Data	119
Appendix D: RegEx Utility	119
Appendix E: Encrypt Utility	122
Appendix F: Upgrade from the Gateway Configuration Tool Project Settings	123
Appendix G: Reserved Attributes Used in the Gateway	136
Appendix H: Handle System Attributes	137
Appendix I: Third-Party Software Dependency	138
Appendix J: Limitations of the Gateway	138

Chapter 1

What is new in AVEVA Gateway for SmartPlant 3D?

These release notes explain the feature updates, new enhancements and known issues of AVEVA Gateway for SmartPlant 3D. They are usually released as part of a major product release.

This section describes the release notes for the AVEVA Gateway for SmartPlant 3D.

Gateway for SmartPlant 3D 6.0.0

A full release of AVEVA NET Gateway for SmartPlant 3D 6.0.0, which replaces the full release 71378 AVEVA NET Gateway for SmartPlant 5.0.12.

The product installable by this release is as follows:

Product Code	Product Name	Version Number
V00FN796	AVEVA Gateway for SmartPlant 3D	6.0.0

Introduction

This Release Note about AVEVA Gateway for Smart Plant 3D describes about the enhancements, fault corrections, prerequisites, and so on.

Content

Prerequisite for this release - Operating System

The Operating systems prerequisites for the Gateway are listed below:

Prerequisites	Version No.
Minimum Supported O/S Version (for each supported platform)	Microsoft Windows Server 2016 (64-bit) Microsoft Windows Server 2019 (64-bit)
Mandatory O/S Patches (for each supported platform)	Not Applicable

Note: The recommended/supported hardware and software configurations are constantly subject to review, so please consult the AVEVA support (<https://softwaresupport.aveva.com>) for the latest recommendations.

Prerequisite for this release - Products

The prerequisites for this Gateway release are as follows:

- Windows Server 2016, 2019 or 2022 (64-bit)
- **.NET Framework 4.8**

Note: Neither the **AVEVA Licensing System** nor **LaaS** are required for this version.

Works (is compatible) with

- AVEVA NET Workhub & Dashboard 5.1.8 and later versions of AIM
- SmartPlant 3D 2011R1
- SmartPlant 3D 2014 R1
- Smart 3D 2016 R1
- Smart 3D 2018
- Microsoft.ACE.OLEDB.12.0 *
- Microsoft SQL Server (64-bit) 2014, 2016 *
- Oracle database 12c Release 3 (12.1.0.2.0) *

Note: * Connections to these files and databases are only required if they are being used for lookups

Supercedes

This release supersedes the previous full release 71378 AVEVA NET Gateway for SmartPlant 3D 5.0.12. As the functions in this new Gateway now fulfil all previous functions in the legacy AVEVA NET Gateway for SmartPlant 3D 5.0.12, the legacy Gateway is now deprecated and will be in bug support only for the next 12 months or the next release of the new Gateway, whichever is shortest. Due to the difference in transformation syntax, there is no automated migration from the legacy configuration to the new configuration.

To Install Product Release

Please read the installation section of the user guide or contact local support.

Note: If the beta version of the Gateway is installed, please ensure that it is uninstalled before installing this 6.0.0 version.

First released on:

This product release is first available as AVEVA NET Gateway for SmartPlant 3D 6.0.0 release number 71415.

List of Enhancements

This is the first release of the Gateway, it's full set of functions are detailed in the User Guide and summarized in the AVEVA NET Gateway for SmartPlant 3D User Bulletin 6.0.0.

List of Fault Corrections

Not applicable as this is the first release, but this new Gateway provides a significant improvement in data extraction speeds compared to the legacy AVEVA NET Gateway for SmartPlant 3D 5.0.12.

Product Quality Statement

The development, maintenance and testing of AVEVA Group's software products are carried out in accordance with the AVEVA Quality Management System lifecycle processes and this document includes a summary of the testing carried out on this release and a list of significant defects known to exist at the time of release.

Product Quality

Development Testing

Developers have carried out unit testing of each enhancement and/or fix in the development environment to verify that any new functionalities have been correctly implemented and identified defects have been corrected.

Regression tests have been run in the development environment to verify that enhancements and/or fixes have not adversely affected the integrity of the product.

System Testing

Independent system testing has been carried out on this product, as released, to verify that it installs correctly in all supported configurations, and that product functionality operates as intended, subject to any known exceptions listed below.

Acceptance Testing

AVEVA's Product Readiness Authority (PRA) is satisfied that the System Testing for this release is sufficient to assure product quality.

Any exceptions found during Acceptance Testing or after release will be reported in the Product Release Latest Update Note.

Exceptions – Significant Defects And Recommended Workarounds

None.

For the latest list of exceptions and other updates, including those for the original full release, see the Product Release Latest Update Note on AVEVA's support web site .

Known Issues

None.

For the original full release, latest list of exceptions and other updates, please see the Product Release Latest Update Note. For details please visit .

Gateway for SmartPlant 3D 6.0

The *AVEVA NET Gateway for SmartPlant 3D* (henceforth, referred to as the Gateway) is a type of CAD File processing gateway, which converts a 3D model and its associated engineering data from a Smart 3D project into a format that can be imported into [AIM](#) or [AIM-A](#). The 3D model is converted into XGL/ZGL format and the associated engineering data is converted into EIWM XML or CSV.

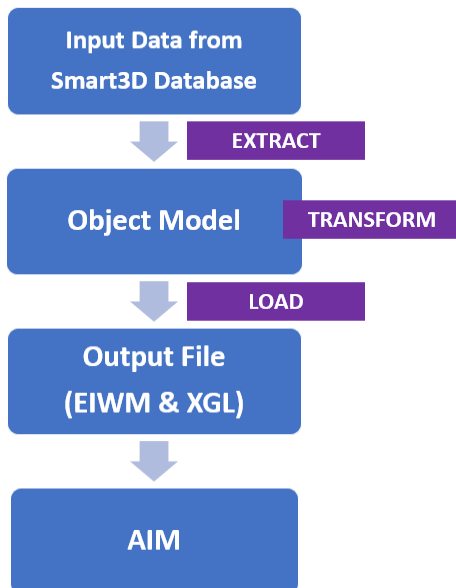
The Gateway provides a user interface with a set of configuration settings and data mappings that are saved as projects. When a Smart 3D project is to be created, you define the required extraction configuration settings to choose the source Smart 3D project and its components, and edit the transformation mapping file(s) to define the Engineering attributes which will be imported into AIM, such as tag and class values.

You can run the Gateway either directly through the user interface or from the command line. When processing is complete, the extracted XGL/ZGL and EIWM XML files are loaded into *AIM* using the Import Controller or into *AIM-A* using the Gateway Data Publisher.

Benefits of the Gateway:

The following are the benefits of the Gateway:

- Connects with the Smart 3D database to extract the information.
- Transforms extracted data from Smart 3D input data through XML base mappings. Results of each transformation can be exported to a .CSV file for inspection.
- Supports Load (output) of objects and attributes to EIWM and graphics to XGL, suitable for *AIM*.
- Supports creation of XGL/ZGL file.
- Provides a summary report of the process.



High Level Diagram of the AVEVA NET Gateway for SmartPlant 3D

This guide explains how to install and use the Gateway. It includes the following key information:

- Extracting the input data from Smart 3D model
- Transforming the extracted data
- Defining Project settings and mappings and generating output files.
- Running the Gateway from the command line and from user interface.
- Exporting of transformed data into AVEVA™ *Asset Information Management* (previously known as AVEVA NET)

This guide is intended for those users who will be using the Gateway to convert various Smart 3D files containing project, engineering and 3D model data into intelligent EIWM XML format and 3D renditions as XGL.

Users should have the following pre-requisite skills to set up, maintain and monitor the Gateway:

- Configuration of the Smart 3D in use.
- SQL expression construction and XML data sources.
- Configuring and using AVEVA *Asset Information Management* import functions.

Chapter 1 Get Started

This chapter details the procedures required to install and run the Gateway. It also describes the hardware and software requirements to install the Gateway.

Hardware Requirements

You must meet the following hardware requirements before you install the Gateway:

Component	Minimum	Recommended
Memory	32 GB	64 GB
Hard Disk Drive	80 GB	
CPU	Intel® Xeon® processor, 3 GHz, Dual core	Intel® Xeon® processor, 3 GHz, Quad core

Usage of RAM

The processing of large or complex files can consume all of the memory (RAM) of the server hosting the Gateway, which impacts other processes running on that server. To avoid this, the Gateway enables a maximum RAM consumption to be set as a percentage of the total, which can be specified in the Project XML file. For example, if the **restrictMemoryForProcess** is set to 80% and the server has 16 GB of RAM, then the Gateway process consumes less than 12.8 GB of RAM. If the **restrictMemoryForProcess** is set to 100%, then no checks are applied to limit the amount of RAM consumed by the Gateway process. The default value is 90%.

Note: To improve performance, use the fastest drive possible for extraction, for example, a Solid State Drive and ensure it has enough capacity to store the model.

Software Requirements

You must meet the following software requirements before you install the Gateway:

Application	Supported Version
.NET Desktop Runtime	x64 8.0.0 or later x86 8.0.0 or later (both x64 and x86 are required.)
Windows Server	2016 (64-bit), 2019 (64-bit)
Visual C++ Redistributable Packages	Microsoft Visual C++ Redistributable 2015-2019 (64 bit)
Access Database Engine (ADE)	2010 (64-bit), 2016 (64-bit) Notes: <ul style="list-style-type: none">When you install ADE, this installs required default drivers for ODBC.

	<ul style="list-style-type: none"> • ADE is only needed for using lookups in mapping.
--	--

Notes:

- The Gateway supports *AVEVA Asset Information Management (AIM)* version 5.1.12.
- AVEVA Licensing System (ALS) and Licensing as a Service (LaaS) are no longer needed.

The following table lists the application and their versions, which are supported by the Gateway:

Application Supported	Supported Version
<i>Hexagon Intergraph SmartPlant® 3D</i>	2011 R1
<i>Hexagon Intergraph SmartPlant® 3D</i>	2014 R1
<i>Hexagon Intergraph Smart® 3D</i>	2016 R1
<i>Hexagon Intergraph Smart® 3D</i>	2018
<i>Hexagon Intergraph Smart® 3D</i>	2019

Lookups requiring Access Database Engine

When using lookups to Excel/Access files, the Gateway reads these files using the Microsoft ADE. For more information about Lookups, see [LookupDataSource](#).

You must first install the Microsoft ADE which can be downloaded from:

<https://www.microsoft.com/en-us/download/details.aspx?id=54920>

The Microsoft ADE 2010 (64-bit) is compatible with Microsoft Office 2010 (64-bit), Microsoft Office 2013 (64-bit) versions.

If you have installed Microsoft Office 2010/2013 x86 (32-bit), this blocks the installation of the Microsoft ADE (64-bit). You need to perform the following to install the Microsoft Access Database Engine (64-bit):

1. Uninstall Microsoft Office 2010/2013 x86 (32-bit) from your system.
2. Install Microsoft Office 2010/2013 (64-bit) in your system.
3. Install Microsoft ADE (64-bit).

Security Guidelines

The Gateway processes and converts data from multiple data sources into a compatible file format, which is loaded into *AIM*. It therefore needs to read data from servers and write the transformed data into files and folders, for which the following security best practices are recommended:

- Use the principle of least privilege:
 - Grant the user account that is used to run the Gateway only read access. Grant write or update access only to the specific files and folders it needs to modify. For example, within a project folder, you can grant:
 - Read-only access to Input folder
 - Write access to Log folder

- Modify access to Output folder, Configuration/Mappings folder and Project file.
- Restrict the database user account with the read-only access to the databases (or the selected tables/views) from which information needs to be read as lookup entries during transformation. It is required to avoid accidental addition, change or deletion of sensitive data.
- In AWS, restrict a user's access to only those AWS resources required by their IAMrole. For example, as the Gateway uses only S3 buckets, you can restrict the IAMrole to access only S3 bucket. You can also define a bucket policy to restrict access to an S3 bucket.
- If the Gateway is configured to read data from an Oracle or an SQL Server database, for example, in a lookup, then to secure the data which is in transit, you can configure an SSL-encrypted connection between the Gateway and the database.
- You do not need to adjust your Firewall settings or User Account Control settings when you install or use the Gateway.

Notes:

- If the security recommendations are not suitable for your environment, you must investigate what is the most suitable approach for your environment and apply those practices.
- If AVEVA Cloud Storage (ACS) is used for output file locations, then ensure that the Transport Layer Security (TLS 1.2) option is selected. Perform the following recommended steps:
 - a. Navigate to **Internet Options** and then **Advanced** tab.
 - b. Under **Security** section, select the option "**Use TLS 1.2**".
 - c. Click **Apply**, and then **OK**.
- If your data needs to comply with local legislation related to protecting personal information, such as GDPR in EU countries, then be aware of any personal information such as e-mail or home addresses that may be in the input source. Either filter them out via transformation mappings or ensure the location and management of the EIWM output file comply with local legislation.

Install the Gateway

You must ensure that the hardware and software requirements described in the previous sections are satisfied before you install the Gateway.

To install the Gateway

1. Run the **Smart3DGatewayInstaller.msi** and follow the on-screen instructions.

Note: You must install the Gateway on the machine that is hosting Smart 3D. The Gateway requires an active Smart 3D license to process the Smart 3D project.

Command line installation

1. Type the name of installer file followed by `/?` to see all options.

Example of Quiet installation: **Smart3DGatewayInstaller.msi /quiet**

Access the Gateway

You can access the Gateway in following two ways:

- Select **AVEVA>NET>AVEVA NET Gateway for SmartPlant 3D** from the **Programs** menu

- Double-click the **AVEVA NET Gateway for SmartPlant 3D 6.1** desktop shortcut.

This loads the Gateway main dialog. This dialog helps create, configure, and run the Gateway projects.

Uninstall the Gateway

We recommend you to remove any previously installed versions of the Gateway from your system before installing a new version.

You can select any of the following ways to uninstall the Gateway:

- Using the Control Panel
- Through a silent uninstall
- Using the Gateway Installer

To **remove** the Gateway using the Control Panel:

1. Go to **Start, Control Panel**, and then click **Programs and Features**.
2. Select AVEVA NET Gateway for SmartPlant 3D from the list of programs.
3. Select Uninstall to remove AVEVA NET Gateway for SmartPlant 3D.

A silent or quiet uninstall is a type of command for a computer program you can use to uninstall without displaying a progress or without indicating the process.

To install the Gateway from the command line:

1. Navigate to the location of [.msi](#) file in the command prompt.
2. Type **/uninstall Smart3DGatewayInstaller.msi** or type **/x Smart3DGatewayInstaller.msi** to uninstall the gateway.


To **remove** the Gateway using the Gateway Installer:

1. From your installation media, double-click the [Smart3DGatewayInstaller.msi](#) file.
2. Click **Next**.
3. Click **Remove**.
4. To confirm that you want to remove the Gateway, click **Remove** again.
5. Follow the on-screen instructions.

Chapter 2 Run the Gateway from the Project Explorer

The **Project** menu enables you to create new projects and browse the existing projects on your computer. You can also load an existing project configuration, update a project configuration using the **Project** menu and execute projects using the **Run** button.

To start working with the **Project** menu of the Gateway:

1. Type SmartPlant 3D Gateway in the Search box on the taskbar in your computer. Alternatively, you can click the AVEVA NET Gateway For SmartPlant 3D 6.1 icon in the installed location or the shortcut icon on your desktop, if that option was selected when installing the Gateway.
2. **AVEVA NET Gateway for SmartPlant 3D** page opens with the **Project** and **Help** menus in the main window.
3. Click the **Help**  icon to get the information relating to the Gateway version, copyright information and product synopsis.

Note: The Gateway supports Universal Naming Convention (UNC) paths. By using UNC paths, you can connect to servers and other workstations without mapping to a drive.

The UNC path syntax is as follows: [\\servername\sharename\directory](#)

You can use this UNC path syntax in configurations and command line execution scripts. You must have the appropriate authorization to read/write to the target directory.

Create Projects

To create a project in the **Project** tab of the Gateway:

1. On the **Project** tab, click **New**.

The screenshot shows the 'Project' dialog box in the AVEVA NET Gateway for SmartPlant 3D. The window has a blue title bar and a dark blue header. Below the header is a 'Project (...)' label and a 'Run' button. A tab bar shows 'Project' (selected), 'Tools', and 'Info'. The main area has a left sidebar with 'Open' and 'New' buttons. The 'New' section contains fields for 'Name' (A2), 'Description' (Test Project), and 'Path' (E:\Gateway\Project). There is a 'Create New' button and a checkbox for 'Include Default Input File'.

2. Type the project name, and project description (optional) in the respective **Name** and **Description** boxes.
3. Type the project path in the **Path** box, or you can browse to the folder using the Browse **path** button, to create the project at the desired location.
4. Click **Create New**.

Note: If you type the required description in the **Name** and **Path** boxes, the **Create New** option is enabled.

If the folder does not exist, a new project folder is created containing the Project configuration file with the same Project Name. When a project is created, a project xml file is created with default setting. When the project is loaded successfully, a message is displayed in the status bar of the window. This message indicates that the project configuration files are loaded into the user interface.

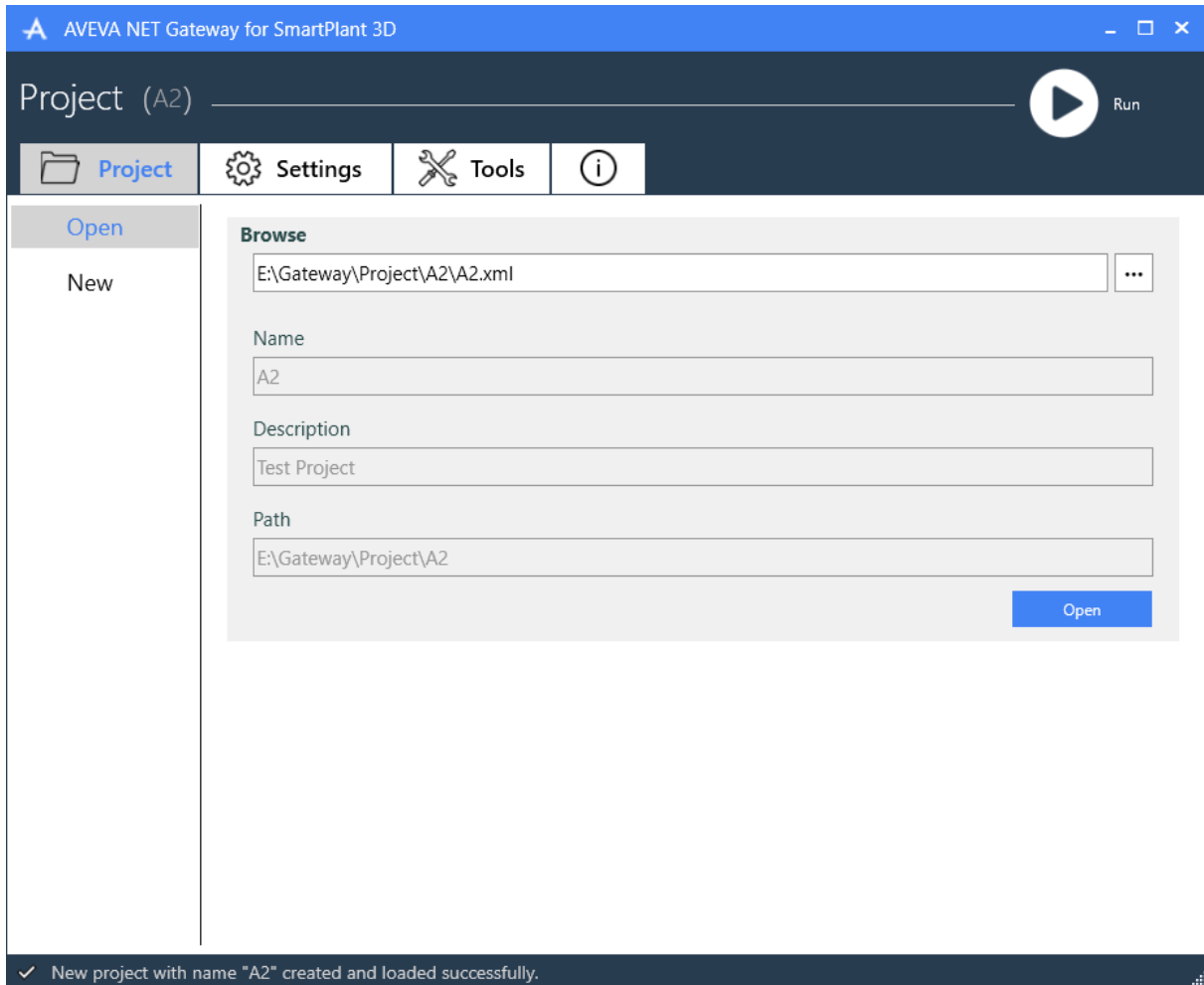
5. Select the Include Default Input File box to create some example input data.

Open Projects

You can open an existing project using the **Project** tab.

To open an existing project:

1. On the **Project** tab, click **Open**.



2. Browse to the **Project XML** file using the **Browse** button. Alternatively, you can enter the name of the project setting file path directly and click **Browse** and select the **Project XML** file.
3. After you have found the **Project XML** file, the remaining fields, for example, **Name**, **Description** and **Path** are populated from the **Project XML** file.
4. Click **Open**. The **Settings** tab is enabled with the contents of **Project XML** file.

Note: The existing project configuration files are parsed for any schema validation issues. If any validation issue is detected, an error message is displayed in the Windows status bar on the relevant page affected by the error.

Define the Project Settings

After the project is successfully loaded, the **Settings** menu is enabled. You can use the **Project** settings to fetch data from the input location. The **Run** option also gets enabled in the menu bar to execute the project.

The **Settings** menu contains the following tabs:

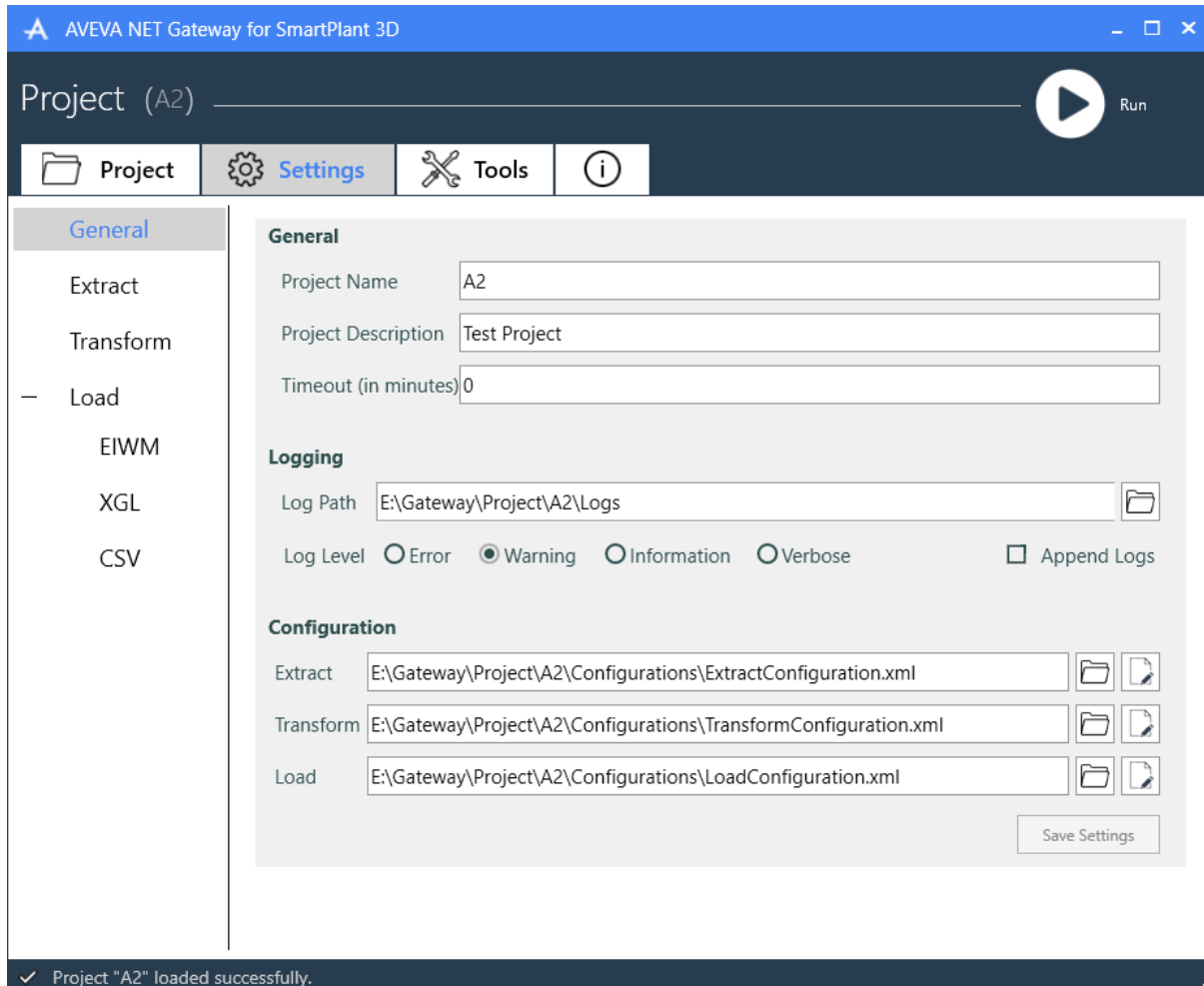
- **General**
- **Extract**
- **Transform**

- **Load**

General Settings

General settings allow you to define the project name, description, log file path and configurations of project files. The **General** settings tab contains the following areas:

- **General:** Defines all the project related information and processing timeout details.
- **Logging:** Defines the log file path and log file name with log level information.
- **Configurations:** Defines configuration details relating to extract, transform and load.



The screenshot displays the 'AVEVA NET Gateway for SmartPlant 3D' application window. The 'Project (A2)' header is visible. The 'Settings' tab is active, showing the 'General' section. The 'Project Name' is 'A2', 'Project Description' is 'Test Project', and 'Timeout (in minutes)' is '0'. The 'Logging' section shows 'Log Path' as 'E:\Gateway\Project\A2\Logs' and 'Log Level' set to 'Warning'. The 'Configuration' section shows fields for 'Extract', 'Transform', and 'Load' configurations, all pointing to XML files in the 'E:\Gateway\Project\A2\Configurations' directory. A 'Run' button is located in the top right corner. A status bar at the bottom indicates 'Project "A2" loaded successfully.'

Note: The triangle symbol near the **General** tab indicates that some fields have unsaved changes or have invalid data.

General Area

You can use the fields in the **General** area to add project-related information, such as the project name, project description and processing timeout details.

To complete the project related information:

1. Type the Project Name and Project Description in the respective fields.

Note: The **Project Name** is a required field and cannot be empty.

2. Type the time in the **Timeout (in minutes)** box to define the maximum execution time of the project.

Note: The **Timeout** value must be a non-negative integer.

3. Click **Save Settings** to save the general settings options.

Logging Area

You can use the fields in the **Logging** area to specify different logging information, such as warning and error messages. The Gateway produces a log file in the location defined by Log Path. Its default value is a Logs subfolder under the project file location.

To specify the Logging information:

1. Type the log file path in **Log Path** box or click the **Browse Path** icon to specify the log file path.
If the log folder path exists, it will create the logs in the specified log path. If the log folder path does not exist, it creates the folder automatically. The default name of the log file name is the name of the input data. The log files will be named as `<input filename>.log`.
2. Specify a **Log Level** to log all events greater than and equal to the specified severity level. For example, if you set the **Log Level** to **Verbose**, all errors, warnings and informational messages are logged.

Note: The **Log Level** value is case-sensitive and must be one of the following: **Error**, **Warning**, **Information** and **Verbose**. The following table shows the severity standard for different Log Levels.

Log Level	Severity
Error	Indicates logs for an error message. An Error is an unexpected condition that is likely to result in erroneous information in the output.
Warning	Indicates logs for an error and warning message. A Warning is a message or a notification that alerts you of a condition that may cause a problem in the future.
Information	Indicates logs for an error, warning and other informational message. An Information is a message with a concise report of a object processing, timestamp, problem trigger, processing details and working of software modules
Verbose	Indicates logs at all levels, that is, error, warning and informational messages. A verbose is a message that provides additional details about the activity start and end, dump data and the values of the variables and arguments that are used. This extra information can be helpful during troubleshooting, but as it creates large log files this option should be avoided in normal runs.

3. Select the Append Logs box in the **General** settings Page to control the value of `appendToLog` in the configuration file. By default, the `appendToLog` value is deselected and false.

The following two scenarios exist under **Append Logs** section:

- If you select Append Logs box, then the first run of the Gateway project will create the log file and add messages to it. The value of appendToLog (in the project's Configuration file) is updated to true. On subsequent runs, that processes the same input source, the messages are appended to this existing log file.
- If you deselect the Append Logs box, then each run of the Gateway project will delete the old log file if it exists and create a new log file and add messages to it.

```
<log level="Warning" folderPath="C:\Users\<user_name>\Test\Logs_Appended"
appendToLog="false" />
```

4. Click **Save Settings** to save the **Logging** area settings.

After processing, the Gateway produces a [Summary.txt](#) file in the Project Logs folder, which contains statistics about the amount of extracted and loaded objects, project path, file processing information, number of errors and warnings and processing time.

Statistic	Source	Description
Number of entities extracted	Extractor	Number of engineering objects in Object Model after extraction
Number of XGL/ZGL objects	XGL Loader	Number of 3D graphical objects in Object Model that were loaded into the output XGL/ZGL file
Number of EIWM objects	EIWM Loader	Number of engineering objects in Object Model that were loaded into the output EIWM file. This may be different from the number of extracted engineering objects if transformation mapping has removed some objects or created new ones or have been excluded by use of the "#EXCLUDE_FROM_EIWM#" attribute, see Attribute Mapping ().

Example:

Processing Started: 01/13/2022 21:41:35

Project Path: c:\temp\A5\A5.xml

A5

Number of EIWM objects: 189696
Number of XGL/ZGL objects: 114505
Number of Errors: 0
Number of Warnings: 66
Processing Time: 14901.70 seconds.

1 files successful
0 files failed

Processing Finished: 01/14/2022 01:49:57

Configuration Area

You can use the fields in the **Configurations** area to define the project configuration details.

To define the configuration settings:

1. Type the configuration details in the **Extract**, **Transform**, **Load** fields, respectively:

- **Extract:** Provide the configuration file details to extract the engineering and graphics data from the Smart 3D model. For new projects, a default configuration file for Extract is created, named as [ExtractorConfiguration.xml](#).


Note: The **Extract** path value must point to a valid extractor configuration file.


- **Transform:** Provide the configuration file details to transform the extracted data. The configuration file for Transform is named as [TransformConfiguration.xml](#).

Note: The **Transform** path value must point to a valid transformer configuration file.

- **Load:** Provide the configuration file details to load the extracted and transformed engineering data. The configuration file for Load is named as [LoadConfiguration.xml](#).

Note: The **Load** path value must point to a valid loader configuration file.

For each specific setting, you can type the path of the configuration file (.xml file) or click **File Selector** at the end of the box. To modify the respective configuration files, click .

1. Click the  icon to browse to the path of configuration settings.
2. Click **Save Settings** to save the **Configuration** field settings.

Note: Save Settings is applicable to all settings of the page. You can save the settings after filling the required information in **General** settings.

Extract Settings

The Gateway extracts model and engineering data from the Smart 3D API. Therefore, the Gateway should be installed on the same machine that also hosts Smart 3D.

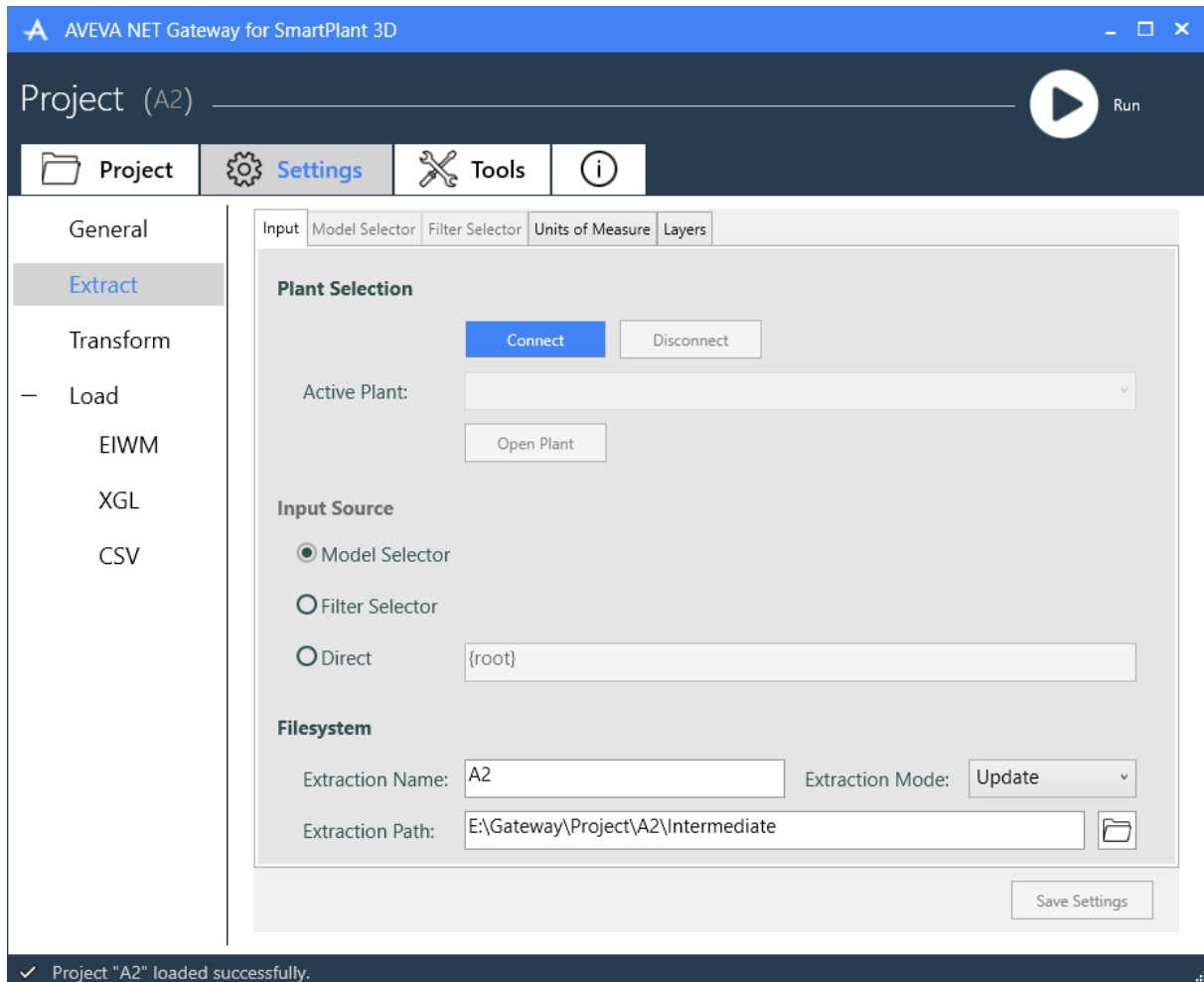
For performance reasons, the extraction is a two-stage process that initially writes the extracted data to an extraction file that can then be updated by subsequent delta extractions. The Gateway's output EIWM and XGL/ZGL files are then generated from this extraction file.

To specify how to extract the data:

1. Click **Extract** to display the **Extract Smart 3D** field.

Extract settings pages opens with a set of tools for the **Extract** operation.

2. Click **Input** menu from the **Extract** page to define the input source and extract file path.

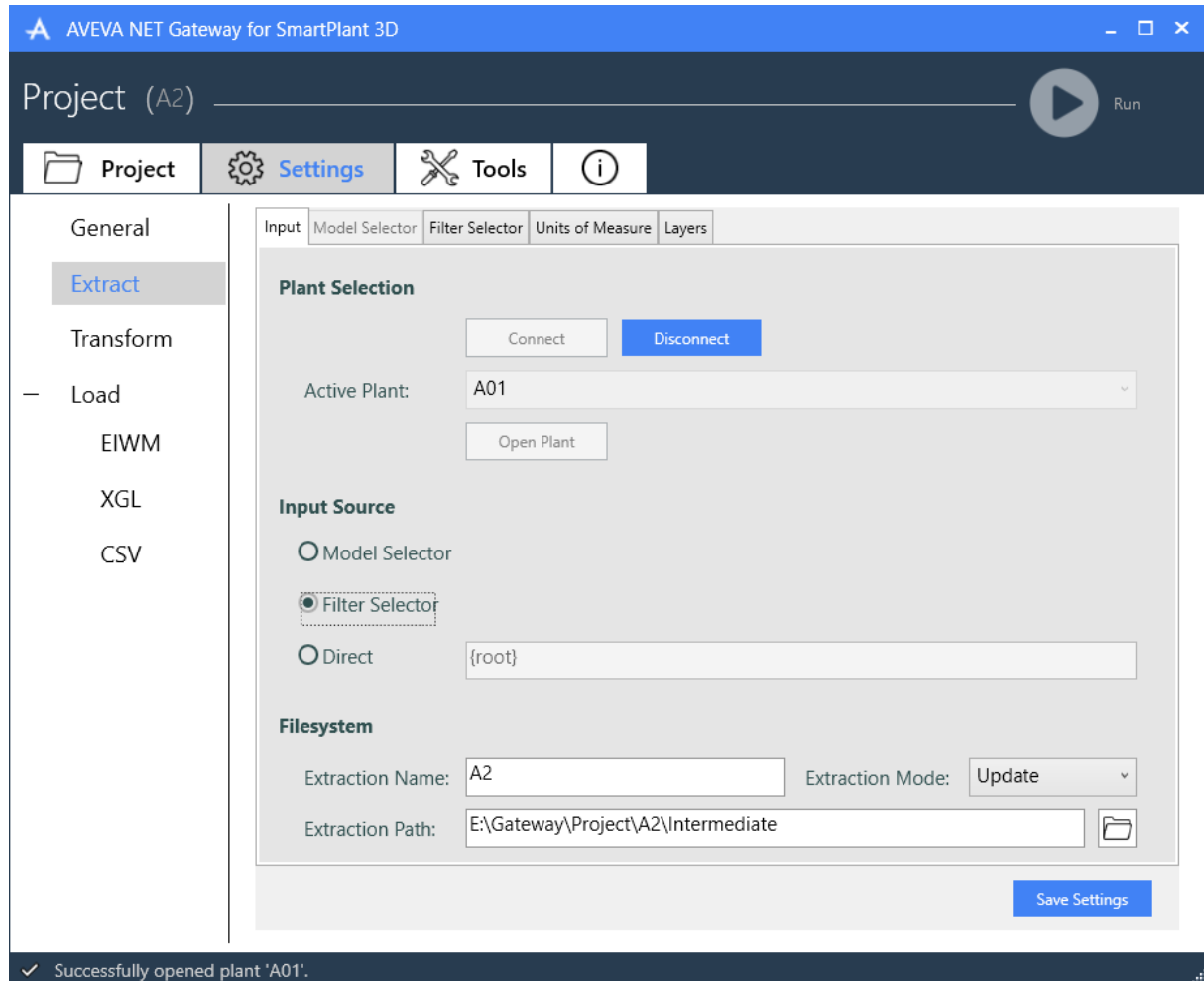


3. In the **Plant Selection** section, click **Connect**.

If successful, a connection is established between the Gateway and Smart 3D. The **Active Plant** drop-down list is populated with models from the active Smart 3D site.

Note: The Gateway user must be the same user who has privileges to run the Smart 3D application. During extraction of data, a Smart 3D license is required to allow communication between the Smart 3D API and the Gateway.

4. Click **Disconnect** to close the connection and release the Smart 3D license.
5. Select the relevant **Active Plant** to be extracted.

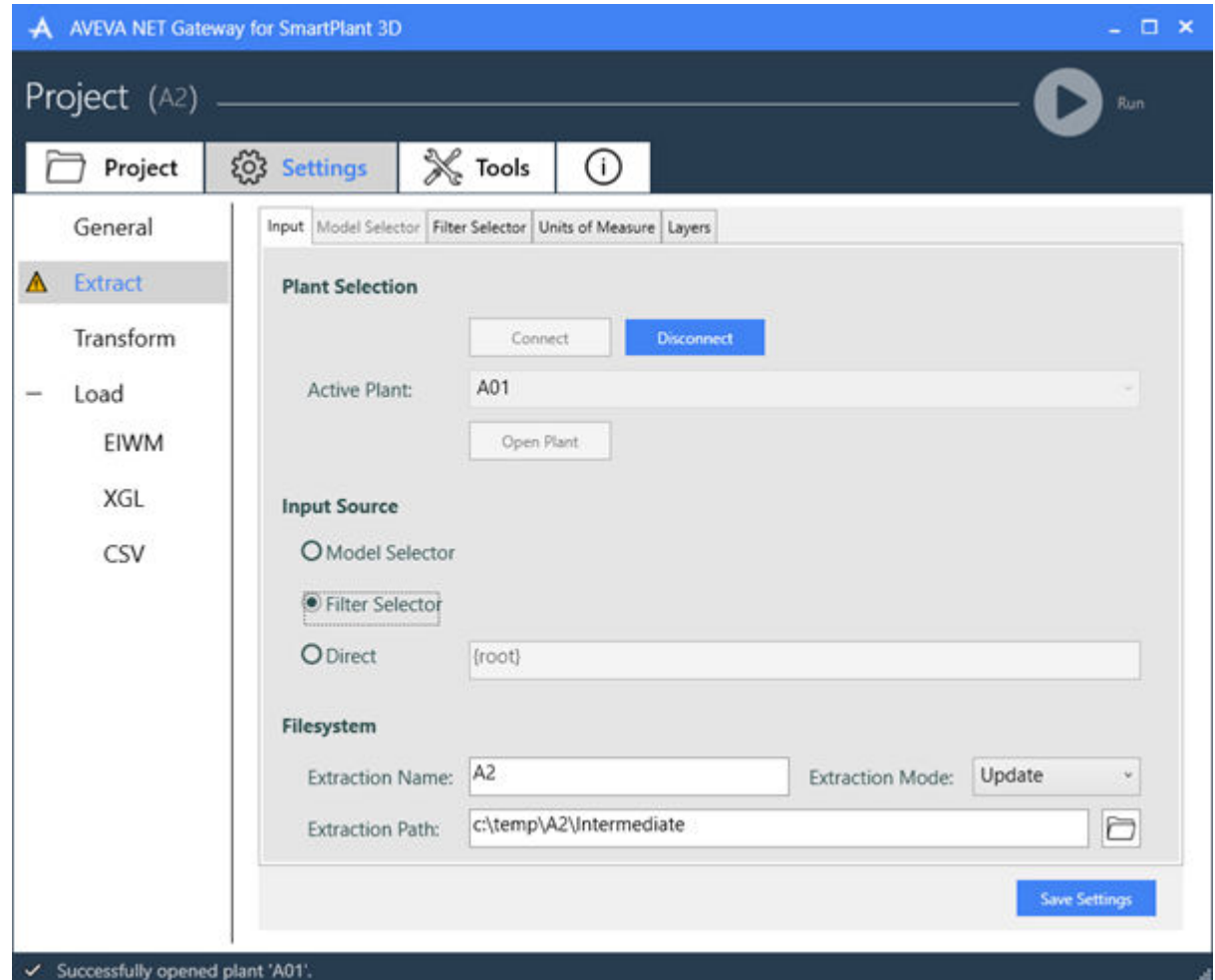


From the **Input Source**, select the specific data in the plant you want to extract. If you select **Model Selector** or **Filter Selector**, then the relevant tab is enabled where you can enter the details of that selector:

- **Model Selector** – When you select this option, the Gateway will display a representation of the System Workspace Explorer from which you select the elements to extract.
 - **Filter selector** – When you select this option, the Gateway will display the filters defined in the Smart 3D project. Only those elements that pass the filter will be extracted.
 - **Direct** – When you select this option, you can manually enter the relevant Smart 3D UID, which will extract the Database Object ID and its children. This field allows for a list of UIDs.
6. In the **Filesystem** section, type the details of the extraction file in **Extraction Name** and **Extraction Path** boxes. Alternatively, you can use the **Browse File Path** option to enter the input data location.

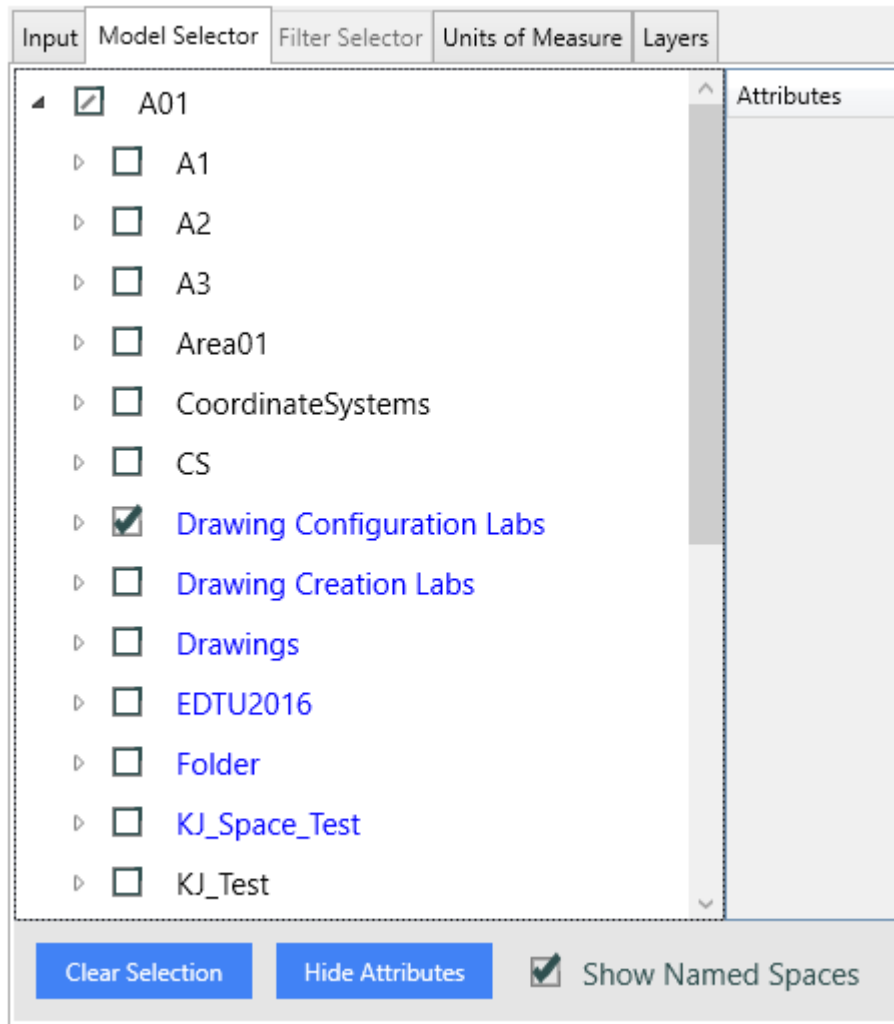
In the **Extraction Modes** section, select the required option from the following options to either extract an existing or a new Input data.

- **Update** – This mode compares the current state of the data in Smart 3D with the data saved in the previous extraction file and only updates the data that has changed; updates of existing objects and new objects; or objects that have been deleted. This mode can greatly reduce the extraction time if only a few changes have been made since the last extraction. If the old data does not exist, update will work as **New**. The Update mode is the default extraction mode.



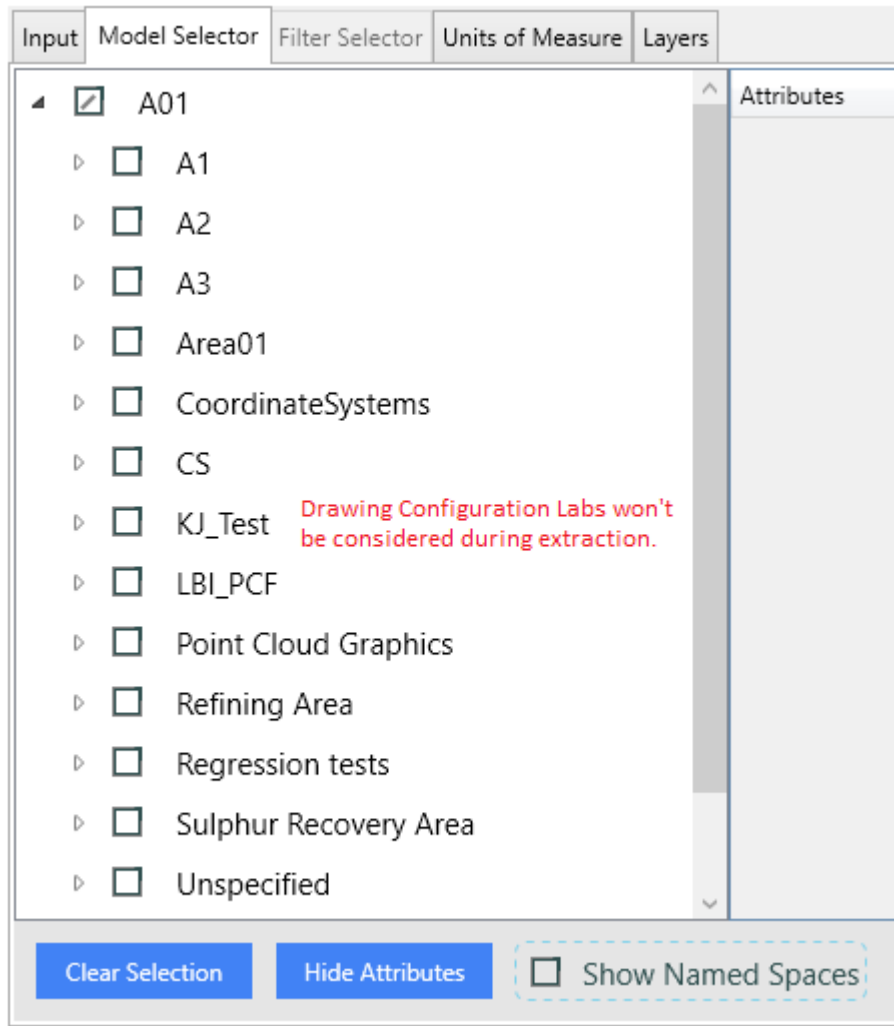
- **New** – This mode creates a new extraction file via a full extraction, any previous extraction file with the same file name will be deleted.
- **Existing** – This mode creates the output EIWM and XGL/ZGL files from the existing extraction file. No new extraction is done here. This can save both time and the use of a Smart 3D license if just the transformation needs to be changed or if the extraction file is shared between machines.

7. From the **Model Selector** option, select the elements to export. You can partially select or deselect elements.

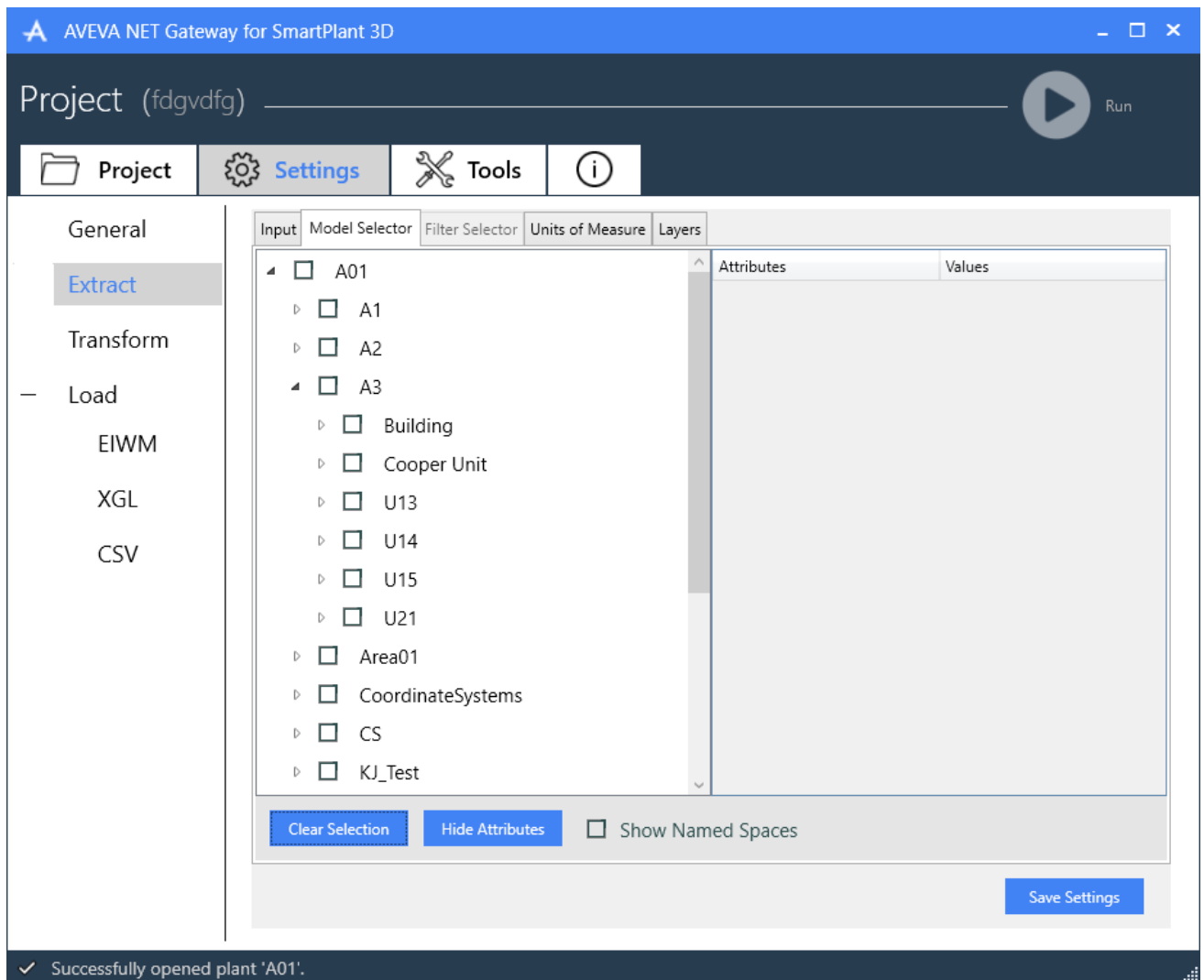


The **Show Named Spaces** checkbox shows/hides space elements in the **Model Selector**.

If you clear the **Show Named Spaces** checkbox, none of the previously selected space elements will be extracted.

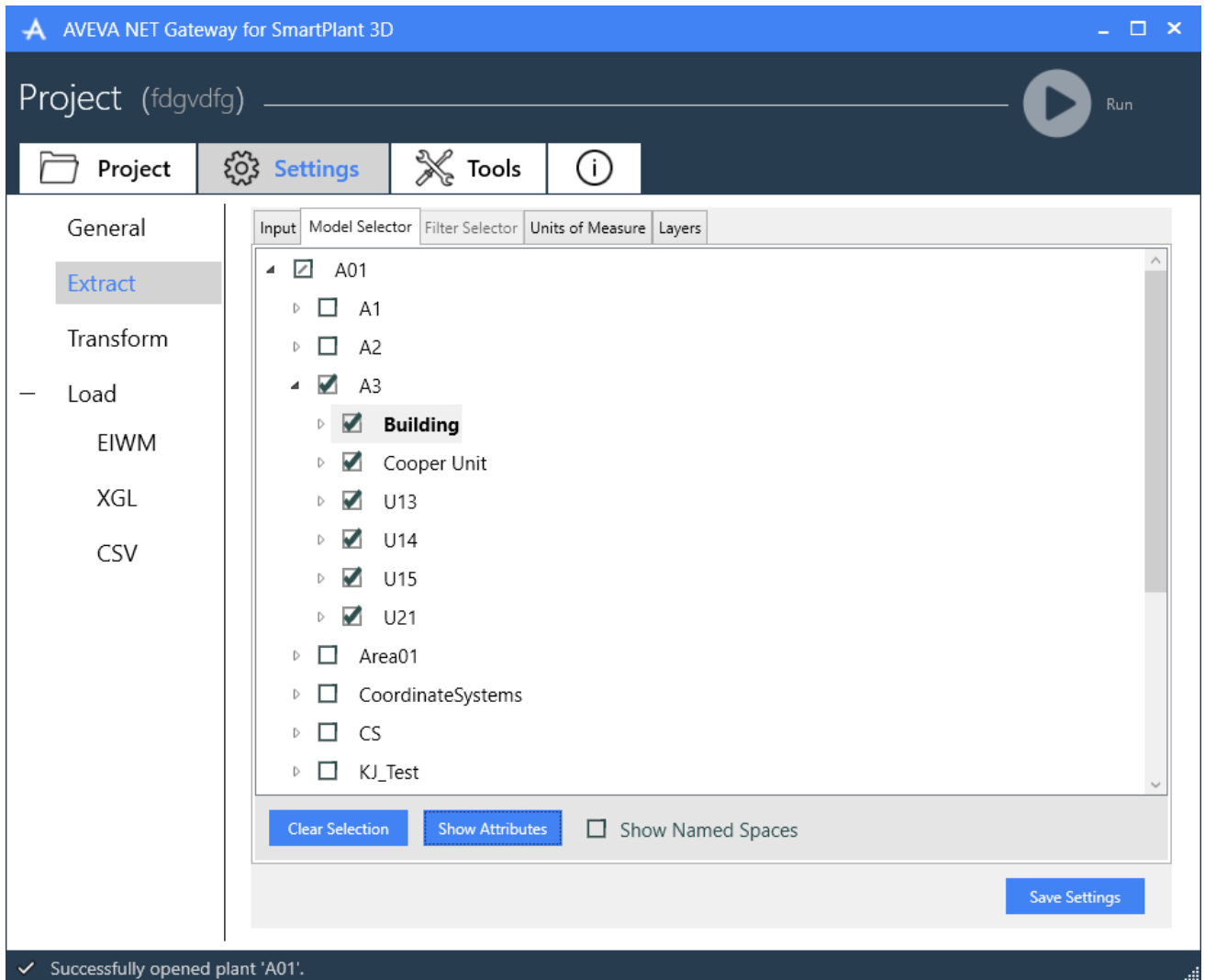


8. Press **Clear Selection** to remove all the selected checkboxes.



9. Press **Show/Hide** button to show or hide additional tab with the attributes of selected component, as shown in the following figures:



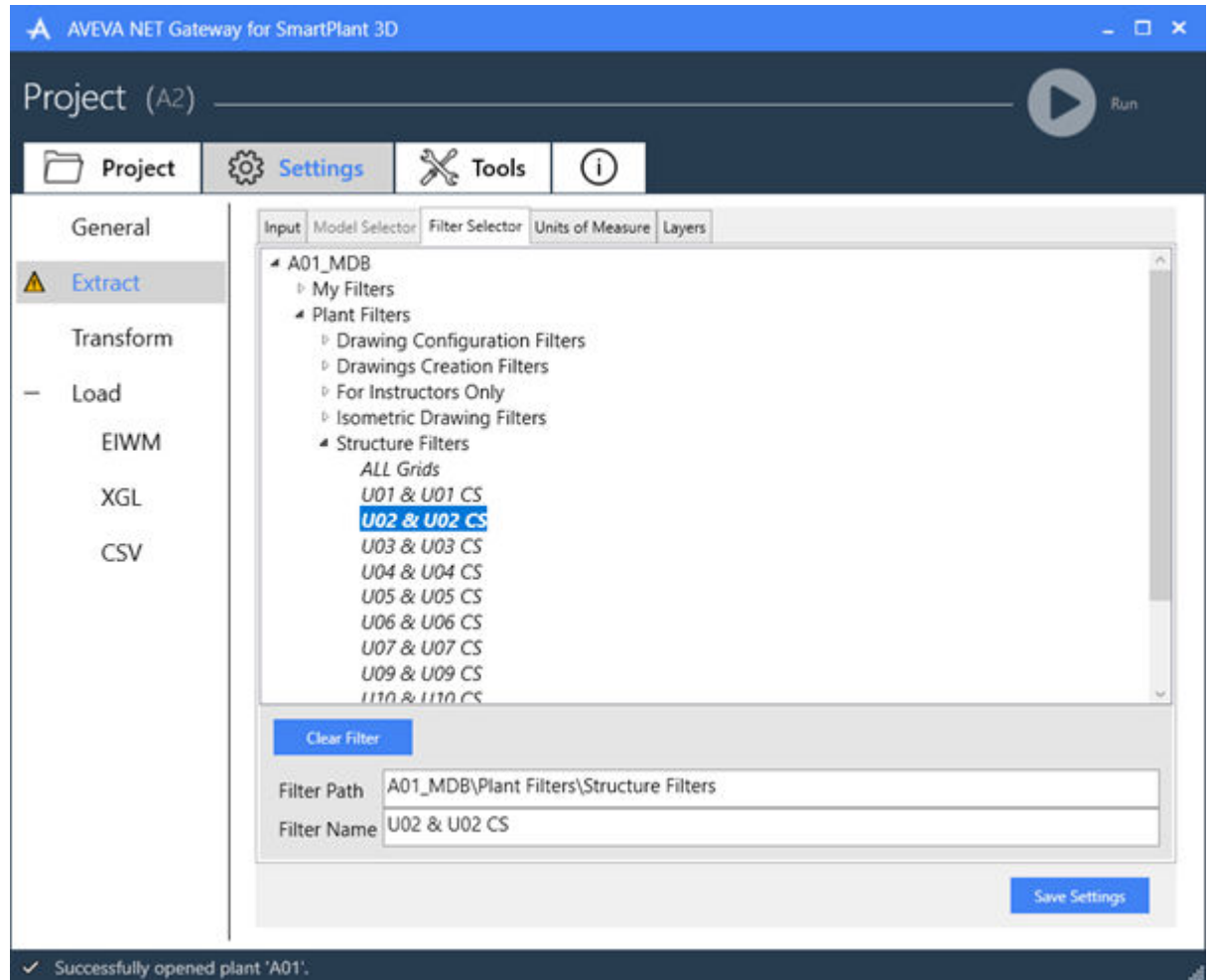


10. Select **Filter Selector** to apply the Smart 3D filter on the extracted items. The **Filter Selector** provides the following fields:

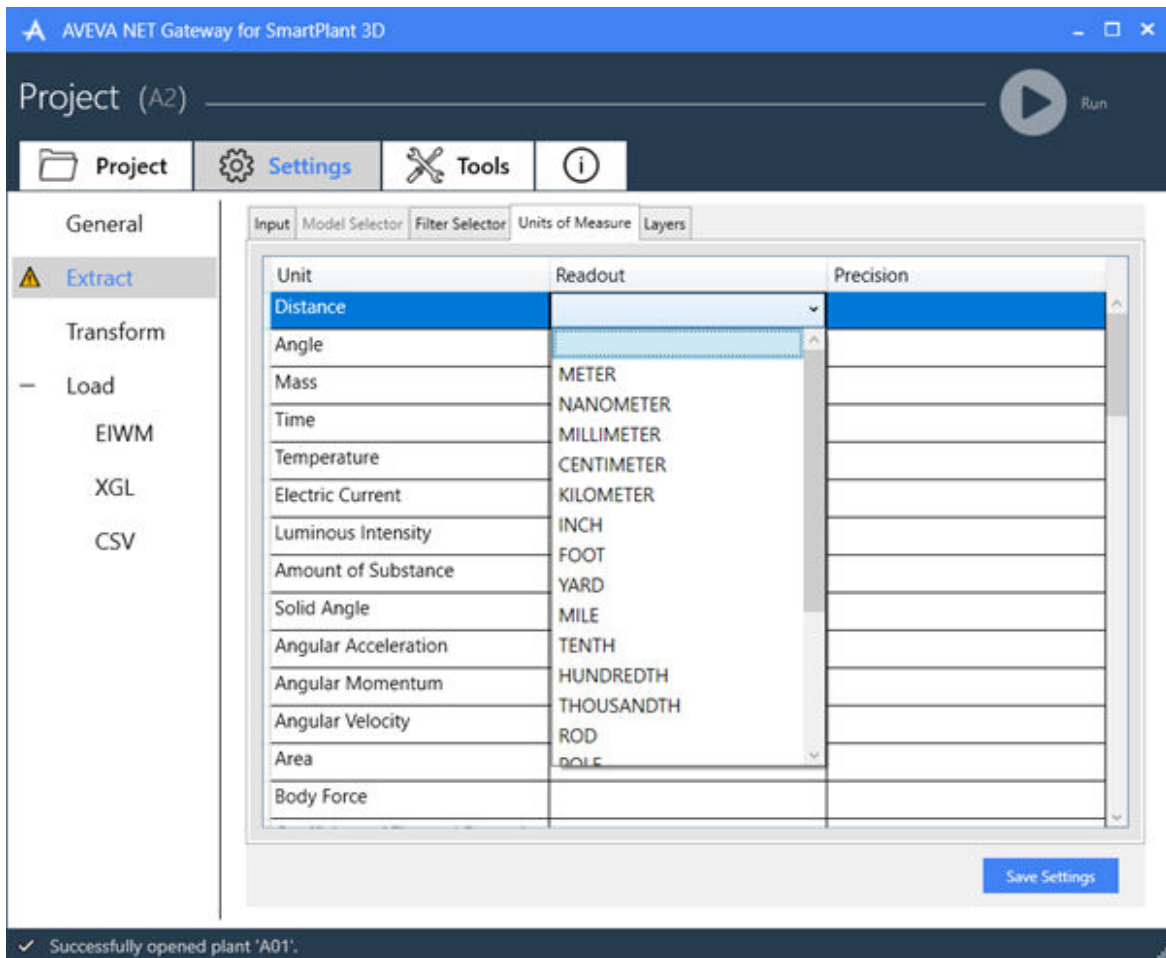
- **Filter Path** – Indicates the path to the location of the filter in Smart 3D.
- **Filter Name** – Displays the name of the applied Smart 3D filter.

Notes:

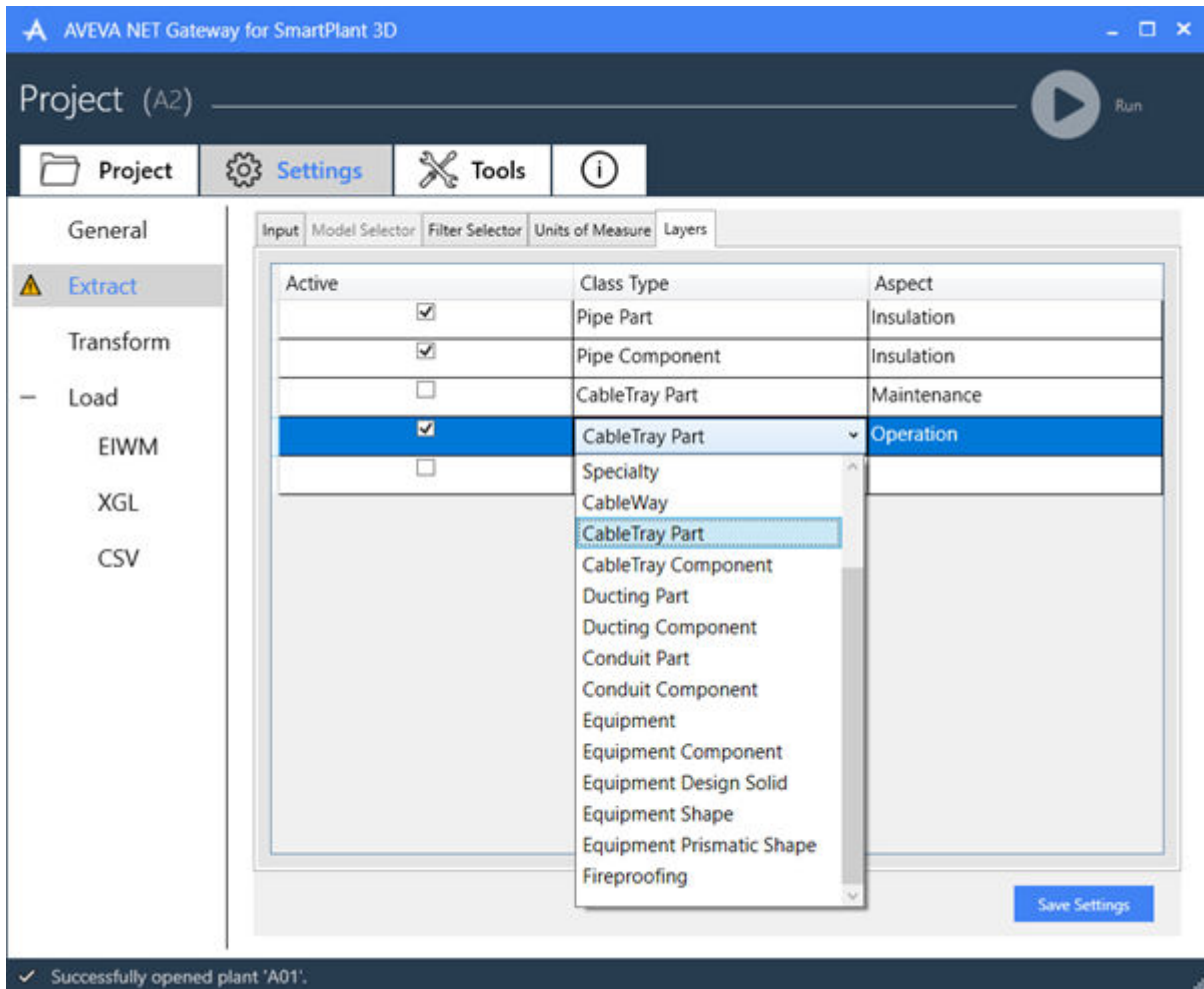
- You can select only one filter for extraction.
- **Filter Path** and **Filter Name** are read-only.



11. Select **Units of Measure** tab to customize the readout and/or precision of selected units. By default, the Gateway reads all the units with default database units.



12. Select the **Layers** tab to define the specific geometry to extract in addition to the **Simple Physical** geometries.



13. Use the **Active** column to select each relevant **Class Type**, and for each **Class Type** select what aspect of that class to extract. By default, the Gateway reads only the geometry that is bounded to the **Simple Physical** aspect.

14. Use **Enter** on the keyboard to add a new row or click on another row to insert a new row.

For example, if column **Active** is enabled for the **Pipe Part Class Type** and an **Insulation Aspect** is selected, then the Gateway will extract Insulation geometries along with **Simple Physical** geometries.

Notes:

- It is possible to extract geometries from more than one additional aspect. For example, if **Class Type** of **CableTray Part** is selected and an aspect of **Maintenance** is chosen, a second row can be created to select the same **Class Type** and an aspect of **Operation** can also be chosen, then the Gateway will extract additional geometries from the **Operation and Maintenance** aspects for **CableTray Part** objects.
- The **Units of Measure** and **Layers** are defined in the Gateway installation and are specific to that version of the Gateway.
- Geometries from **Simple Physical** are always extracted, they cannot be disabled.

Transform Settings

Transform configuration references to mapping configurations (defined in separate mapping files) that can be used to select and transform the input's objects and their properties. All the transform-related settings are grouped into a set of extensions. Each extension section in the configuration file is optional and can be repeated any number of times. These extensions are used to modify particular parts of an object and its properties.

The default configuration file contains three transform extensions:

- BaseMapping:** Modifies the engineering data of a model's objects present in the extracted data.


```
<MappingTemplate componentVersion="2.2.0.0" sourceProductName="AVEVA NET Gateway for
SmartPlant 3D" componentName="BaseMapping"
  <ObjectMappings regExTimeoutSecs="10">
    <!-- Modify Document Object(Manifest) ObjectID -->
    <Object>
      <Conditions>
        <Attribute name="#TYPE#" pattern="^manifest$" />
      </Conditions>
      <ObjectID value="#MODEL_NAME#" />
    </Object>
    <!-- Start of mapping -->
    <Object>
      <Attributes keepUnmappedAttributes="true">
        <!-- Set attribute names -->
        <Attribute name="API::ClassName">
          <Name value="ClassName" />
        </Attribute>
        <Attribute name="API::ObjectID">
          <Name value="UID" />
        </Attribute>
        <Attribute name="IJNamedItem::Name">
          <Name value="Name" />
        </Attribute>
      </Attributes>
      <!-- Set ObjectID & ClassID -->
      <ClassID value="[API::ClassName]" />
      <ObjectID value="[IJNamedItem::Name]-[API::ObjectID]" />
    </Object>
    <!-- End of mapping -->
  </ObjectMappings>
</MappingTemplate>
```
- AVEVANETPresentationMapping:** Adjusts the output graphics properties, such as materials and colour. It can also be used to segment the input model to be included in the XGL rendition.


```
<presentation xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://
www.w3.org/2001/XMLSchema-instance" sourceProductName="AVEVA NET Gateway for
SmartPlant 3D" componentName="TransformerPresentationMapping"
  componentVersion="2.2.0.0"
  <segments>
    <segment toSuffix="" includeParts="false" />
  </segments>
</presentation>
```
- CSVReporting:** Exports the extracted input data from the extracted data in the form of a readable CSV file format. For more information, see [Appendix A: Mapping Configuration](#) ().


```
<configuration xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://
www.w3.org/2001/XMLSchema-instance" sourceProductName="AVEVA NET Gateway for
SmartPlant 3D" componentName="LoadCSV" componentVersion="2.2.0.0" >
  <Outputs>
    <S3 isSelected="false">
      <Authentication instance="true">
        <CredentialFile path="default" profileName="default" />
      </Authentication>
      <Region />
      <BucketName />
      <OutputDirectory />
    </S3>
    <FileSystem isSelected="true">
      <OutputPath>c:\temp\A7\Output</OutputPath>
    </FileSystem>
    <AvevaCloudStorage isSelected="false">
      <StoreName />
      <OutputDirectory />
    </AvevaCloudStorage>
  </Outputs>
  <dataType value="CSV" />
  <csvReport addHeaderRow="true" columns="#ObjectType#, ClassID, ObjectID,
ClassName, Name" />
  <keepInputFolderStructureForOutput apply="true" />
  <annotations level="Basic" />
</configuration>
```

Notes:

- As a number of mapping configuration files can be associated to a project configuration, order of execution of these files is determined as follows:
 - When only one [DefaultBaseMapping](#) extension is added to the transform configuration, all the mapping entries from different mapping nodes are consolidated and applied on the extracted data at once.
 - When multiple [DefaultBaseMapping](#) extensions are added to the transform configuration, individual extensions are executed in order, independent of each other.
- Mappings applied as part of an extension are overridden by the other extension if conflicting mapping entries are present in its **mapping** configurations.

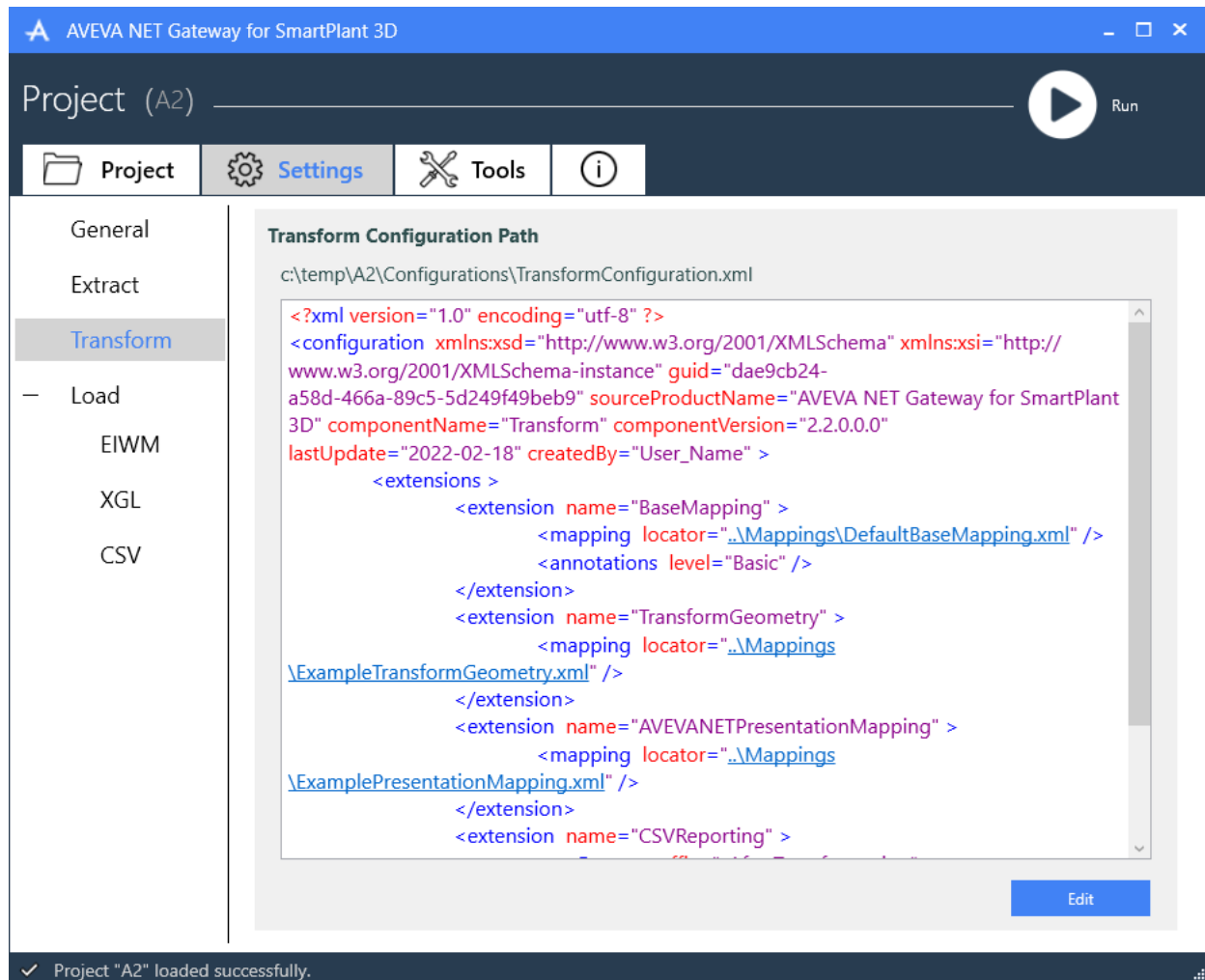
The **Transform** tab in the user interface presents the active project's transform settings as read-only content. The content has anchors linked on the special attribute **locator** that helps in navigating to the configuration file of an individual extension. When you run the project, it invokes the transformation extensions as configured in the configuration file and modifies the output. For more information about the transformation functions and available syntax, see [Appendix A: Mapping Configuration](#) ().

To update transform settings:

1. Click the **Transform** tab in the **Project Settings** page to open the **Transform** details page.
2. Click Edit to open the settings in a **Notepad** where you can modify the default configuration settings relating to Transform.
3. If required, modify dependent mapping files by clicking on the anchor links placed in the main configuration view.

Note: Settings are automatically saved upon saving and closing the **Notepad** session.

- After all the settings have been saved successfully, click **Run** to process the input data.



Notes:

- Any update in the configuration document must be saved before running the process.
- You can set tracking of changes provided by each Transform extension using annotations feature as shown in the above XML configuration. For more information about annotations, see [Appendix C: Tracking Changes in Data](#) ().
- A CSV Report that is defined in the Transformation configuration file can have its output path defined by the [outputFolder](#) parameter. This parameter is optional, and when it is not set, the CSV file is written to the EIWM location by default. Currently, this parameter can keep path for the File System but not for the S3 Bucket.
- For more information about handling system attributes, see [Appendix G: Handling of System Attributes](#) ().

Load Settings

The Gateway uses the **Load** component to load the selected and transformed objects into AIM via EIWM files, including hot-spotting these objects in their XGL/ZGL 3D model representation. Load component also exports the transformed data into various other CSV files.

EIWM

The **EIWM** setting under the **Load** menu contains various options to define the settings to store EIWM Output data along with the FileSystem information.

The screenshot displays the 'AVEVA NET Gateway for SmartPlant 3D' application window. The 'Project (A2)' is selected, and the 'Load' menu is open, with 'EIWM' highlighted. The 'EIWM' settings panel is visible, showing the following configuration:

- Data Output Type:** EIWM (selected from a dropdown)
- Output Destination:** ☒ File System, ☐ S3, ☐ AVEVA Cloud Storage
- File System:**
 - Output Path:** E:\Gateway\Project\A2\Output (with a folder icon)
 - ☒ Replicate Input Structure
 - ☐ Custom Folder Structure
 - File Name:** (empty text box)
 - Folder Structure:** (empty text box)
 - ☒ Add Description to Header
 - ☒ Create Document Object
 - ☒ Generate Trigger File
 - ☐ Create Alias in EIWM Objects
 - Alias Attribute:** (empty text box)
 - Project Context:** Avngate

At the bottom, the 'Log View' panel shows the following messages:

Log Type	Timestamp	Message
Information	3/18/2025 12:18:40	(XGL Loader): Exporting XGL data to file: 'E:\Gateway\Project\A2\Output\A2.xgl'
Information	3/18/2025 12:18:40	Finished processing

A status bar at the bottom indicates 'Process successful.'

The **EIWM** setting contains the following options:

- **Data Output Type:** Normally set to EIWM where the output Engineering data will be written to an EIWM XML file. Selecting **None** means that no Engineering data is written, for example, when only a CSV report is needed. You can configure this in the Load EIWM Configuration file using the `dataType` element, for example, `<dataType value="EIWM" />`.

Note: These values can be a case-insensitive representation of either "EIWM" or "NONE".

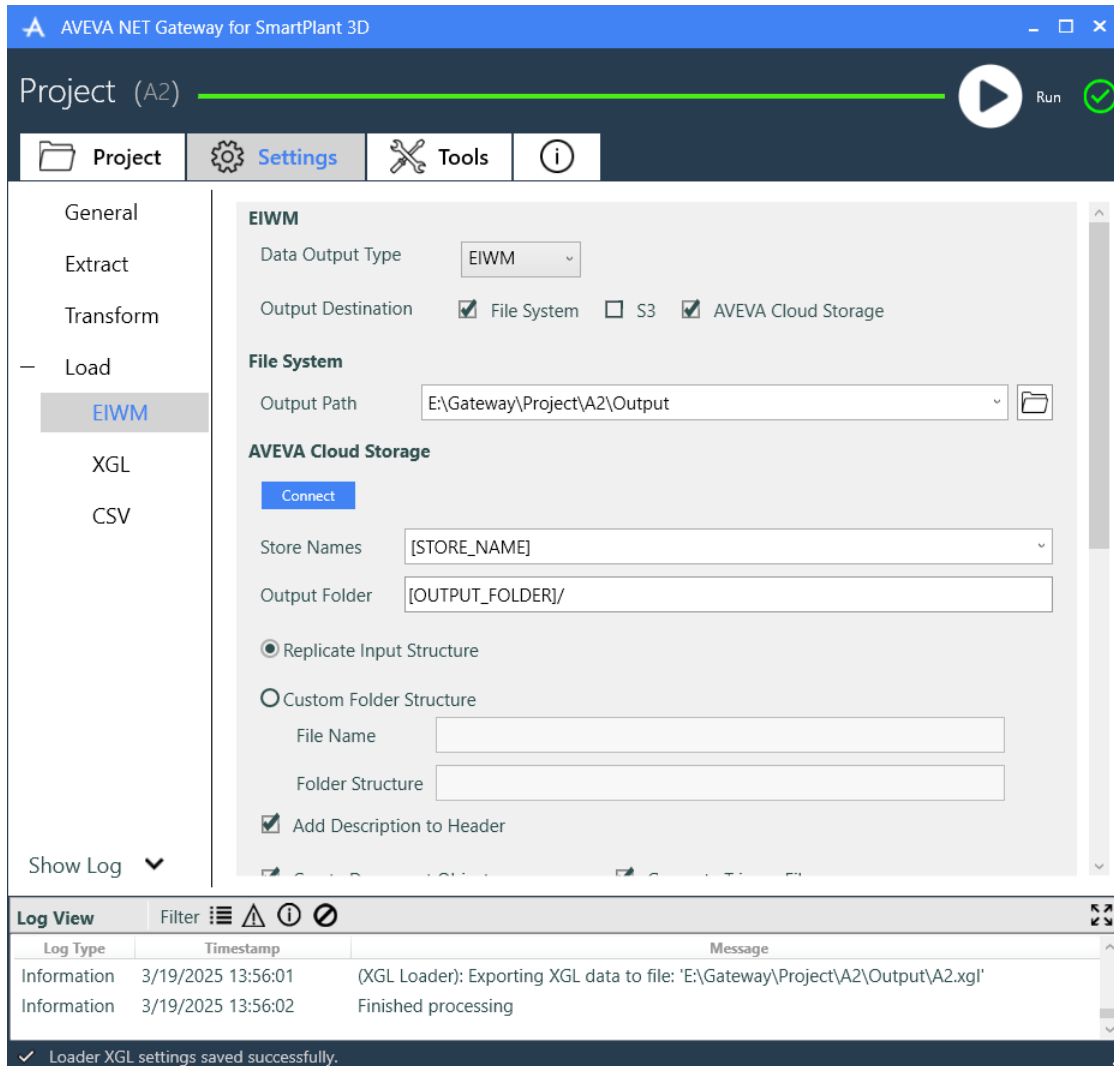
- **Output Destination:** Select whether the output will be written to a **File System** or an Amazon Web Services (AWS) **S3 Bucket** or **AVEVA Cloud Storage**. You can configure this in the Load EIWM Configuration file using the OutputDestination element, for example, <OutputDestination Destination="FileSystem">.

Note: The [Destination](#) values can be a case-insensitive representation of either "FileSystem" or "S3", or "Both". You can select all these types independently or select one of the S3 and AVEVA Cloud Storage type with FileSystem destination type together.

- **File System:** The file system is normally the path that defines the AIM staging area.
 - **Output Path:** Type the Output file path or click the **Browse for Folder** tab to enter the output folder path.
- **S3:** To access an S3 bucket requires the relevant authentication details like credential file, region information, bucket name and output directory. For more information, see [Accessing an AWS S3 Bucket](#) ().

The screenshot displays the AVEVA NET Gateway for SmartPlant 3D application. The main window is titled "Project (A2)" and includes a "Run" button with a green checkmark. The "Settings" tab is selected, showing the "EIWM" configuration section. Under "Data Output Type", "EIWM" is selected. Under "Output Destination", both "File System" and "S3" are checked. The "File System" section shows the "Output Path" as "E:\Gateway\Project\A2\Output". The "S3" section has "Instance Profile" checked, with fields for "Credentials File" (default), "Profile Name" (default), "Region" ([REGION_NAME]), "Bucket Name" ([BUCKET_NAME]), and "Output Folder". A "Test Connection" button is present. At the bottom, the "Log View" shows two messages: "Information 3/19/2025 13:56:01 (XGL Loader): Exporting XGL data to file: 'E:\Gateway\Project\A2\Output\A2.xgl'" and "Information 3/19/2025 13:56:02 Finished processing". A status bar at the very bottom indicates "Loader EIWM settings saved successfully."

- **AVEVA Cloud Storage:** For more information about accessing AVEVA Cloud Storage, see [Accessing AVEVA Cloud Storage](#) ().



- **Add Description to Header:** Select this option to add the metadata to a header of EIWM file. You can configure this using the `addDescriptionToHeader` element, for example, `<addDescriptionToHeader apply="false" />`.

Note: The `apply` value must be an `xsd` schema supported Boolean value. The valid values for `xsd:boolean` are `true`, `false`, `0` and `1`. Attribute values that are capitalized (for example, `TRUE`) or abbreviated (for example, `T`) are not valid.

- **Replicate Input Structure:** Select this option to generate an output folder structure from the base input path. Any sub-folders below the input locator will be replicated in the output destination, and the files generated from the input data in that structure will be in the corresponding output sub-folders. You can configure this using the `FolderStructure` attribute of the `OutputDestination` element, for example, `<OutputDestination Destination="FileSystem" FolderStructure="keepInputFolderStructureForOutput ">`
- **Custom Folder Structure:** Select this option to enable the **File Name** and **Folder Structure** fields. To write all output files to the **Output Folder**, leave the **File Name** and **Folder Structure** fields empty.

You can use the **File Name** field to create multiple smaller EIWM files.

Custom File Name Structure

File Name is configurable and the expected entries in a **File Name** field can be the same as defined in the Folder Structure. If the project context is **avngate** or **IPE**, then the suffix **"_avngate.xml"** or **"_null.xml"** is derived from the default project context value by default. Any other value for the project context results in the **EIWM:Context** attribute value being used as the suffix to the file name.

- **Scenarios:**

- An example configuration that creates an output file for each object in a data source:

```
<OutputDestination Destination="FileSystem"
FolderStructure="CustomFolderStructureForOutput">
  <S3>
    <Authentication instance="true">
      <CredentialFile path="default" profileName="default" />
    </Authentication>
    <Region />
    <BucketName />
    <OutputDirectory />
  </S3>
  <FileSystem>
    <OutputPath>C:\Test \TestProject\Output</OutputPath>
  </FileSystem>
  <keepInputFolderStructureForOutput />
  <CustomFolderStructureForOutput>
    <FileName value="[eiwm:id]" />
    <FolderStructure value="" />
  </CustomFolderStructureForOutput>
</OutputDestination>
```

In this case, an XML file is created with the eiwm:id prefixed to the file name to generate a unique file name, for example, **p-100_avngate.xml**.

Note: Performance of the Gateway may be compromised while generating a very large number of files.

- An example configuration that distributes the output across different directories by classification:

```
<OutputDestination Destination="FileSystem"
FolderStructure="CustomFolderStructureForOutput">
  <S3>
    <Authentication instance="true">
      <CredentialFile path="default" profileName="default" />
    </Authentication>
    <Region />
    <BucketName />
    <OutputDirectory />
  </S3>
  <FileSystem>
    <OutputPath>C:\Users\<user_name>\Desktop\sdfsfgsd\Output</OutputPath>
  </FileSystem>
  <keepInputFolderStructureForOutput />
  <CustomFolderStructureForOutput>
    <FileName value="output" />
    <FolderStructure value="[eiwm:ClassID]" />
  </CustomFolderStructureForOutput>
</OutputDestination>
```

The above results in a sub-directory being created for each classification. Each directory contains a file 'output.avngate.xml', if the Project context is set to [avngate](#). If Project context is other than avngate, the suffix will be _null.xml.

- An example configuration that creates a single file per object within the classification sub-directories:

```
<OutputDestination Destination="FileSystem"
FolderStructure="CustomFolderStructureForOutput">
  <S3>
    <Authentication instance="true">
      <CredentialFile path="default" profileName="default" />
    </Authentication>
    <Region />
    <BucketName />
    <OutputDirectory />
  </S3>
  <FileSystem>
    <OutputPath> C:\Test \TestProject\Output</OutputPath>
  </FileSystem>
  <keepInputFolderStructureForOutput />
  <CustomFolderStructureForOutput>
    <FileName value="[eiwm:id].xml" />
    <FolderStructure value="[eiwm:ClassID]" />
  </CustomFolderStructureForOutput>
</OutputDestination>
```

The above setting results in a sub-directory being created for each classification. Each classification directory contains XML files created with the object's [ID](#) matching the classification.

- Produce Revisions in Output File Names:

If you create an output file per object/document, the Gateway will by default put all revisions of a document into a single file. For example, the following settings produce an output file for all revisions of a document:

```
<OutputDestination Destination="FileSystem"
FolderStructure="CustomFolderStructureForOutput">
  <S3>
    <Authentication instance="true">
      <CredentialFile path="default" profileName="default" />
    </Authentication>
    <Region />
    <BucketName />
    <OutputDirectory />
  </S3>
  <FileSystem>
    <OutputPath> C:\Test \TestProject\Output</OutputPath>
  </FileSystem>
  <keepInputFolderStructureForOutput />
  <CustomFolderStructureForOutput>
    <FileName value="[eiwm:id] " />
    <FolderStructure value="[eiwm:ClassID]" />
  </CustomFolderStructureForOutput>
</OutputDestination>
```

If you want to have a single output file for each revision, then the configuration section must have the revision added to it:

```
<OutputDestination Destination="FileSystem"
FolderStructure="CustomFolderStructureForOutput">
  <S3>
    <Authentication instance="true">
      <CredentialFile path="default" profileName="default" />
    </Authentication>
    <Region />
    <BucketName />
    <OutputDirectory />
  </S3>
  <FileSystem>
    <OutputPath> C:\Test \TestProject\Output</OutputPath>
  </FileSystem>
  <keepInputFolderStructureForOutput />
  <CustomFolderStructureForOutput>
    <FileName value="[eiwm:id]_[eiwm:revision] " />
    <FolderStructure value="[eiwm:ClassID]" />
  </CustomFolderStructureForOutput>
</OutputDestination>
```

- An example configuration that creates an output folder for the nested context in a data source:

```
<OutputDestination Destination="FileSystem"
FolderStructure="CustomFolderStructureForOutput">
  <S3>
    <Authentication instance="true">
      <CredentialFile path="default" profileName="default" />
    </Authentication>
    <Region />
    <BucketName />
    <OutputDirectory />
  </S3>
  <FileSystem>
    <OutputPath>C:\Test\TestProject\Output</OutputPath>
  </FileSystem>
  <keepInputFolderStructureForOutput />
  <CustomFolderStructureForOutput>
    <FileName value="[eiwm:id]" />
    <FolderStructure value="[eiwm:context]" />
  </CustomFolderStructureForOutput>
</OutputDestination>
```

In this case, the value for `[eiwm:context]` is used in the folder structure. If its value is **"HigherContext"**, then a folder `C:\Test\TestProject\Output\HigherContext` will be created for the output files. If the context is nested, then a folder structure will be created, for example, `C:\Test\TestProject\Output\HigherContext\LowerContext`, corresponding to the object's context in the EIWM. For example:

```
<Context>
  <ID> HigherContext </ID>
  <Context>
    <ID> LowerContext </ID>
  </Context>
</Context>
```

Custom Folder Structure

You can manage the EIWM files by specifying a customized **Folder Structure**. Smaller EIWM files are needed to allow more efficient imports or when large EIWM files (approximately > 60 MB) fail to import via the AIM Import Controller.

The base output path for **FileSystem** is defined by the **Output Path**. The base **Output Path** for S3 is defined by the **Output Folder**.

Expected Values:

The expected values in a Folder Structure are listed in the following table:

Expected Entries in a Folder Structure	Description
Engineering attributes	<p>Attributes extracted from the data source along with the attributes created during the base mapping can be used as elements in the folder structure, for example, [Tag], [Custom Attribute] and so on. Also reserved attributes, such as [TemplateID], [ObjectID], [ClassID], [ContextID], [ObjectName] and [Revision], can be used.</p> <hr/> <p>Note: The nested context must be resolved as separate sub folders in the hierarchy.</p>
EIWM reserved attributes	<p>EIWM reserved attributes can be used as elements in the folder structure:</p> <p>[eiwm:id], [eiwm:classid], [eiwm:name], [eiwm:revision] or [eiwm:context]</p>
Manifest attributes	<p>For more information about manifest attributes, see Manifest References ().</p>
Reserved placeholders	<p>Reserved placeholders can be used as elements in the folder structure:</p> <ul style="list-style-type: none"> • [Date]: Provides the value of the current date. Default format for date is dd-MM-yyyy. This attribute provides a format string, for example, [Date{d.MMM.yy}]. • [Time]: Provides the value of the current time. Default format for date is hh-mm-ss. This attribute provides a format string, for example, [Time{H.mm}]. • [DateAndTime]: Provides the value of the current date and time. Default format for date is dd-MM-yyyy hh-mm-ss. This attribute provides a format string, for example, [Date{d.MMM.yy H:mm}]. <hr/> <p>Note: The formats can be from any of the formats as specified under the MSDN DateTime Formats.</p>
Static values	<p>The static values are simple text values and are left unchanged and can be defined without any square brackets.</p>

Expected Entries in a Folder Structure	Description
	Example: <code><FileName value="output_[eiwm:id].xml" /></code> . Here "output_" is a static value.

Notes:

- Every "\" in the folder hierarchy is treated as a delimiter and is resolved as a subfolder.
- Configured attributes that resolve to an empty value are skipped during the path generation.
- The values that are not resolved are skipped and a warning is logged.
- Characters, such as '/', ':', '*', '?', '"', '<', '>' and '|', are not allowed and are skipped if found in the resolved attribute values or in the static values. A warning is logged.
- Attributes that need to be resolved must be written within square brackets, for example, [name], [id] or [DateAndTime].
- **Create Document Object:** A Document Object is the metadata object present in the EIWM that describes the input data source. It holds the information about the data source for extraction like filename, date and time of extraction and so on. Default value of createDocumentObject option is true.

The configuration entry looks as follow.

```
<configuration>
...
<createDocumentObject apply="true" />
...
</configuration>
```

This option when set to 'true' allows you to include the document object and its related association "is referenced in" with all other objects in the output EIWM file.

When set to 'false', it directs the EIWM Loader not to include the Document object and its related association "is referenced in" from the Object Model.

Note: You can modify the same setting from both configuration and the EIWM Loader GUI page.

- **Project Context:** By default, it is set to [Avngate](#). If the project context is undefined in the base mapping, the EIWM objects will be written to an `<output>_avngate.xml` file and each object's context will be set to [Avngate](#). This allows for pre-processing by the AIM Import Controller to overwrite [Avngate](#) with a Vnet file definition for the project context (refer to the AIM User Guide).

If the project context value is left blank or set to any other value, the EIWM objects will be written to an `<output>_null.xml` file and each object's context will be set to empty if undefined in the base mapping.

- **Generate Trigger File:** After output file generation, to trigger the Import Controller, a **trigger.start** file is created in the staging area. It passes through three stages: start, process and finish. When the file reaches the finish stage, the output files are imported into AIM.

Note: Each loader has the option to generate a trigger file in the output folder defined for that loader and must be independently selected to generate it. This means that when a common folder is being used for more than one loader then they need to be set to be consistent with each other, otherwise ambiguous imports into AIM may result. Note that the trigger file is created after the loader output file, which may occur before the other loader output file is created. In this case a second trigger file would be created, which the Import Controller will then convert to a trigger queue file.

- **Create Alias in EIWM Objects:** Select this option to create an alias in the EIWM objects from the Gateway. You can configure this using the `createAlias` element, for example, `<createAlias apply="false" attribute="" />`.

Note: The `apply` value must be an `xsd` schema supported Boolean value. The valid values for `xsd:boolean` are `true`, `false`, `0` and `1`. Attribute values that are capitalized (for example, `TRUE`) or abbreviated (for example, `T`) are not valid.

Alias Attribute: If you select **Create Alias in EIWM Objects**, you can enter the name of the attribute that is used for the alias. Default value for this option is empty.

Notes:

- Alias is created only when the value of the `Alias` attribute defined in the loader configuration differs from the value of the `Object ID` in the mapping file.
- No aliases will be created when the value of the `apply` attribute is set to `false`.
- When the value of an `attribute` is set to either empty or invalid, the default `Alias` attribute used is `"GlobalID"`. No aliases will be created if the attribute `"GlobalID"` is not present.

Examples:

- If the source system has an object with attributes `"GlobalID"` and `"AlternateID"` with values `"3$pEhtFpv31QFndkpGikoC"` and `"X 101"`, respectively, the Load configuration file and mapping configuration file are set to the following values:

Load Configuration file:

```
<outputEIWM>
.....
<createAlias apply="true" attribute ="AlternateID" />
.....
</outputEIWM>
```

Mapping configuration is defined as:

```
<Object>
  <ObjectID value="[GlobalID]" />
  <ClassID value="PUMP" />
</Object>
```

This allows the loader to create aliases in the output EIWM objects, where the associated objects have the IDs per alias attribute defined in the configuration, `"X 101"` in this case.

The output object should look similar to the following, where the IDs of associated objects are the values of attribute defined in the `Alias` attribute field in the GUI or the value in the `createAlias` element in the Load configuration XML.

```
<Object>
<ID>3$pEhtFpv31QFndkpGikoC</ID>
  <Context>
    <ID>Avngate</ID>
  </Context>
<ClassID>PUMP</ClassID>
<Association type="is identified by">
  <Object>
    <ID>X 101</ID>
    <Context>
      <ID>Avngate</ID>
```

```

        </Context>
        <ClassID>PUMP</ClassID>
    </Object>
</Association>
.....
</Object>

```

If the source system has an object with an attribute "GlobalID" having a value "3\$pEhtFpv31QFndkpGikoC" and does not contain the attribute "AlternateID", the Load configuration file and mapping configuration file are set to the following values (when Alias attribute is not present in the source system):

Load configuration file:

```

<outputEIWM>
.....
<createAlias apply="true" attribute ="AlternateID" />
.....
</outputEIWM>

```

Mapping configuration is defined as:

```

<Object>
    <ObjectID value="[GlobalID]_AVA" />
    <ClassID value="PUMP" />
</Object>

```

The output object should look similar to the following, where the default alias attribute "GlobalID" is used to create the alias, as the Alias attribute "AlternateID" is not present in that particular object on the source system.

```

<Object>
    <ID>3$pEhtFpv31QFndkpGikoC_AVA</ID>
    <Context>
        <ID>Avngate</ID>
    </Context>
    <ClassID>PUMP</ClassID>
    <Association type="is identified by">
        <Object>
            <ID>3$pEhtFpv31QFndkpGikoC</ID>
            <Context>
                <ID>Avngate</ID>
            </Context>
            <ClassID>PUMP</ClassID>
        </Object>
    </Association>
    .....
</Object>

```

- If the source system has an object with attribute "GlobalID" having a value "3\$pEhtFpv31QFndkpGikoC" and the Load configuration file and mapping configuration file are set to the following values:

Load configuration file:

```

<outputEIWM>
.....
<createAlias apply="true" attribute ="GlobalID" />

```

```
.....
</outputEIWM>
```

Mapping configuration is defined as:

```
<Object>
  <ObjectID value="[GlobalID]" />
  <ClassID value="PUMP" />
</Object>
```

The output object should look similar to the following, where no alias is created, as both the **ObjectID** and **Alias** attribute are mapped to the same attribute "GlobalID".

```
<Object>
  <ID>3$pEhtFpv31QFndkpGikoC</ID>
  <Context>
    <ID>Avngate</ID>
  </Context>
  <ClassID>PUMP</ClassID>
  .....
</Object>
```

- **Save Settings:** After you have selected the required settings, click **Save Settings** to save the **Engineering** project settings.

The following code shows an example of Load configuration:

```
<configuration xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://
www.w3.org/2001/XMLSchema-instance"
sourceProductName="AVEVA.NET.Gateway.SmartPlant3D" componentName="Load"
componentVersion="2.2.0.0" >
  <components>
    <component name="LoadEIWM" locator=".\\LoadEIWMConfiguration.xml" />
    <component name="LoadXGL" locator=".\\LoadXGLConfiguration.xml" />
    <component name="LoadCSV" locator=".\\LoadCSVConfiguration.xml" />
  </components>
</configuration>
```

The following code shows an example of Load EIWM configuration:

```
<configuration xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://
www.w3.org/2001/XMLSchema-instance"
sourceProductName="AVEVA.NET.Gateway.SmartPlant3D"
componentName="LoadEIWM" componentVersion="2.2.0.0" >
  <OutputDestination Destination="S3|FileSystem|S3&FileSystem">
    <S3>
      <Authentication instance="true||false">
        <CredentialFile path="default" profileName="default"/>
      </Authentication>
      <Region></Region> <!-- Required, defines region of the
destination S3 bucket -->
      <BucketName></BucketName> <!-- Required, defines the
destination S3 bucket -->
      <OutputDirectory></OutputDirectory> <!-- Optional, If not
defined files will be uploaded to the base of S3 bucket -->
    </S3>
    <FileSystem>
      <OutputPath></OutputPath>
    </FileSystem>
  </OutputDestination>
</configuration>
```

```

</OutputDestination>
<dataType value="EIWM/NONE"/>
<keepInputFolderStructureForOutput apply="true" />
<generateTriggerStart apply="true" />
<addDescriptionToHeader apply="true" />
<projectContext value="Avngate" />
<createAlias apply="true" attribute="GlobalID" />
<file maxObjectCount="0" suffix="_part_" suffixNumberFormat="DDD" />
<annotations level="Basic" />
</configuration>

```

Note: You can set tracking of changes provided by EIWM Load on data using annotations feature. For information about annotations, see [Appendix C: Tracking Changes in Data](#) ().

Output File Format

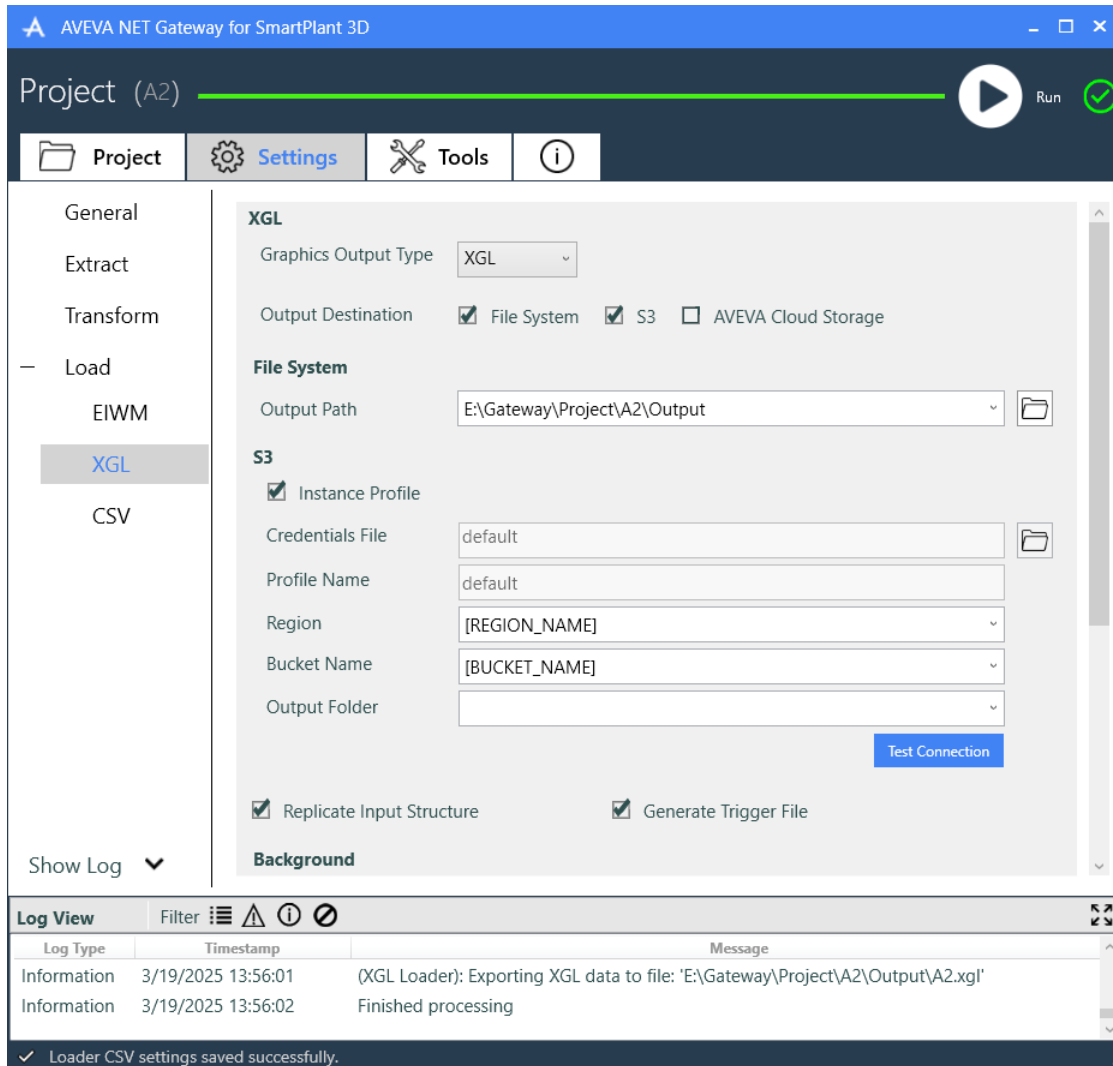
The output setting allows you to set the format and naming convention of the output XML. This setting can be modified using the below-mentioned attributes in the Loader configuration file: `<file maxObjectCount="0" suffix="_part_" suffixNumberFormat="DDD" />`

You can modify the output file formatting as follows:

- **Splitting Output File:** You can split the output across multiple files based on the number of objects identified during conversion as valid EIWM objects. This output setting can be modified using `<file maxObjectCount="0">` attribute in the configuration file. The `<file maxObjectCount=` parameter accepts the value as non-negative integers. This parameter defines the maximum number of objects permitted to be serialized in a single file. If the object count exceeds the count defined in the setting, the objects will be serialized across multiple output files such that no file contains more objects than set by this setting. The `maxObjectCount` value = "0" is the default value and it defines that all the objects should be serialized into a single output file.
- **Suffix for multipart output files:** You can set the suffix to be appended to the multipart output files. Suffix will only be appended to the file as the output file is split based on the exceeding `maxObjectCount` value or based on the multiple template mapped to the EIWM objects. This setting can be modified using `<file ... suffix="_part_"` attribute in the loader configuration setting. The output for above-defined setting is as follows:
 - `Wall_part_001_null.xml`
 - `Wall_part_002_null.xml`
- **Suffix number format:** You can set the serial number format as required. This setting can be accessed using `<file ... suffixNumberFormat="DDDD">` attribute in the configuration setting. Using the above settings, the output files should be as follows:
 - `Wall_part_0001_null.xml`
 - `Wall_part_0002_null.xml`

XGL

Load exports graphical data together with metadata to XGL/ZGL format. You can select the graphical data output type XGL/ZGL format under the **Load, XGL** settings.

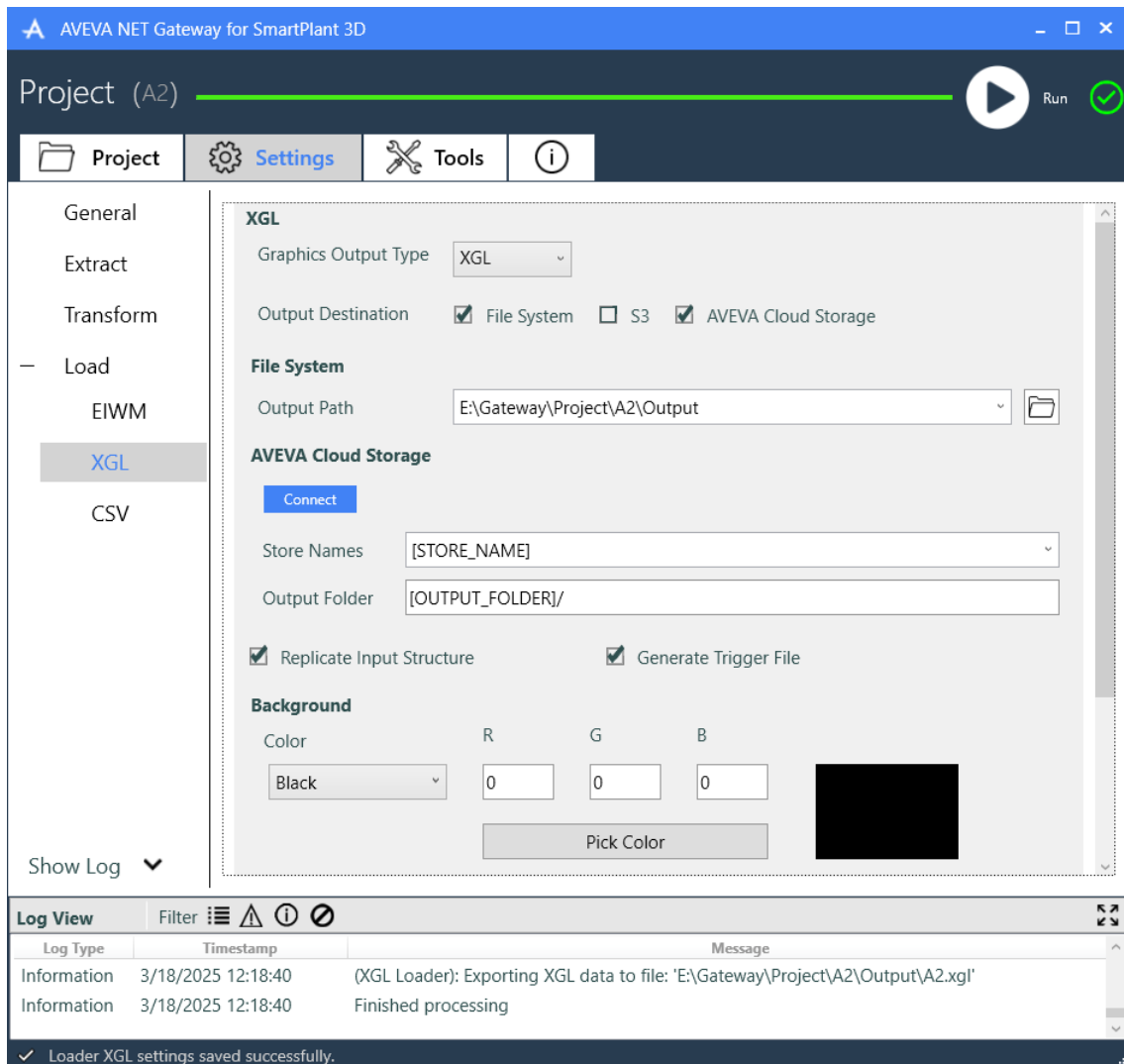


After you select the required settings, click **Save Settings** to save the Load geometry settings of graphical data.

The **XGL** setting contains the following options:

- **Graphics Output Type:** Select the output file type for storing the graphics, such as XGL (XML format)/ZGL (compressed XGL format). You can use **None** option if no 3D graphical data is needed, for example, if only the EIWM file is needed. Selecting **None** means that no graphical data is written, for example, if only the engineering data via EIWM XML is to be imported into AIM.
- **Output Destination:** Select whether the output will be written to a **File System** or an **S3** or AVEVA Cloud Storage. You can select all these types independently or select one of the S3 and AVEVA Cloud Storage type with Filesystem destination type together.
 - **File System:** The file system is normally the path that defines the AIM staging area.
 - **Output Path:** Type the Output file path or click **Browse for Folder** button to enter the output file path.
 - **S3:** To access an S3 bucket requires the relevant authentication details like credential file, region information, bucket name and output directory. For more information, see [Accessing an AWS S3 Bucket](#) ().

- **AVEVA Cloud Storage:** For more information about accessing AVEVA Cloud Storage, see [Accessing AVEVA Cloud Storage](#) ().



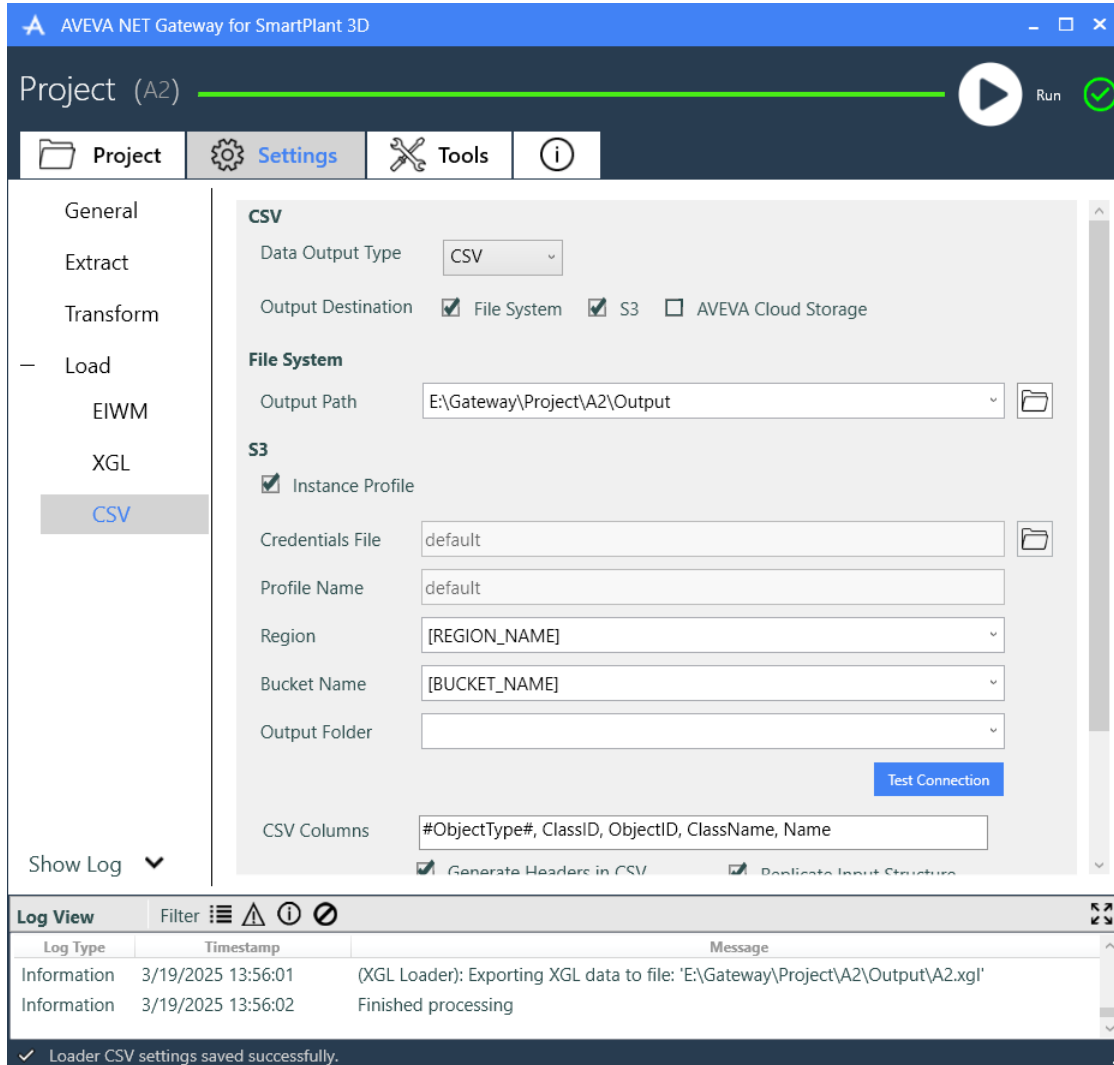
- **Background:** You can select the background colour of drawings. The background colour of the **XGL/ZGL** files generated by the project can be selected by any of these methods:
 - **Colour:** Select the required colour from the **Colour** list, for example, Black.
 - **R, G, B:** Enter the **RGB** values of the colour in the fields provided.
 - **Pick Colour:** Click **Pick Colour** and select the required colour from the standard colour selection dialog. A preview of the selected colour is displayed next to the **Pick Colour** button.
- **View Direction:** From the **View Direction** list, select the direction from which the drawings are initially viewed. You can select only one option from the list, for example, **View from the Front**.
- **Save Settings:** Click **Save Settings** to save the settings.

Note: You can set tracking of changes provided by XGL Load on data using annotations feature. For information about annotations, see [Appendix C: Tracking Changes in Data](#) ().

CSV

This option enables you to configure the **CSV** settings. The **CSV** setting contains the following options:

- **Output Destination:** Select the output destination type to store the output file type in the Destination folder. The destination type can be **Filesystem** or **S3** or **AVEVA Cloud Storage**. You can select all these types independently or select one of the S3 and AVEVA Cloud Storage type with Filesystem destination type together.
- **File System:** he file system is normally the path that defines the AIM staging area.
 - **Output Path:** Type the Output file path or click Browse for Folder tab to enter the output file path.



AVEVA NET Gateway for SmartPlant 3D

Project (A2) Run

Project Settings Tools

General
Extract
Transform
Load
EIWM
XGL
CSV

CSV

Data Output Type: CSV

Output Destination: ☒ File System ☒ S3 ☐ AVEVA Cloud Storage

File System

Output Path: E:\Gateway\Project\A2\Output

S3

☒ Instance Profile

Credentials File: default

Profile Name: default

Region: [REGION_NAME]

Bucket Name: [BUCKET_NAME]

Output Folder:

Test Connection

CSV Columns: #ObjectType#, ClassID, ObjectID, ClassName, Name

☒ Generate Headers in CSV ☒ Replicate Input Structure

Show Log

Log View Filter

Log Type	Timestamp	Message
Information	3/19/2025 13:56:01	(XGL Loader): Exporting XGL data to file: 'E:\Gateway\Project\A2\Output\A2.xgl'
Information	3/19/2025 13:56:02	Finished processing

✓ Loader CSV settings saved successfully.

- **S3:** To access an S3 bucket requires the relevant authentication details like credential file, region information, bucket name and output directory. For more information about **S3** Bucket Details, see [Accessing an AWS S3 Bucket](#) ().

The screenshot shows the AVEVA NET Gateway for SmartPlant 3D application window. The title bar reads "AVEVA NET Gateway for SmartPlant 3D". The main window has a dark blue header with "Project (A2)" and a "Run" button with a green checkmark. Below the header is a navigation bar with icons for Project, Settings (selected), Tools, and Information. The left sidebar contains a tree view with options: General, Extract, Transform, Load, EIWM, XGL, and CSV (selected). The main content area is titled "CSV" and contains the following settings:

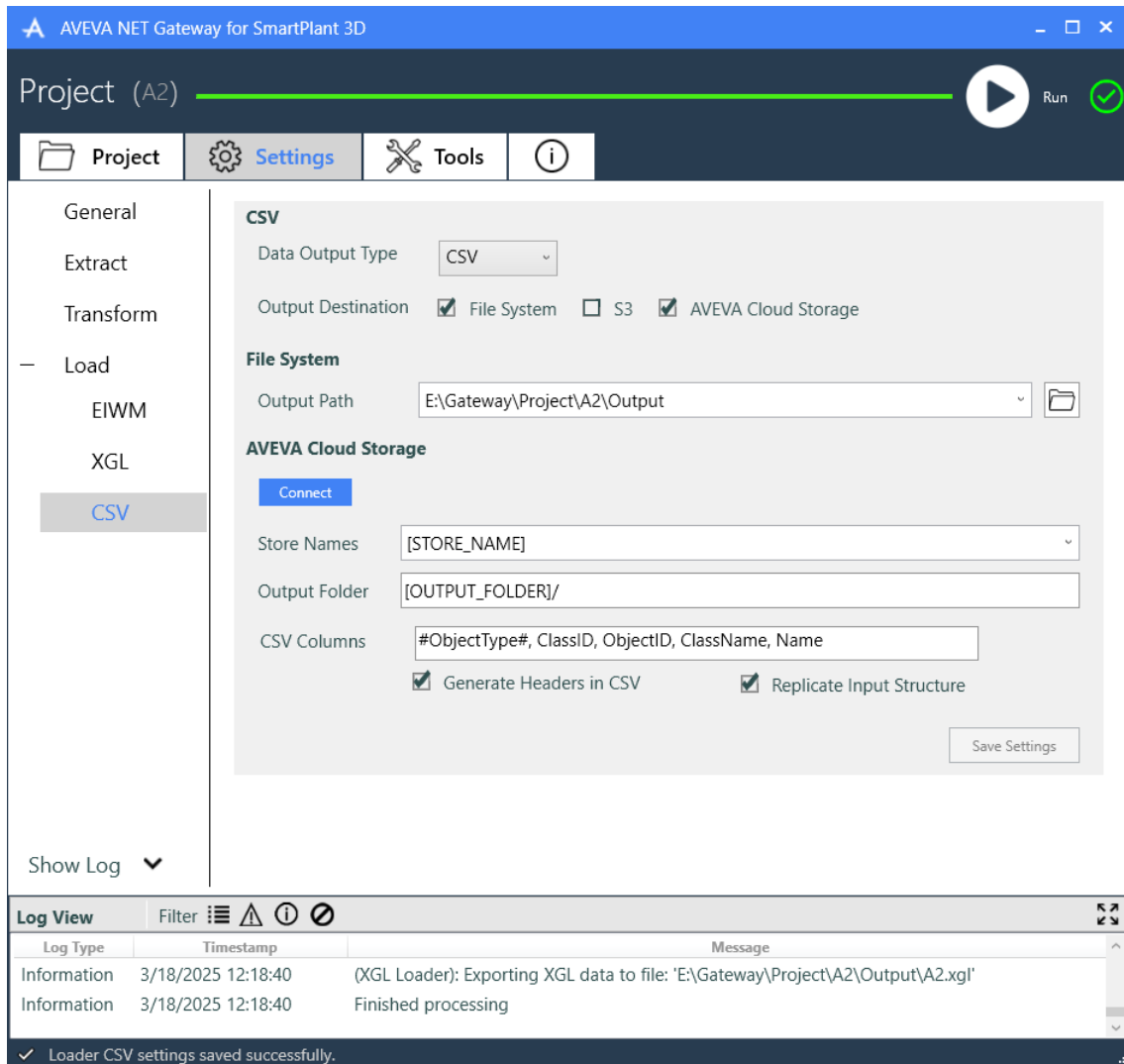
- Data Output Type:** CSV (dropdown)
- Output Destination:** ☒ File System, ☐ S3, ☒ AVEVA Cloud Storage
- File System:**
 - Output Path:** E:\Gateway\Project\A2\Output (with a folder icon)
- AVEVA Cloud Storage:**
 - Connect:** (button)
 - Store Names:** [STORE_NAME] (dropdown)
 - Output Folder:** [OUTPUT_FOLDER]/ (text field)
 - CSV Columns:** #ObjectType#, ClassID, ObjectID, ClassName, Name (text field)
 - ☒ Generate Headers in CSV
 - ☒ Replicate Input Structure
 - Save Settings:** (button)

At the bottom left of the main content area is a "Show Log" button with a dropdown arrow. Below the main content area is a "Log View" section with a filter icon and a table of log messages:

Log Type	Timestamp	Message
Information	3/18/2025 12:18:40	(XGL Loader): Exporting XGL data to file: 'E:\Gateway\Project\A2\Output\A2.xgl'
Information	3/18/2025 12:18:40	Finished processing

At the very bottom of the window, a status bar shows a green checkmark and the message: "Loader CSV settings saved successfully."

- **AVEVA Cloud Storage:** For more information about accessing AVEVA Cloud Storage, see [Accessing AVEVA Cloud Storage](#) ().



- **CSV Columns:** In the **CSV Columns** field, add the additional comma separated columns to export the objects' attributes and associations in the CSV export.
- **Generate Headers in CSV:** If you select this setting then the names of the attributes or associations are written to the first row of the CSV file. The header level information contains configuration file data and date of creation and so on.
- **Replicate Input Structure:** If you select this option, the output file is generated in a duplicate path. This is replicated input folder structure in output folder if you point on input location of **input** data containing also subfolders with files of such type.
- **Save Settings:** After you have selected the required settings, click this to save the CSV settings.

Execute Projects

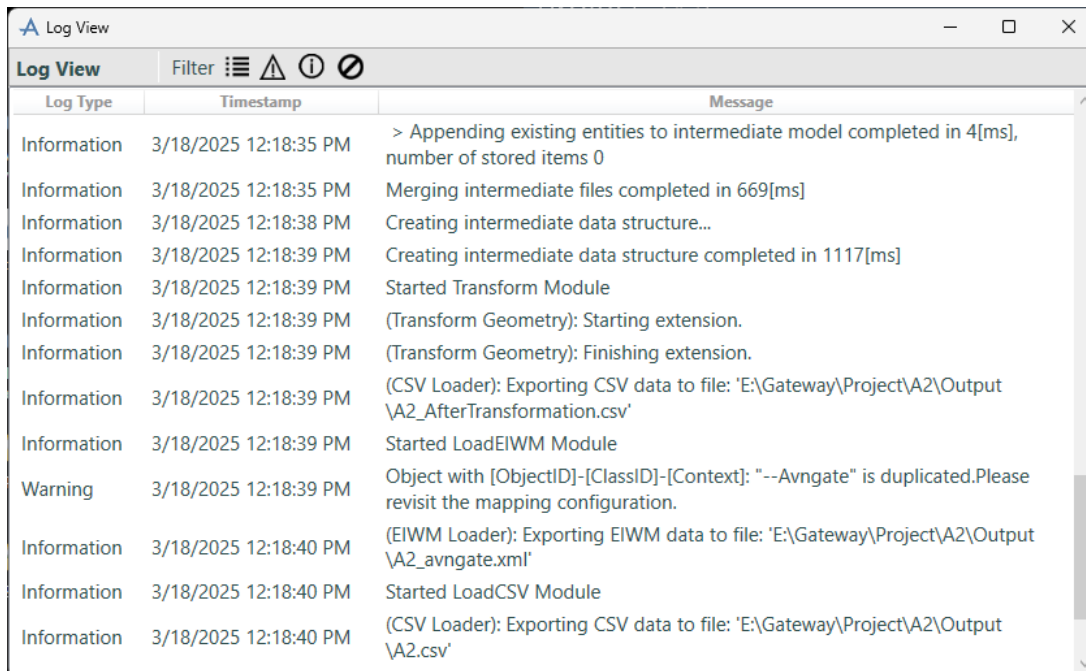
After you have selected the required project settings you can execute the project. While executing the project configuration, the Gateway searches for the input location and generates output files.

To execute the selected project:

1. Click **Save Settings** to save the configuration settings of **Extract, Transform** and **Load**.
2. After the successful saving of the configuration files, it enables the **Run** button.
3. Click **Run** to execute the project.

Note: The input data in the Input Location, as defined in **Extract** page, are processed and the resulting output files written to the output location. After the project has been executed successfully, the **LOG** button is enabled.


Show Log: Allows you to view the log information, which the application generates during project execution. Click **Show Log** to retrieve the Log information containing log type, date, time and message, as shown below:



Log Type	Timestamp	Message
Information	3/18/2025 12:18:35 PM	> Appending existing entities to intermediate model completed in 4[ms], number of stored items 0
Information	3/18/2025 12:18:35 PM	Merging intermediate files completed in 669[ms]
Information	3/18/2025 12:18:38 PM	Creating intermediate data structure...
Information	3/18/2025 12:18:39 PM	Creating intermediate data structure completed in 1117[ms]
Information	3/18/2025 12:18:39 PM	Started Transform Module
Information	3/18/2025 12:18:39 PM	(Transform Geometry): Starting extension.
Information	3/18/2025 12:18:39 PM	(Transform Geometry): Finishing extension.
Information	3/18/2025 12:18:39 PM	(CSV Loader): Exporting CSV data to file: 'E:\Gateway\Project\A2\Output\A2_AfterTransformation.csv'
Information	3/18/2025 12:18:39 PM	Started LoadEIWM Module
Warning	3/18/2025 12:18:39 PM	Object with [ObjectID]-[ClassID]-[Context]: "--Avngate" is duplicated.Please revisit the mapping configuration.
Information	3/18/2025 12:18:40 PM	(EIWM Loader): Exporting EIWM data to file: 'E:\Gateway\Project\A2\Output\A2_avngate.xml'
Information	3/18/2025 12:18:40 PM	Started LoadCSV Module
Information	3/18/2025 12:18:40 PM	(CSV Loader): Exporting CSV data to file: 'E:\Gateway\Project\A2\Output\A2.csv'

Notes:

- You can click the four icons on the **Log View** window to get the respective log messages about **Error**, **Warning**, **Information** and **Verbose** based on your selection in the **Logging** field.

You can also click  to open this Log information in another pop-up window.

- By default, the messages relating to Information level is displayed if you do not select any of the four log level options.
- Timestamp is based on the defined system time set.

Project Version Upgrade

Automatic configuration upgrades are not supported from 5.0.x versions to 6.0.x versions, due to the difference in mapping syntax.

If you open the old project 6.0.x versions, the upgrade mechanism converts them into the latest project versions and creates a copy of the project file if you do not have any. This mechanism works both in GUI mode and in the command line mode.

All the original files are stored in the project directory with the `.bkp` extension. To use these backed up files with the previous version of the Gateway, remove the upgraded files and change the names of the backup files to their original names.

Note: The upgrade report can be found in the `upgrade_log.txt` file, placed in a new project location.

For upgrading the projects from the command line, see the [Run the Gateway from the Command Line](#).

Access an AWS S3 Bucket

An **S3** is a container for objects stored in Amazon Web Service S3. Every object is contained in a bucket.

S3 Bucket Details

Access Authorization to connect to S3 services is granted by one of two possible methods:

- via the EC2 Instance Profile: Select this check box to ensure that the EC2 server on which the Gateway is run has an instance profile to access the S3 bucket, using the Identity Access Management (IAM) role attached to the EC2 instance. You can configure the same using the `instanceProfile` attribute of the element `<Authentication instanceProfile="false">`.

Note: The `instanceProfile` value must be an xsd schema supported Boolean value. The valid values for `xsd:boolean` are `true`, `false`, `0` and `1`. Values that are capitalized (for example, `TRUE`) or abbreviated (for example, `T`) are not valid.

- via the user's Credential File
 - Define the **location and name** (on your local server) of the Credential File to make requests to AWS.

Note: If Instance Profile value is set to `true`, then the **Credential File** path value must point to a valid AWS Credential file.

- Define the relevant Profile Name defined in the Credential File.

Note: If **Instance Profile** value is set to `true`, then **Profile Name** is a required field and cannot be empty.

Bucket Identification of where the file is located then needs to be provided via the following parameters:

- **Region:** Define the Region in which Amazon S3 Bucket is deployed.
- **Bucket Name:** (Case-sensitive) Define the name of the Bucket which hosts the file's location.
- File Description:
 - Object Key: (Case-sensitive) For Extractors, define this object key to read from S3 bucket. This should be the Object Key definition which may contain any folders and the file's name.
 - Output Folder: For Loaders, defining the full object key is not required. Select the output folder in which output EIWM//XGL/CSV file should be placed. This is optional to allow you to specify the folder part of the Object Key as the Gateway automatically generates the filename (normally derived from the input source or changed by the mapping configuration of the `#MODEL_NAME#` manifest attribute). If the output folder does not exist it will be created. When writing to a bucket this will just be the folder part of the Object Key as the filename is normally derived from the input source or changed by the Gateway's mapping configuration.

Note: When loading to S3 bucket the object key is not required and the object key is automatically set to the output name of EIWM (applicable only to the loader).

When you want to access an AWS S3 Bucket, you must authenticate your calls. It depends on where the Gateway is instantiated:

- When running the Gateway on an EC2 instance (that is, from within AWS): You should rely on the Instance profile for authentication of your calls to the S3 bucket.

In this case, the instance profile internally authenticates the call for the user (that is, without the use of any credential file). Instance profiles use an EC2 attached role to authenticate calls to other Amazon Resource Names (ARN).

- When running the Gateway on their own server (that is, from outside AWS): You should rely on the credential file for authentication of their calls to the S3 bucket. A credential file is stored on your system and it contains AWS credentials for accessing AWS resources.

- The Gateway also supports the temporary credentials. The format for temporary credential file is as follows:

```
*[aveva-iam-user]
region=YOUR_REGION_HERE
aws_access_key_id=YOUR_ACCESS_KEY_HERE
aws_secret_access_key=YOUR_SECRET_KEY_HERE
[default]
source_profile=aveva-iam-user
role_arn=arn:aws:iam::YOUR_ACCOUNT_NO_HERE:role/YOUR_ROLE_NAME_HERE
region=YOUR_REGION_HERE*
```

- You can set your credentials in the AWS credentials profile file on your local Windows system, located at:

`C:\Users\USERNAME \.aws\credentials`

- You can save the credential file format without any extension or with only .txt extension. For more information, refer to the relevant AWS documentation on AWS command line interface.
- There can be multiple profiles in a credential file. You must be very careful when handling a credential file. It must never be shared.
- When reading or writing to S3 buckets, files can be located in folders. In this case, the folder name must be specified in the configuration.

Note: For security reasons, it is recommended to restrict a user's access to only those AWS resources required by their IAM role. For example:

1. Create an IAM user whose policy does not have access to any AWS resource.
2. Create an IAM role with the following policy to access a specific Amazon S3 Bucket:

```
{
  "Version": "2012-10-17",
  "Statement": [
    { "Effect": "Allow", "Action": [ "s3:ListBucket" ], "Resource": [ "arn:aws:s3::<s3-
bucket-name>" ] }
    ,
    { "Effect": "Allow", "Action": [ "s3:PutObject", "s3:GetObject", "s3:DeleteObject",
"s3:PutObjectAcl" ], "Resource": [ "arn:aws:s3::<s3-bucket-name>/*" ] }
  ]
}
```

Note: The above example is the minimum recommended configuration. If it is not suitable for your environment, you should investigate what is the most suitable approach for your environment.

3. Assign this role to the IAM user created in Step 1 above.

You can also define a bucket policy to restrict access to an S3 bucket. For more information, refer to the relevant AWS documentation on how to restrict access to S3 buckets.

Access AVEVA Cloud Storage

Prerequisites:

To access AVEVA Cloud Storage (ACS), you need to have a Connect account with the store(s) that will contain the input or output files.

Ensure that the following system environment variables are defined:

Variable	Value
AVEVA_CLOUD_AUTHORITYURI	https://signin.connect.aveva.com
AVEVA_CLOUD_ENVIRONMENT	prod
AVEVA_CLOUD_LOGOUTURI	https://signin.connect.aveva.com/v2/logout

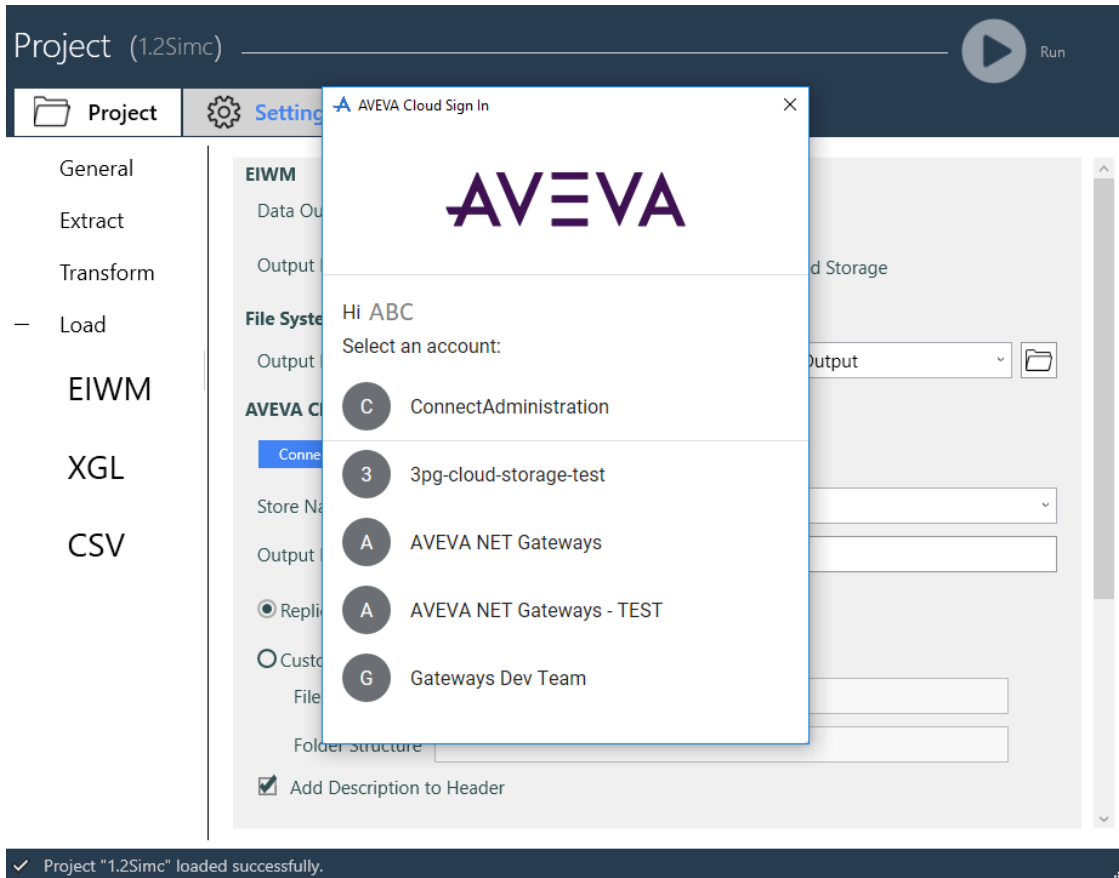
Note: The above-listed environment variables are mandatory for connecting to ACS.

ACS Gateway Configuration:

In the relevant AVEVA Cloud Storage section of the Extract and/or Load configuration, select the **Store Names** combo box.

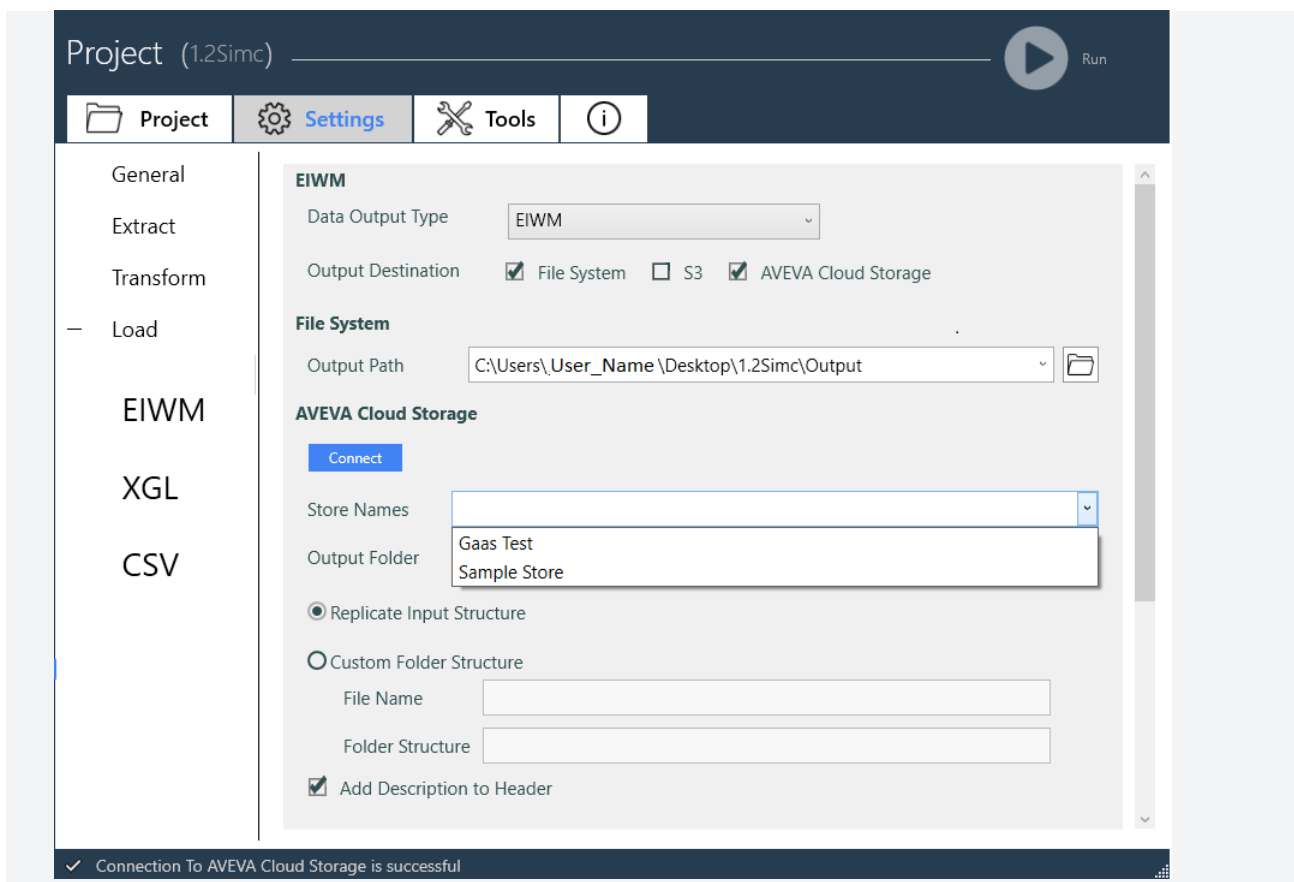
When configuring for the first time, the AVEVA Cloud Sign In window appears.

1. Select the relevant account that contains the Store (folder) that will be used. When you select this account, it authenticates and establishes a connection with AVEVAConnect and a token is downloaded into Userprofile/.aveva folder with the client ID ending with the word "Credential", for example, **AVEVA.Gateway.SmartPlant3D.ACS_credentials**.



Notes:

- It is assumed that this one account will allow access to all of the ACS folders required by the Gateway.
 - The refresh token normally has a lifetime of 6 months. When a token becomes expired, you will get the following error message: "Existing token may be expired. Please use Connect button from Loader pages in GUI mode to create credentials file."
1. After the connection is successful, the below status bar displays the message **Connection to AVEVA Cloud Storage is successful** and **Store Names** will contain the list of available stores.



2. Select the relevant ACS Store.
3. Save the changes.

The Loaders configuration files such as EIWM, XGL, and CSV files are stored in the Outputs folder in the following configuration format:

```
<Outputs>
  <S3 isSelected="false">
    <Authentication instance="true">
      <CredentialFile path="default" profileName="default" />
    </Authentication>
    <Region />
    <BucketName />
    <OutputDirectory />
  </S3>
  <FileSystem isSelected="true">
    <OutputPath>C:\Output</OutputPath>
  </FileSystem>
</AvevaCloudStorage isSelected="false"><StoreName /><OutputDirectory /></AvevaCloudStorage></Outputs>
<OutputOptions FolderStructure="keepInputFolderStructureForOutput">
  <keepInputFolderStructureForOutput />
  <CustomFolderStructureForOutput>
    <FileName value="" />
    <FolderStructure value="" />
  </CustomFolderStructureForOutput>
</OutputOptions>
<dataType value="EIMM" />
<createDocumentObject apply="true" />
<generateTriggerStart apply="true" />
<addDescriptionToHeader apply="true" />
<projectContext value="Avngate" override="false" />
<createAlias apply="false" attribute="" />
<file maxObjectCount="0" suffix="_part_" suffixNumberFormat="DDD" />
<annotations level="Basic" />
</configuration>
```

The following are the different conditions of output file destinations and the related loader configurations:

- If you select only the FileSystem, the loader configuration will be as follows:

```
<OutputDestination Destination="FileSystem">
  <S3>
    <Authentication instance="true">
      <CredentialFile path="default" profileName="default" />
    </Authentication>
    <Region />
    <BucketName />
    <OutputDirectory />
  </S3>
  <FileSystem>
    <OutputPath>C:\Users\User_Name\Desktop\1.2Simc\Output</OutputPath>
  </FileSystem>
</OutputDestination>
```

- If you select only S3, the loader configuration will be as follows:

```
<OutputDestination Destination="S3">
  <S3>
    <Authentication instance="true">
      <CredentialFile path="default" profileName="default" />
    </Authentication>
    <Region>TestRegion</Region>
    <BucketName>S3Output</BucketName>
    <OutputDirectory>C:\Users\User_Name\Desktop\1.2Simc\Output</OutputDirectory>
  </S3>
  <FileSystem>
    <OutputPath>C:\Users\User_Name\Desktop\1.2Simc\Output</OutputPath>
  </FileSystem>
</OutputDestination>
```

If you select both Filesystem and S3, the loader configuration will be as follows:

```
<OutputDestination Destination="Both">
  <S3>
    <Authentication instance="true">
      <CredentialFile path="default" profileName="default" />
    </Authentication>
    <Region>TestRegion</Region>
    <BucketName>S3Output</BucketName>
    <OutputDirectory>C:\Users\User_Name\Desktop\1.2Simc\Output</OutputDirectory>
  </S3>
  <FileSystem>
    <OutputPath>C:\Users\User_Name\Desktop\1.2Simc\Output</OutputPath>
  </FileSystem>
</OutputDestination>
```

- When you create a new project in the Gateway, then FileSystem is the default output destination.

Note: You can select multiple output destinations simultaneously such as File System and ACS or ACS and S3 or all the three output destinations together.

Chapter 3 Run the Gateway from the Command Line

Command Line parameters override the values set in the project configuration but don't change the project setting values.

At the command line, type the following:

```
"C:\Program Files\AVEVA\AVEVA NET Gateways\Gateway For SmartPlant
3D\AVEVA.NET.Gateways.SG3D.exe" <-cp> <-lp> <-op> <-extcp> <-trnsfcp> <-ldrcp ><-EXTPT> <-
EXTNM> <-EXTMD> <-EXTRT> <-EXTFL> <-to> <-MA> <-CONTEXT>
```

where the parameters are listed as follows:

Command Line Parameter	Description
-cp <project configuration file path>	(mandatory) Project configuration file path, for example, "C:\Users\<username>\Gateway for SmartPlant 3D Projects\A1".
-lp <log folder path>	(optional) Log folder path.
-op <output folder path>	(optional) Output folder path for all the the output EIWM/XGL/CSV files.
-extcp <configuration file path for Extract configuration>	(optional) Configuration file path of Extract configuration.
-trnsfcp <configuration file path for Transform configuration>	(optional) Configuration file path of Transform configuration.
-ldrcp <configuration file path for Load configuration>	(optional) Configuration file path of Load configuration.
-EXTPT <path for intermediate model file>	(optional) Overrides the path of the intermediate extraction file defined in the extractor configuration.
-EXTNM <name of generated output file>	(optional) Overrides the name of the intermediate extraction file defined in the extractor configuration.
-EXTMD <extraction mode>	(optional) Overrides the extraction mode – valid values are: NEW UPDATE EXISTING.
-EXTRT <set of items to be extracted (UID notation)>	(optional) Specifies the items to be extracted via their Smart 3D UID notation.
-EXTFL <name of Smart 3D filter>	(optional) Overrides the name of the Smart 3D filter to select items for extraction.
-to <time in minutes>	(optional) Timeout - Specifies the time after which the application stops processing data. Example for stop processing after 10 minutes: -to 10 . Refer to the General Settings page for more information.

Command Line Parameter	Description
<code>-ma <additional manifest attribute name></code>	(optional) Allows the passing of a manifest attribute value from the command line. You can specify any number of manifest attributes with this syntax through command line.
<code>-context <project context></code>	(optional) Sets the context for EIWM objects.

Notes:

- **Logging Functionality:** When you run the Gateway from command prompt, all processing messages are directed to the log file. This allows you to use the command line in batch mode, without waiting for the Gateway to complete the execution.

Prior to project execution (such as command line parameters validation logs or project configuration file load logs) Gateway messages are logged in the default log location:

```
C:\Users\<username>\AppData\Roaming\AVEVA\AVEVA NET Gateway for SmartPlant
3D\Logs\3DDataGateway_Log_<GUID>.txt
```

During project execution, Gateway messages are then logged in the project log file as defined either in the command prompt (see below) or in the configuration file.

- Any optional parameter used at the command line takes priority over the equivalent configuration in the project file. This is done without overwriting the value in project configuration file. Using these command line parameters means you can re-use the same configuration to extract data from the same model using different filters or UIDs without having to create separate Gateway project configurations.
- The Gateway validates the command line syntax and the contents of configuration files. If any errors are detected, processing will be terminated and the relevant message will be displayed on the console.

Examples:

The following examples assume that the Gateway is installed in the default directory "`C:\Program Files\AVEVA\AVEVA NET Gateways\Gateway for SmartPlant 3D`" and the Gateway project location is "`C:\Users\<username>\Gateway for SmartPlant 3D Projects\A1.xml`".

- Running project as it is.

```
@echo off
echo Running gateway...
set GATEWAY=C:\Program Files\AVEVA\AVEVA NET Gateways\Gateway for SmartPlant 3D
set APP=AVEVA.NET.Gateways.SG3D.exe
@echo on
call "%GATEWAY%\%APP%" -CP "C:\Users\<username>\Gateway for SmartPlant 3D
Projects\A1.xml"
@echo off
echo Completed
@echo on
```

The example above shows how to run the Gateway project from the command line. The project will be executed as it is defined in the configuration files via the UI.

```
@echo off
echo Running gateway...
```

```
set GATEWAY=C:\Program Files\AVEVA\AVEVA NET Gateways\Gateway for SmartPlant 3D
set APP=AVEVA.NET.Gateways.SG3D.exe
@echo on
C:\Users\<username>\Gateway for SmartPlant 3D Projects\A1.xml
call "%GATEWAY%\%APP%" -CP "A1.xml"
@echo off
echo Completed
@echo on
```

This example differs from the previous one in how the path of the project is passed to the Gateway. In this example, it uses a relative path to the current directory.

- Changing the output path at runtime.

```
@echo off
echo Running gateway...
set GATEWAY=C:\Program Files\AVEVA\AVEVA NET Gateways\Gateway for SmartPlant 3D
set APP=AVEVA.NET.Gateways.SG3D.exe
@echo on
cd "C:\Users\<username>\Gateway for SmartPlant 3D Projects\A1.xml"
call "%GATEWAY%\%APP%" -CP "A1.xml" -OP "NewOutputFolder"
@echo off
echo Completed
@echo on
```

Parameter -OP is used to override output path defined in the Gateway project "A1". Note that the new path is provided relative to the current directory, so the final output folder:

```
"C:\Users\<username>\Gateway for SmartPlant 3D Projects\A1\NewOutputFolder"
```

- Changing the log path at runtime.

```
@echo off
echo Running gateway...
set GATEWAY=C:\Program Files\AVEVA\AVEVA NET Gateways\Gateway for SmartPlant 3D
set APP=AVEVA.NET.Gateways.SG3D.exe
@echo on
cd "C:\Users\<username>\Gateway for SmartPlant 3D Projects\A1"
call "%GATEWAY%\%APP%" -CP "A1.xml" -LP "Logs_new\001"
@echo off
echo Completed
@echo on
```

This example differs to the previous one by using the -LP parameter to change the location of the log files. The backslash symbol used in the path tells the Gateway to create a folder if it doesn't exist. The new folder location will be:

```
"C:\Users\<username>\Gateway for SmartPlant 3D Projects\A1\Logs_new\001"
```

- Working with extraction mode.

```
@echo off
echo Running gateway...
set GATEWAY=C:\Program Files\AVEVA\AVEVA NET Gateways\Gateway for SmartPlant 3D
set APP=AVEVA.NET.Gateways.SG3D.exe
cd "C:\Users\<username>\Gateway for SmartPlant 3D Projects\A1"
echo Initialize... %TIME%
```

```
call "%GATEWAY%\%APP%" -CP "A1.xml" -EXTMD "NEW" -EXTPT -OP "C-NW"
echo Finished "New" at %TIME%, return code %errorlevel%
call "%GATEWAY%\%APP%" -CP "A1.xml" -EXTMD "UPDATE" -EXTPT -OP "C-UP"
echo Finished "Update" at %TIME%, return code %errorlevel%
call "%GATEWAY%\%APP%" -CP "A1.xml" -EXTMD "EXISTING" -EXTPT -OP "C-EX"
echo Finished "Existing" at %TIME%, return code %errorlevel%
call "%GATEWAY%\%APP%" -CP "A1.xml" -EXTMD "blablabla" -OP "C-EX"
echo Finished "blablabla" at %TIME%, return code %errorlevel%
echo Completed
@echo on
```

The above example shows how to control the extraction mode. The same project is run four times with different extraction modes. Different processing times are required for each mode:

- **NEW:** The Gateway generates an intermediate file and produces the output files.
- **UPDATE:** to the Gateway updates an existing intermediate file with only the items that have changed since the last time it was created or updated, and then generates the output files.
- **EXISTING:** The Gateway only produces output files from the contents of the existing intermediate file.
- **Error:** the last call uses an invalid option and the Gateway will log an error and return an execution code other than 0 (0 means success that is described in the following Exit Codes section).
- Changing the extraction name (which changes the name of the output files) and the location of intermediate files.

```
@echo off
echo Running gateway...
set GATEWAY=C:\Program Files\AVEVA\AVEVA NET Gateways\Gateway for SmartPlant 3D
set APP=AVEVA.NET.Gateways.SG3D.exe
@echo on
cd "C:\Users\<username>\Gateway for SmartPlant 3D Projects\A1"
call "%GATEWAY%\%APP%" -CP "A1.xml" -EXTPT "MYPATH" -EXTNM "MYNAME"
@echo off
echo Completed
@echo on
```

The above example shows how to change the location of the intermediate files (-EXTPT extraction path) and how to change the name of the output files by changing the extraction name (-EXTNM). Path values can be provided either in absolute or relative form. This example uses relative form so the path will be generated in the current directory. If the extraction name contains symbols like slash or backslash then subfolder(s) will be created for intermediate and output files. If in the above example, the extraction name is expressed as -EXTNM "MY/NAME\AA" then the output file will be **AA.xml** and will be placed in a subfolder NAME of subfolder MY of the output folder defined in the project. For example, if the output folder defined in the project is "C:\Users\<username>\Gateway for SmartPlant 3D Projects\A1\Output" then the name of the output graphical file would be "C:\Users\<username>\Gateway for SmartPlant 3D Projects\A1\Output\MY\NAME\AA.xml".

- Creating output on basis of selected items.

```
@echo off
echo Running gateway...
set GATEWAY=C:\Program Files\AVEVA\AVEVA NET Gateways\Gateway for SmartPlant 3D
set APP=AVEVA.NET.Gateways.SG3D.exe
cd C:\Users\<username>\Gateway for SmartPlant 3D Projects\A1
```

```
@echo on
call "%GATEWAY%\%APP%" -CP "A1.xml" -EXTRT "{/1/1}" -EXTNM "A01U11"
call "%GATEWAY%\%APP%" -CP "A1.xml" -EXTRT "{/1/2}" -EXTNM "A01U12"
call "%GATEWAY%\%APP%" -CP "A1.xml" -EXTRT "{/2/1}" -EXTNM "A02U01"
call "%GATEWAY%\%APP%" -CP "A1.xml" -EXTRT "{/2/2}" -EXTNM "A02U02"
call "%GATEWAY%\%APP%" -CP "A1.xml" -EXTRT "{/2/3}" -EXTNM "A02U03"
call "%GATEWAY%\%APP%" -CP "A1.xml" -EXTRT "{/2/4}" -EXTNM "A02U04"
@echo off
echo Completed
@echo on
```

This example extracts data as defined by the configured SmartPlant model. Extracted items are defined by the **-EXTRT** parameter and the name of the output file is defined by the **-EXTNM** parameter. Notation like **{/1}** means that the entire first system is going to be extracted, **{/1/1}** means all items of the first member of the first system are extracted, **{/1/2}** means all items of the second member of the first system are extracted, etc. There is no depth limit, so **{/3/2/2/1}** is still valid. As the result of execution of the above example, a set of graphical files will be created:

A01U11.xml – with content of unit 11 of area 1
A01U12.xml – with content of unit 11 of area 1
A02U01.xml – with content of unit 1 of area 2
A02U02.xml – with content of unit 2 of area 2
A02U03.xml – with content of unit 3 of area 2
A02U04.xml – with content of unit 4 of area 2

```
@echo off
echo Running gateway...
set GATEWAY=C:\Program Files\AVEVA\AVEVA NET Gateways\Gateway for SmartPlant 3D
set APP=AVEVA.NET.Gateways.SG3D.exe
cd C:\Users\<username>\Gateway for SmartPlant 3D Projects\A1
@echo on
call "%GATEWAY%\%APP%" -CP "TEST-A001.xml" -EXTRT "{/2/1} {/2/2}" -EXTNM
"A02(U01+U02)"
@echo off
echo Completed
@echo on
```

This example produces output file that contains all graphical items from two systems **{/2/1}** and **{/2/2}**.

Notes:

- If more than one system is going to be exported then the identifiers must be separated with the space character, for example: **"{/2/1} {/2/2} {/2/3}"**.
- Identifier **{root}** can be used to export the entire content of the model, but this is not recommended, especially for big models.
- If the UID of the system to be extracted is known then it may be used instead of the **"{/2/1}"** notation, for example: **"{00033458-0000-0000-6903-030071496404}"**.
- By default space items are not extracted even if selected explicitly by identifiers. To enable the extraction of space items special instruction **{{NS}}** must be passed together with identifiers, for example: **-EXTRT "{{NS}} {/8}"**.

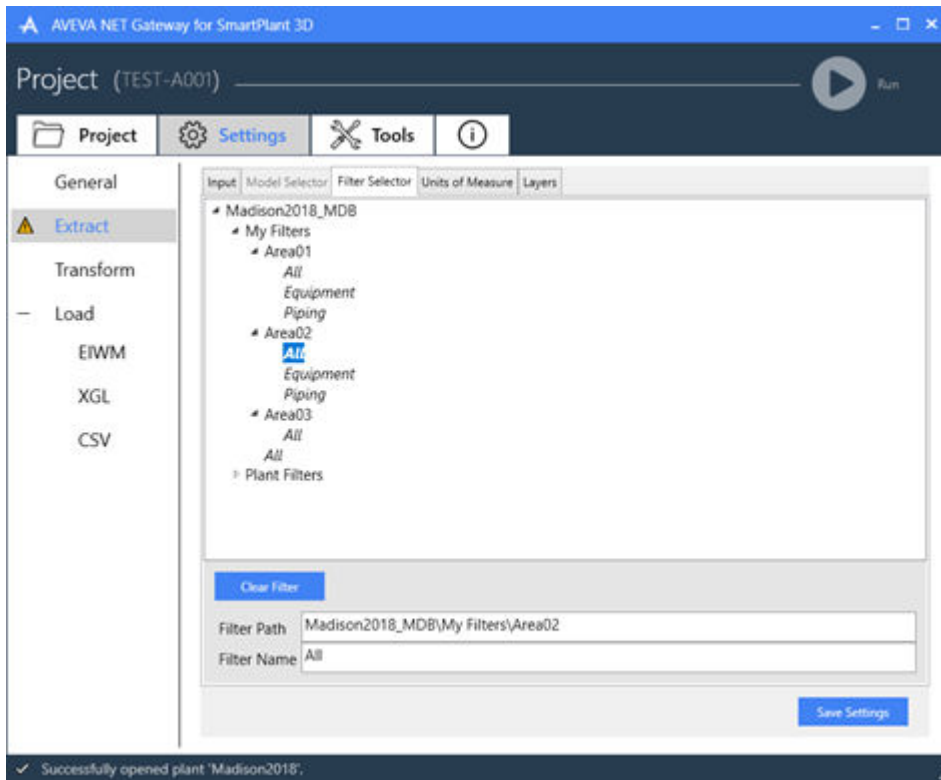
The following script returns all of the items of the {root} object of a plant named "Madison2018", from which you can determine the identifiers in the {/1/1} level.

```
@echo off
echo Getting information...
set GATEWAY=C:\Program Files\AVEVA\AVEVA NET Gateways\Gateway for SmartPlant 3D
set APP=x86\svcS3d.exe
echo Members of {root}
call "%GATEWAY%\%APP%" -Checkout -OpenPlant "Madison2018" -GetMembers "{root}"
@echo on
```

- Creating output on basis of a selected filter defined in the SmartPlant project.

```
@echo off
echo Running gateway...
set GATEWAY=C:\Program Files\AVEVA\AVEVA NET Gateways\Gateway for SmartPlant 3D
set APP=AVEVA.NET.Gateways.SG3D.exe
cd C:\Users\<username>\Gateway for SmartPlant 3D Projects\A1
@echo on
call "%GATEWAY%\%APP%" -CP "A1.xml" -EXTFL "Madison2018_MDB|My Filters|Area01|ALL" -EXTNM "A01/ALL"
call "%GATEWAY%\%APP%" -CP "A1.xml" -EXTFL "Madison2018_MDB|My Filters|Area02|ALL" -EXTNM "A02/ALL"
call "%GATEWAY%\%APP%" -CP "A1.xml" -EXTFL "Madison2018_MDB|My Filters|Area03|ALL" -EXTNM "A03/ALL"
call "%GATEWAY%\%APP%" -CP "A1.xml" -EXTFL "ALL" -EXTNM "ANY/ALL"
@echo off
echo Completed
@echo on
```

This example shows how to use filters. The above four calls are using different filters named in the same way. The first three calls are providing the exact path to the filter that is going to be used in the extraction. The fourth call will run the first found filter named "ALL" so might be none from the above - this method is recommended only for filters where the user is sure they have unique names, otherwise the full path to the filter should be provided. Full path of the filter should be preceded with the name of the model database that the filter will work on. This name can be easily discovered in the UI of the Gateway. In the below screenshot, this name is just a name of the root element from the 'Filter Selection' tab page.



Notes:

- You need to fill all mandatory fields in the configuration file, such as project name and project path.
- If the project configuration file is incorrect, the command line prompt throws an exception and the execution process is stopped.

Example:

Project paths, such as extractor config path, loader config path, transform config path, input path, log path and output path mentioned in the project configuration file must be valid. Otherwise, the execution of the configuration file stops.

Output path scenarios are as follows:

- If you provide an output path in the command line, then the provided path is the final output path. If you do not provide any output path, then the output path defined in the load configuration file is used.
- If the output path provided by you is valid, then a check is conducted to determine whether the output path exists. If the output path does not exist, then a directory is created for the output files.
- If you provide an invalid output path format, then the command line shows an exception message showing the invalid output path.

Exit Codes:

The following command line exit codes are defined for **AVEVA.NET.Gateways.SG3D.exe**.

Exit Code	Definition
0	Success

1	Gateway processing stopped because of general error
3	Gateway processing canceled
7	Processing failed in extractor
9	Processing failed in transformer
11	Processing failed in ZGL loader
13	Processing failed in EIWM loader
18	File processing stopped due to timeout
19	Processing failed in CSV loader
1000	Gateway processing failed

Note: The exit code returned from the Gateway can be captured in the `%errorlevel%` variable when executed in batch mode.

Status Messages from the Gateway Processing

The log file contains all information, warnings, and errors relevant to a project's processing of source data. However, this is often not a convenient format for you to assess whether the processing has proceeded as expected. A better way to monitor the status of a Gateway's processing is to use an additional option for generating a JavaScript Object Notation (**JSON**) report file. This option is controlled by the project setting in the selected section of the Gateway (this option is set to **false** by default).

```
<GenerateCategorizedLog>true</GenerateCategorizedLog>
```

When set to **true**:

- A JSON report file is written in the same location as the log file. The level of detail in the JSON report file depends on the project settings.
- If the Gateway process fails, then an additional JSON error report file containing details of the process and the error message is written in the same location, named "`<InputFileName>_Error.json`".

The JSON report contains headers with general information as listed below:

Field	Value Example	Description
"Product"	"AVEVA Components Reporting Utility"	Reporter program name.
"ReportingVersion"	"X.X.X.X"	Reporter program version.
"ReportedProduct"	"AVEVA Gateway for SmartPlant 3D"	Gateway name.
"ReportedProductVersion"	"Y.Y.Y.Y"	Gateway version number.
"StartDateTime"	"06-02-2020 19 06 43"	Processing start time.
"FinishDateTime"	"06-02-2020 19 06 44"	Processing end time.
"ElapsedTime"	"00 00 00 (0.97 seconds)"	Time of the processing.

Field	Value Example	Description
"SourceName"	"Sample.dwg"	Name of source data.
"SuccessfulObjectsNumber"	1255	Number of successfully processed objects.
"ProcessingResult"	"Success"	Overall result of the processing: Success/Warnings/Errors/Failure.
"Configurations"	...	List of configuration files.
"OutputData"	...	List of output files.
"Options"	...	Additional processing options.
"Reports"	...	List of messages (see the following note).

Notes: The [ProcessingResult](#) field is set according to the following rules:

- "Success", if no warnings or errors are detected.
- "Warnings", if at least one warning is detected but no errors.
- "Errors", if at least one minor error is detected but none that have halted the processing, for example, one or more objects might have been skipped but the entire source was read and processed.
- "Failure", if a major error has caused the processing to be terminated early.

The JSON report then contains one or more status messages:

Field	Value	Description
"ID"	2	Number of the message.
"DateTime"	"06-02-2020 19 06 43"	Timestamp of the message.
"SubComponent"	"AutoCAD 2D"	Source Gateway's component of the generated message.
"Severity"	"Warning"	Severity of the message: Debug/Information/Warning/Error/ErrorProcessingFailed/ErrorCritical.
"Code"	11644681	Unique ID of the message.
"CodeName"	"WAR_EXT_AUT_B1AF09"	Unique name of the message.
"Category"	"Extract"	Category of the message (see the following table.).
"Message"	"Could not write document.csv".	Text message.
"Remediation"	"Check that the output CSV file isn't open in another application."	Optional remediation.

Categories should be used to scan the report for types of messages that might require further action:

Message Category	Message Description
Access	Issues connected with accessing provided location, file or database.
Break	Break of processing called by user of timeout.
Debug	Low level technical information describing processing details. Only for severity Debug.
Environment	Issue influenced by system environment like missing memory, read only file.
Extract	Problem with extracting some source data like wrong item or unrecognized API behavior.
Information	Typical informational official message not connected with bug nor low level processing. Only for severity Information.
Initialize	Issues connected with initialization of the processing.
Internal	Issues connected with program's code which is not yet recognized and categorized but protected against program crash.
Invalid	Issues with provided data: input file, configuration, connection string.
MissingData	Expected data required for processing is missing in expected location.
Overload	Issue with the amount of data or crossing an allowed range.
Redundancy	Multiple and not necessary items of any type.
Reformat	Automatic adjusting data to expected format during conversion due to output format requirements.
Respond	No response or an unexpected response for the query sent by the program.
Support	Limited or no support of the type of source data or configuration item.
Transform	Issue/information about one of the transform mapping feature used on processed data.
Unknown	Can be used for unhandled exceptions like additional protection for native errors thrown by 3rd party libraries.

JSON Report Limitations:

The report is generated during the processing of each input file. Therefore, it does not contain messages connected with issues which arise before the processing of the file has started, for example, use of incorrectly formatted configuration files. In such cases, the message is delivered as a pop-up when in GUI mode or command line messages in case of batch mode. This information is also written to the [Summary.txt](#) file in the log folder.

Chapter 4 Appendix A: Mapping Configuration

Mapping configuration files are generally defined in XML format. You can edit these configuration files in an XML Editor, Notepad or any Text Editor. The high-level structure of a mapping configuration file is as follows:

```
<MappingTemplate>
  <DataSourceLookups>
    <CsvLookupDataSource>...</CsvLookupDataSource>
    <MSAccessLookupDataSource>...</MSAccessLookupDataSource>
    <MSExcelLookupDataSource>...</MSExcelLookupDataSource>
    <MSSqlLookupDataSource>...</MSSqlLookupDataSource>
    <OracleLookupDataSource>...</OracleLookupDataSource>
  </DataSourceLookups>
  <ManifestAttributes>
    < Attribute>...</Attribute>
    < Attribute>...</Attribute>
  </ManifestAttributes>
  <Datasets>
    <Dataset>...</Dataset>
    <Dataset>...</Dataset>
  </Datasets>

  <TemplateLookups>
    <Template>...</Template>
    <Template>...</Template>
  </TemplateLookups>
  <ObjectMappings regexTimeoutSecs="10">
    <Object>...</Object>
    <Object>...</Object>
    <Object>...</Object>
  </ObjectMappings>
</MappingTemplate>
```

Each configuration setting is detailed in the following sections.

LookupDataSource

Any **Value** (except for the name of an attribute) may be specified as a **lookup** from an external data source. This data source can be one of the following:

- A delimited text file for example, a [.CSV](#) file
- A Microsoft Excel spreadsheet file
- A Microsoft Access database file
- A Microsoft SQL Server database
- An Oracle database

You must define the data sources to be used in lookups in the mapping configuration file. The following is an example of the definition of a Microsoft Excel data source:

```
<MSExcelLookupDataSource
id="Excel Map"
```

```
file="C:\Mapping\Data Value Lookup\Mapping.xls"
table="Sheet1"
sourceColumn="[Source]"
targetColumn="[Destination]" />
```

You can specify any number of data sources in a mapping configuration file. To make use of a lookup, you must reference the Id of the data source within a mapping entry:

```
<Attribute name="Name" >
<Value value="[Name]" >
<Lookup id="Excel Map" >
<FailAction action="EmptyValue" />
</Lookup>
</Value>
</Attribute>
```

In the above example, an Excel spreadsheet is used to store a list of lookup values. The value of the [Name](#) attribute from the source system is passed to the lookup. This value is compared with the contents of the Source column in the Sheet1 of the Excel file and if a matching value is found, the contents of the Destination column is returned and used as the attribute value in the output.

Sheet1 is constructed as follows:

1	Source	Destination
2	keyvalue1	newValue1
3	keyvalue2	newValue2
4	keyvalue3	newValue3
5	keyvalue4	newValue4
6

If no matching values are found in the data source, you can use the values specified in the [FailAction](#) element:

- **FixedValue** – The fall-back value to use if specified, as the [Value](#) attribute of [FailAction](#) element.
- **EmptyValue** – Used for the output if no matching value is found.
- **DiscardElement** - Used to remove the mapped attribute or association on which the lookup is applied.
- **DiscardObject** - Used to remove the engineering object completely from object model on which the lookup is applied.

Notes:

- **DiscardElement** cannot be applied on the Dataset attributes defined in configuration, such as the dataset's ClassID, for example,

```
<Datasets>
  <Dataset id="Dataset1" >
    ...
    <ClassID value="DATASETClass1" />
  </Dataset>
</Datasets>
```

- **DiscardObject** cannot be applied to objects that have been assigned as Datasets, therefore, Lookup checks on input objects intended to become datasets should be done prior to converting them to datasets.
- Lookups may return an attribute name (in square brackets) and have this resolved to the attribute value in the output.

The following data sources are supported:

```
<CsvLookupDataSource
  id="CSVLookup"
  file="C:\Mapping\Data Value Lookup\Mapping.csv"
  separator=","
  provider="Microsoft Access Text Driver (*.txt, *.csv)"
  sourceColumn="Source"
  targetColumn="Destination" />
<MSAccessLookupDataSource
  id="MSAccessLookup"
  file="C:\Mapping\Data Value Lookup\Mapping1.accdb"
  query="SELECT [Source], [Destination] FROM [Table1]"
  provider="Microsoft Access Driver (*.mdb, *.accdb)"
  sourceColumn="Source"
  targetColumn="Destination" />
<MSExcelLookupDataSource
  id="ExcelLookup"
  file="C:\Mapping\Data Value Lookup\Mapping1.xls"
  provider="Microsoft Excel Driver (*.xls, *.xlsx, *.xlsm, *.xlsb)"
  extendedProperties="Excel 12.0; HDR=YES"
  query="SELECT [Source], [Destination] FROM [$Sheet2]"
  sourceColumn="Source"
  targetColumn="Destination" />
<MSSqlLookupDataSource
  id="MSSqlLookup"
  query="SELECT [Key], [Value] FROM [Table1]"
  connectionString="Driver={SQL Server}; Server=serverxxx; Database=Databasexxx;
  UID=userxxx; PWD=pwdxxx;"
  sourceColumn="Key"
  targetColumn="Value" />
or
<MSSqlLookupDataSource
  id="MSSqlLookup"
  connectionString="WZz2JoEigxXpWGjsUwdTAXjSG7N8rI61d+aHz503TrfeCNNcLPpmOg=="
  connectionStringEncrypted="true"
  query="SELECT [Key], [Value] FROM [Table1]"
  sourceColumn="Key"
  targetColumn="Value" />
<OracleLookupDataSource
  id="OracleLookup"
  connectionString="WZz2JoEigxXpWGjsUwdTAXjSG7N8rI61d+aHz503TrfeCNNcLPpmOg=="
  connectionStringEncrypted="true"
  query="SELECT [Key], [Value] FROM [Table1]"
  sourceColumn="Key"
  targetColumn="Value" />
or
<OracleLookupDataSource
```



```
id="OracleLookup"
connectionStringEncrypted="false"
connectionString="Driver={Oracle in OraClient12Home1}; DBQ=serverAddressxxx;
UID=userxxx; PWD=passwordxxx;"
query="SELECT Key, Value FROM Table1"
sourceColumn="Key"
targetColumn="Value" />
```

Notes:

- You can store the encrypted connection string details for [MSSqlLookupDataSource](#) and [OracleLookupDataSource](#) as the connecting string may contain sensitive information such as [user name](#) and [password](#). The encrypted connection string can be created using the Encrypt tool present in the **Tools** tab as described in [Appendix D: Encrypt Utility](#) (). If you encrypt the [connectionString](#) value, the [connectionStringEncrypted](#) attribute value must be set to true.

- The [extendedProperties](#) attribute is optional in the below-mentioned Lookup data sources. If not specified, these defaults to the following values:

Lookup Data Sources	Value
MSExcelLookupDataSource	Excel 12.0;HDR=YES

- Only one of the attributes either **table** or **query** can be used interchangeably in Lookup data sources mentioned below. The value of attribute [sourceColumn](#) and [targetColumn](#) has to be a column name from the mentioned table or query value.

Lookup Data Sources	Value
CsvLookupDataSource	Not Applicable
MSExcelLookupDataSource	Yes
MSAccessLookupDataSource	Yes
MSSqlLookupDataSource	Yes
OracleLookupDataSource	Yes

- For security reasons, it is advised that the permission of the DB (database) user account should be restricted to read-only.
- In order to ensure the security of data which is in transit between the Gateway and an Oracle or an SQL Server database, you can configure an SSL-encrypted connection between the two.
- As the encrypted password is associated with the local machine where the Gateway is running, other machines cannot use the configuration file directly. You will have to encrypt the password before using the configuration created in another machine. You can obtain an encrypted password using the encryption utility.

Deriving Object ID and Class ID from Lookup

In some cases, the [filename](#) does not correspond to the document name ([ObjectID](#)). For example, the filename may contain extra text to indicate its status or version. In this case, the correct document name can be assigned to the Document Object's ID via a lookup to an external source, such as a document register.

The following example shows how to derive the ObjectID and the ClassID from an Excel file.

Example of an Excel file content:

Source	Destination
Sample3D	NewObjectName
3D Model	NewObjectClass

The following example modifies the ObjectID and ClassID of the Document Object (Manifest) by matching the input data to the Source column and using the Destination column from the lookup file.

Lookup Declaration:

```
<DataSourceLookups>
  <MSExcelLookupDataSource
    id="ExcelLookup"
    file="C:\Temp\Example.xlsx"
    table="CSVLookup"
    sourceColumn="Source"
    targetColumn="Destination" />
</DataSourceLookups>
```

Object ID and Class ID Transformations:

```
<Object>
  <Conditions>
    <Attribute name="#TYPE#" pattern="^manifest$" />
  </Conditions>
  <ObjectID value="[ObjectID]">
    <Lookup id="ExcelLookup" >
    </Lookup>
  </ObjectID>
  <ClassID value="[ClassID]">
    <Lookup id="ExcelLookup" >
    </Lookup>
  </ClassID>
</Object>
```

Default Value Mapping in Attribute

While creating new attributes, you can provide a default value in the base mapping if no attribute value is found in the data source. If no value is found in the data source for a mapped property/characteristic, then that will be excluded from the output file unless you define a default for it.

```
<Object>
  .....
  <Attributes>
    <Attribute>
      <Name value="Pressure" />
```

```

        <Value value="[max_pressure]" default="NA" />
    </Attribute>
</Attributes>
</Object>

```

In the above example, an attribute Pressure is created in the object. The value for this attribute is resolved using the [max_pressure] attribute belonging to the same object, for example, most probably extracted from the source data. If the referenced attribute (max_pressure) is not present for this object, then the default value 'NA' will be used for the Pressure attribute value.

The result of this mapping is those objects with the [max_pressure] attribute present in the source will have the following characteristic in EIWM:

```

<Characteristic>
  <Name>Pressure</Name>
  <Value>30</Value>
</Characteristic>

```

and for the objects that do not have the [max_pressure] attribute will have the following characteristic in EIWM:

```

<Characteristic>
  <Name>Pressure</Name>
  <Value>NA</Value>
</Characteristic>

```

TemplateLookups

Each object present in the extracted data needs an attribute TemplateID for it to convert into its equivalent EIWM object. TemplateLookups associates a TemplateID to individual objects present in the extracted data.

```

<TemplateLookups>
  <Template id ="default" >
    <TemplateID value="[object_name]" >
      <Transforms>
        <!-- Append 'Template'.-->
        <InsertAfter pattern="^.*$" value=" Template" />
      </Transforms>
    </TemplateID>
  </Template>
  <Template id ="template1">
    <TemplateID value="[object_name]" >
    </TemplateID>
  </Template>
  <Template id =" template2">
    <TemplateID value="123" />
  </Template>
</TemplateLookups>
<ObjectMappings regExTimeoutSecs="10">
  <Object>
    <Conditions>
      <Attribute name="ClassName" pattern="Door" />
    </Conditions>
    <TemplateID id =" template1" />
    <ObjectID value="[GlobalId]">
    </ObjectID>
  </Object>

```

```
</Object>
</ObjectMappings>
```

Note: If an object is not associated with any template Lookup ID, a default [TemplateID](#) is generated with the EIWM file name.

Transforms

Any value in the mapping (except for the name of an attribute) permits transformation of the source value.

```
<Attribute>
<Conditions>
<Attribute name="ClassName" pattern="PUMP" />
<Attribute name="Name" pattern="^.*$" />
</Conditions>
<Name value="Maximum Pressure" />
<Value value="[max_pressure]">
<Transforms>
<LowerCase />
<Remove pattern="^[a-z]{3}" />
<Replace pattern="\d{6}" value="yyyyyy" />
<InsertAfter pattern="yyyyyy" value="Before " />
<InsertBefore pattern="yyyyyy" value=" After" />
<UpperCase />
</Transforms>
</Value>
<Units value="psi" />
</Attribute>
```

The following transformations of the source value are available:

Transformation	Description
Remove	Permits parts of the source value to be removed. The pattern specifies a regular expression used to identify part of the value to remove.
Replace	Permits parts of the source value to be replaced. The pattern specifies a regular expression used to identify part of the value to replace, and the value specifies the replacement text.
Keep	If matched to the pattern, Keep will replace the source value with the part of the text that matches the pattern. The pattern specifies a regular expression. If the pattern is not matched, then the source value is set to blank and a warning is logged.
InsertAfter	Permits the insertion of text after a position in the value specified by a Pattern and the value specifies the text to insert.

InsertBefore	Permits the insertion of text before a position in the value specified by a Pattern, and the value specifies the text to insert.
UpperCase	Converts the value to upper case.
LowerCase	Converts the value to lower case.
Compose	Reconstructs a string. The string value can be shortened, divided with regular expression and the parts concatenated in any order using a constructor mask. Parts can be used multiple times.

You can include any number and type of transformations for a given value. The values specified for the Replace and Insert transforms may reference other attributes from the source system, for example, [attribute_name]. If you specify multiple transformations, the transforms are processed in the order listed in the configuration. Transforms may be combined with lookups. In this case, the transform is applied first and the resulting value is passed to the lookup as a key. When using transformations, it is recommended that you specify pattern to ensure the attribute value is in the format expected by the transformation sequence.

Example of Compose Delimiter:

Input: "Tool designation: P-01A 4000"

<Transforms>

<Keep pattern="[A-Z]-\d{2}[A-Z]" />

<Compose delimiters="-" constructor="type: {1}, series: {2}" />

</Transforms>

Output: "type: P, series: 01A"

Description:

- [Keep](#) parameter shortens input into "P-01A".
- [delimiters](#) "-" then divides it into parts: "P" and "01A".
- [constructor](#) parameter inserts the parts {1} and {2} with respect to the masks: "type: {1}, series: {2}".

Regular Expression Evaluation Timeout

Evaluation of complex regular expressions can sometimes take a significant amount of time. You may optionally specify a timeout value (in seconds), for evaluation of regular expressions used in mapping entries specified in the file.

Note: If not specified, the default value for this timeout is 60 seconds.

```
<ObjectMappings regexTimeoutSecs="10">
```

If the regular expression is not evaluated within the time specified, an error is raised.

Timestamp References

You can use a special placeholder value to insert the current date and time into a value.

This placeholder is specified as `[DateAndTime]`.

```
<Template>
  <TemplateID value="[DateAndTime] Template" />
</Template>
```

The Template Mapping entry in the above example generates a template ID from the current date and time concatenated with the text Template.

The format of the date and time inserted into the value is: `dd-MM-yyyy HH:mm:ss`.

AutoNumber References

`AutoNumber` is a special placeholder used to generate an automatically incremented numeric counter. You can use it to create uniquely identified values while resolving the duplicate mapped values.

This placeholder is either specified as `[AutoNumber]` where the sequence start value is defaulted to "1" or `[AutoNumber:<sequence start value>]`.

Notes:

- The value of sequence start value should be an integer greater than or equal to "0". If you enter a negative value, then the sequence start value is defaulted to "1".
- Re-processing another version of the input data may result in different numbers being assigned as the order of the objects may change.

Example:

```
<Object>
  <Conditions>
    <Attribute name="Name" pattern="Door"/>
  </Conditions>
  <ObjectID value="[Name]_[AutoNumber]">
  </ObjectID>
</Object>
```

Above example indicates that if you have multiple objects in your source system having `Name` attribute with their value matching "Door", then the value of the `Name` attribute along with the special `[AutoNumber]` placeholder can be used in deriving the unique `ObjectID` values such as `Door_1`, `Door_2`, `Door_3` and so on.

```
<Object>
  <Conditions>
    <Attribute name="Name" pattern="Door"/>
  </Conditions>
  <ObjectID value="[Name]_[AutoNumber:11]">
  </ObjectID>
</Object>
```

Above example indicates that if you have multiple objects in your source system having `Name` attribute with their value matching "Door", then the value of the `Name` attribute along with the special `[AutoNumber]` placeholder can be used in deriving the unique `ObjectID` values such as `Door_11`, `Door_12`, `Door_13` and so on.

InternalId References

For internal tracking purposes, a GUID is generated within the Gateway's processing logic whenever an object is extracted. This may be re-used if necessary, for example, if you are unable to use any attribute or combination of attributes as a unique [ObjectID](#).

You can use a special place holder value to insert the internally generated GUID into a value. This place holder is specified as [\[InternalId\]](#).

```
<Object>
  <ObjectID value="[InternalId]">
</ObjectID>
</Object>
```

The Mapping entry in the above example generates an [ObjectID](#) from the internally generated GUID.

Note: Using [InternalId](#) is not recommended because the value is entirely independent of the input data and the value of an [InternalId](#) is not repeated. Hence re-processing the same input data results in different values of [InternalId](#) per extracted object.

Manifest References

The following table lists the supported manifest attribute placeholders:

File Object Attribute Placeholder	Value
#TYPE#	Fixed value " manifest ".
#MANIFEST#	Fixed value " SG3D ".
#CURRENT_USER#	User name of the person who is currently logged on to the Windows operating system during execution.
#DATE_TIME#	File Object creation timestamp.
#GRAPHICS_FORMAT#	Fixed value " tessellated ".
#MODEL_NAME#	Populated with the name assigned to Extraction name in Filesystem section.
#DEFINED_SEGMENTS#	FALSE <ul style="list-style-type: none"> • BaseMapping extension is executed before "AVEVANETPresentationMapping" extension. • No Presentation Mapping. • Lack of 'segments' section in Presentation Mapping. • 'Segments' section exists without any 'Segment' node. TRUE <ul style="list-style-type: none"> • Segment's node exists (one or more).

File Object Attribute Placeholder	Value
#KEEP_FOLDER_TREE#	TRUE, if input folder hierarchy is preserved in output generation.
#TARGET_LOCATION#	Output folder path.
#SEGMENT_NAME#	Segment name of the graphics drawing.
#UNITS#	Source drawing units.

Note: Do not change the following attributes:

- #INPUT_FOLDER#
- #TYPE#
- #MANIFEST#
- #CURRENT_USER#
- #DATE_TIME#
- #GRAPHICS_FORMAT#
- #INPUT_LOCATION#
- #INPUT_FOLDER#
- #KEEP_FOLDER_TREE#

The following mapping entry indicates how to use any of the **Manifest** attribute values. For example, to add a new attribute named **IDENTIFIER** with the value of the manifest **"#MODEL_NAME#"** attribute, into all the objects that match the condition where the **Tag** attribute is present. As attribute **"#MODEL_NAME#"** belongs to the associated document object you must use the prefix **"associated:"** to resolve the value from its associated objects.

```
<Object>
  <IncludeAssociatedObjectAttributes>
    <AssociationType pattern="^is referenced in$" />
    <Conditions>
      <Attribute name = "#TYPE#" pattern = "^manifest$" />
    </Conditions>
  </IncludeAssociatedObjectAttributes>
  <ObjectID value="[Id]" />
  <Attributes>
    <Attribute name="Tag" pattern="^[A-Za-z]+$" >
      <Name value="IDENTIFIER" />
      <Value value="[associated:#MODEL_NAME#] [Tag]" />
    </Attribute>
  </Attributes>
</Object>
```

User-Defined Manifest Attributes

You can define your own list of custom attributes, which can be used during transformation as other pre-defined manifest attributes. The following is the way to add these user-defined manifest attributes to the configuration file.

```
<ManifestAttributes>
  <Attribute name="USER_ATTRIBUTE1" value="VALUE1" />
  <Attribute name="#USER_ATTRIBUTE2#" value="VALUE2" />
</ManifestAttributes>
```

These user-defined manifest attributes can then be referenced during transformation to resolve other values.

```
<Object>
  <IncludeAssociatedObjectAttributes>
    <AssociationType pattern="^is referenced in$" />
    <Conditions>
      <Attribute name = "#TYPE#" pattern = "^manifest$" />
    </Conditions>
  </IncludeAssociatedObjectAttributes>
</Attributes>
  <Attribute name="Tag" pattern="^[A-Za-z]+$" >
    <Name value="IDENTIFIER" />
    <Value value="[associated:USER_ATTRIBUTE1] [Tag]" />
  </Attribute>
  <Attribute>
    <Name value="User Attribute" />
    <Value value="[associated:#USER_ATTRIBUTE2#]" />
  </Attribute>
</Attributes>
</Object>
```

The above mapping entry indicates that for all the objects matching the condition, if the **Tag** attribute is present in this object, then a new attribute named **IDENTIFIER** is added. The attribute value is taken from the value of the user-defined manifest attribute "USER_ATTRIBUTE1", appended with the **Tag** attribute value. It also creates a new attribute named "User Attribute" whose value is taken from the value of the user-defined manifest attribute "#USER_ATTRIBUTE2#".

Note: Manifest attributes can also be defined on the command line. For more information, see [Running the Gateway from the Command Line](#) ().

The document object's class is read from an attribute named **ClassID**.

By default, its value is 3D Model derived from the **#MODEL_NAME#** value, but it can be changed in base mapping configuration by identifying the document object in a Conditions statement and updating the "ClassID" attribute value:

```
<Object>
  <Conditions>
    <Attribute name="#type#" pattern="manifest" />
  </Conditions>
  <Attributes>
    <Attribute name="ClassID">
      <Value value="Test class"/>
    </Attribute>
  </Attributes>
</Object>
```

Note: You must not remove either the Manifest object or any of its attributes as these attributes such as **#MODEL_NAME#**, **#TARGET_LOCATION#** are needed for loader functions. You can control whether the Manifest

object is included in the EIWM as the Document object via the [createDocumentObject](#) option. For more information, see [Load Settings](#) ().

The following example shows how to remove the attributes from other objects without affecting the [Manifest](#) object.

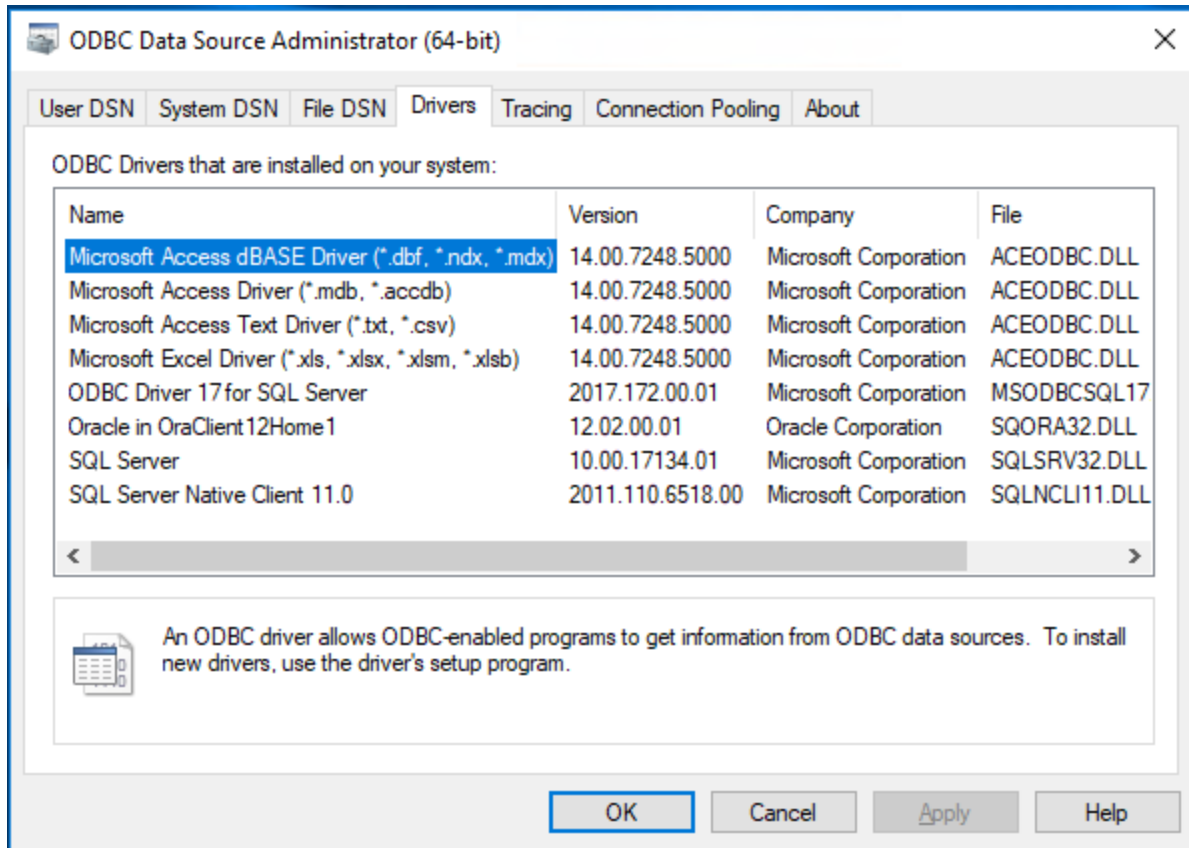
```
<ObjectMappings regExTimeoutSecs="10">
  <Object>
    <Conditions>
      <Attribute name="#TYPE#" pattern="manifest$" />
    </Conditions>
    <TemplateID id ="default" />
  </Object>
  <Object>
    <TemplateID id ="default" />
    <ObjectID value="[TAG]"/>
  </Object>
  <ClassID value ="[Classification]" />
  <Attributes>
    <Attribute name="*" >
      <Remove />
    </Attribute>
  </Attributes>
</Object>
</ObjectMappings>
```

Migrate Oledb to ODBC

Older versions of the Gateway used OleDb drivers to connect to lookup sources, but as OleDb is no longer supported, ODBC drivers should be used instead.

Installing the Microsoft Access Database Engine (ADE) installs the required ODBC drivers for Excel, Access, CSV (Delimited Text), SQL Server and Oracle.

The provider names for each type of ODBC driver can be obtained from the Windows ODBC Data Source Administrator, for example, in Windows 10:



Connection String Changes

As the [LookupDataSource](#) configurations are specific to a project it is not possible to provide automatic upgrades from configurations that specified OleDb connections to ODBC connections, so they must be updated manually.

Note: There is no change in the Query syntax when migrating from OleDb to ODBC.

The following are the connection string changes:

Excel:

Defining an Excel lookup using an OleDb driver would be similar to:

```
<MSExcelLookupDataSource id="ExcelLookup1" file="<FilePath>"
provider="Microsoft.ACE.OLEDB.12.0"
extendedProperties="Excel 12.0; HDR=YES" query="SELECT [Source], [Destination] FROM
[Sheet1$]"
```

This should be updated to support an ODBC driver:

```
<MSExcelLookupDataSource id="ExcelLookup1" file="<FilePath>"
provider="Microsoft Excel Driver (*.xls, *.xlsx, *.xlsm, *.xlsb)"
extendedProperties="Excel 12.0; HDR=YES" query="SELECT [Source], [Destination] FROM
[Sheet1$]"
```

Access:

Defining an Access lookup using an OleDb driver would be similar to:

```
<MSAccessLookupDataSource id="MSAccessLookup" file="<FilePath>"
query="SELECT [Source], [Destination] FROM [ObjectID]" sourceColumn="Source"
targetColumn="Destination" />
```

This should be updated to support an ODBC driver:

```
<MSAccessLookupDataSource id="MSAccessLookup" file="<FilePath>"
provider="Microsoft Access Driver (*.mdb, *.accdb)"
query="SELECT [Source], [Destination] FROM [ObjectID]" sourceColumn="Source"
targetColumn="Destination" />
```

CSV:

Defining a CSV lookup using an OleDb driver would be similar to:

```
<CsvLookupDataSource id="CSVLookup" file="<FilePath>"
separator="," sourceColumn="Source" targetColumn="Destination" />
```

This should be updated to support an ODBC driver:

```
<CsvLookupDataSource id="CSVLookup" file="<FilePath>"
provider="Microsoft Access Text Driver (*.txt, *.csv)" separator="," sourceColumn="Source"
targetColumn="Destination" />
```

Notes: "Provider" in Excel, Access and CSV is optional. If Provider is not provided, default values would be:

- Excel - Microsoft Excel Driver (*.xls, *.xlsx, *.xlsm, *.xlsb)
- Access - Microsoft Access Driver (*.mdb, *.accdb)
- CSV - Microsoft Access Text Driver (*.txt, *.csv)

SQL Server:

Defining an SQL server lookup using an OleDb driver would be similar to:

```
connectionString="Provider=SQLOLEDB; Server=serverxxx; Database=Databasexxx; User
ID=userxxx; Password=pwdxxx;"
```

This should be updated to support an ODBC driver:

```
connectionString="Driver={SQL Server}; Server=serverxxx; Database=Databasexxx; UID=userxxx;
PWD=pwdxxx;"
```

'Defining an Oracle lookup using an OleDb driver would be similar to:

Oracle:

Defining an SQL server lookup using an OleDb driver would be similar to:

```
<OracleLookupDataSource id="OracleLookup"
connectionString="Provider=OraOLEDB.Oracle;
Data Source=serverAddressxxx;
User ID=userxxx; password=passwordxxx;"
query="select Source, Destination from DocumentLookup"
sourceColumn="Source" targetColumn="Destination" />
```

This should be updated to support an ODBC driver:

```
<OracleLookupDataSource id="OracleLookup" connectionString="Driver={Oracle in
OraClient12Home1};
```

```
DBQ=serverAddressxxx;
UID=userxxx; PWD=passwordxxx;"
query="select Source, Destination from DocumentLookup"
sourceColumn="Source" targetColumn="Destination" />
```

Note: Alternative ODBC Drivers are available, but the connectionString details will be different. Examples of these are:

- <https://go.microsoft.com/fwlink/?linkid=2168524> for SQL Server
- <https://www.oracle.com/in/database/technologies/releasenote-odbc-ic.html> for Oracle

Base Mapping Types

The following sections describe general details about the types of mapping used in the Gateway.

Object Mapping

A generic object consists of a list of attributes that defines the characteristics of that object. It also contains a list of associations to other generic objects.

The **Object Mapping** section is classified as follows:

- **Transform mapping** settings to modify an existing object/remove it.
- **Load mapping** settings to make the object ready for AIM load.

The following is an example of Object Mapping:

```
<Object>
...
<Conditions>
  <Attribute name="Class" pattern="EQUIPMENT" />
</Conditions>
<TemplateID id="default" />
<ObjectID value="ID_[object_name]" />
<ObjectName value="[object_name]" />
<ClassID value="[Class]" />
<ContextID value="[object_context]"/>
<Revision value="[Versionlabel]"/>
<IncidentalClassifications>
  <IncidentalClassification value="void" />
</IncidentalClassifications>
<Attributes>
  <Attribute name="Identifier" >
    <Value value="[Tag]" />
  </Attribute>
  <Attribute>
    <Conditions>
      <Attribute name="Class" pattern="EQUIPMENT" />
      <Attribute name="Subtype" pattern="PUMP" />
    </Conditions>
    <Name value="Maximum Pressure" />
    <Value value="[max_pressure]"/>
    <Units value="psi" />
  </Attribute>
```

```

</Attributes>
<Associations>
  <Association attribute="Nozzle_ID" attributePattern="^.*$" >
    <Type value="Has Nozzle" />
    <TargetID value="[Nozzle_ID]" />
    <TargetClassID value="NOZZLE" />
    <ContextID value="[object_context]" />
  </Association>
</Associations>
...
</Object>

```

Transform Mapping Settings

Transform mapping settings contain the following mappings that can be applied to a generic object.

Object Remove Mapping:

```

<Object>
  <Conditions>
    <Attribute name="object_class" pattern="Door" />
  </Conditions>
  <Remove />
</Object>

```

This mapping entry indicates that for all the objects matching the condition, remove the objects so they are no longer considered for the output files.

Note: The mandatory attributes needed for objects in EIWM such as the [ObjectID](#) identifier of an object (aka tag) and its [ClassID](#) are mapped just like any other attribute and refer to the Attribute mapping section.

To set custom output file names, add the following attributes to the [Manifest](#) object:

[#TARGET_NAME_EIWM#](#): Sets output name for EIWM.

[#TARGET_NAME_XGL#](#): Sets output name for XGL.

[#TARGET_NAME_CSV#](#): Sets output name for CSV.

Example Base Mapping:

```

<Object>
  <Conditions>
    <Attribute name="#TYPE#" pattern="^manifest$" />
  </Conditions>
  <Attributes>
    <Attribute>
      <Name value="#TARGET_NAME_EIWM#" />
      <Value value="custom eiwm name" />
    </Attribute>
  </Attributes>
  <Attribute>
    <Name value="#TARGET_NAME_XGL#" />
    <Value value="custom xgl name" />
  </Attribute>
  <Attribute>
    <Name value="#TARGET_NAME_CSV#" />
    <Value value="custom csv name" />
  </Attribute>

```

```
</Attributes>
</Object>
```

Notes:

- The Output name for **EIWM** is overridden by setting the **File Name** under **Custom Folder Structure** if set.
- The output name for **XGL** is overridden by **Presentation Mapping - Segmentation** if set.

Attribute Mapping

Attribute Mapping determines how attributes from the source system are transformed to match the requirement of the target system. For each attribute from the source system, each mapping entry is examined to determine whether the attribute matches.

Note: The attribute qualifier is required in attribute mapping entries. You may specify a [pattern](#) property to match only attributes with a value that conforms to the regular expression.

Attribute mapping entries permit the following Values to be specified:

- Conditions to match
- Name of the attribute
- Value of the attribute
- Units of the attribute

A sample Attribute mapping example is as follows:

```
<Object>
...
  <Attributes>
<Attribute>
<Name value="XYZ" />
<Value value="ABC" />
</Attribute>
</Attributes>
...
</Object>
```

Attribute Mapping entries can be broadly classified into the following:

Create:

The simplest form of attribute mapping entry to create a new attribute is shown below:

```
<Object>
...
  <Attributes>
    <Attribute>
<Name value="Identifier" />
<Value value="[Tag]" />
    </Attribute>
  </Attributes>
...
</Object>
```

This mapping entry adds a new attribute named [Identifier](#) to the source system. The attribute value is taken directly from the value of the [Tag](#) attribute from the source system.

The attribute mapping entry to create a new attribute matching a simple condition, is shown below:

```
<Object>
...
  <Attributes>
    <Attribute name="Tag" pattern="^[A-Za-z]+$" >
      <Name value="Identifier" />
      <Value value="[Tag]" />
    </Attribute>
  </Attributes>
...
</Object>
```

This mapping entry indicates that for all the objects matching the condition, if the **Tag** attribute in the source system is present, then a new attribute named **Identifier** is added. The attribute value is taken directly from the value of the Tag attribute from the source system.

The attribute mapping entry to create a new attribute taking multiple attributes with their corresponding patterns to match specific format is shown below:

```
<Object>
...
  <Attributes>
    <Attribute>
      <Conditions>
        <Attribute name="ClassName" pattern="Window" />
        <Attribute name="SubType" pattern="Solid" />
      </Conditions>
      <Name value="Type" />
      <Value value="[SubType] [ClassName]">
        <Transforms>
          <Replace pattern="\d{2}" value="yyy" />
        </Transforms>
      </Value>
      <Units value="psi" />
    </Attribute>
  </Attributes>
...
</Object>
```

You may use a lookup to obtain the attribute value. You may use transforms to modify the attribute value from the source system.

Transforms may be combined with a lookup, in which case the transform is applied to the source value and the resulting value is passed to lookup:

```
<Object>
...
  <Attributes>
    <Attribute name="Tag" pattern="^[A-Za-z]+$" >
      <Name value="Identifier" />
      <Value value="[Tag]">
        <Transforms>
          <Replace pattern="\d{2}" value="yyy" />
        </Transforms>
      <Lookup Id="Attribute Value Map" >
        <FailAction action = "FixedValue" value="[Name]" />
      </Lookup>
    </Attribute>
  </Attributes>
...
</Object>
```



```

    </Value>
  </Attribute>
</Attributes>
...
</Object>

```

Update:

The simplest form of attribute mapping entry is to update an existing attribute:

```

<Object>
  ...
  <Attributes>
<Attribute name="Tag" pattern="^[A-Za-z]+$" >
<Value value="abcdef123" />
<!-- One can do transform or lookup here on the value if required -->
</Attribute>
</Attributes>
...
</Object>

```

This mapping entry indicates that for all the objects matching the condition, the Tag attribute in the source system, if present and matching the pattern then, the attribute value of Tag is fixed to abcdef123.

```

<Object>
  ...
<Attributes>
<Attribute name="Tag">
<Value value="ID#[Tag]" />
<!-- One can do transform or lookup here on the value if required -->
</Attribute>
</Attributes>
...
</Object>

```

This mapping entry indicates that for all the objects matching the condition, the Tag attribute in the source system, if present, then the attribute value of Tag is prefixed with the text ID#.

You can also use [IncludeAssociatedObjectAttributes](#) to modify the value of an [attribute](#), as shown below:

```

<Object>
  <IncludeAssociatedObjectAttributes>
    <AssociationType pattern="^FillsVoids$" />
    <Conditions>
      <Attribute name = "ClassName" pattern = "^OPENINGELEMENT" />
      <Attribute name = "Name" pattern = "^element1$" />
    </Conditions>
  </IncludeAssociatedObjectAttributes>
  <Attributes >
    <Attribute name="Tag" pattern="^[A-Za-z]+$" >
      <Name value="Identifier" />
      <Value value="[Tag] [associated:Tag]" />
    </Attribute>
  </Attributes>
  ...
</Object>

```

In the above case, the value of the **Tag** attribute along with the value of **Tag** of the first associated object matching the condition is used to derive the **attribute** value. You must use the prefix "**associated:**" to resolve the value from its associated objects.

```
<Object>
...
<IncludeAssociatedObjectAttributes>
  <AssociationType pattern="^FillsVoids$" />
  <Conditions>
    <Attribute name = "ClassName" pattern = "^OPENINGELEMENT" />
    <Attribute name = "Name" pattern = "^element1$" />
  </Conditions>
</IncludeAssociatedObjectAttributes>
  <Attributes >
    <Attribute>
      <Name value="Identifier" />
      <Value value="[associated:Tag]" appendSeparator=";" />
    </Attribute>
  </Attributes>
...
</Object>
```

When there is more than one associated object with a **Tag** attribute and you need to include all of them, all these values can be aggregated into a single symbol-separated list value. In the above case, the value of the **attribute Tag** present in all associated objects matching the condition is used to derive the attribute **Identifier** value. All the values are joined using the separator string ";" defined by **appendSeparator**. You must use the prefix "**associated:**" to resolve the value from its associated objects.

The following example supports the case when there are associated object attributes more than one descendant level below the main object.

```
<Object>
...
  <Conditions>
    <Attribute name = "ClassName" pattern = "^DOOR$" />
  </Conditions>
  <TemplateID id ="default" />
  <ObjectID value="[GlobalId]" />
  <IncludeAssociatedObjectAttributes>
    <AssociationType pattern="^Materials$" />
    <Conditions>
      <Attribute name = "ClassName" pattern = "^MATERIAL$" />
    </Conditions>
  </IncludeAssociatedObjectAttributes>
  <Attributes>
    <Attribute>
      <Name value="Materials" />
      <Value value="[associated{descendantLevel=2}:Name]" appendSeparator=";" />
    </Attribute>
  </Attributes>
...
</Object>
```

In the above case, the material values are in **Material** objects located two levels down in the object hierarchy: **DOOR** is associated (via the inverse relationship **RELASSOCIATESMATERIAL**) with **MATERIALLIST**, which is directly associated with two **MATERIAL** objects. You must therefore use the prefix

"associated{descendantLevel=<level>}:" to resolve the value from its associated objects that are specified to a certain level in the object hierarchy. All of these values are joined using the separator string ";" defined by `appendSeparator`.

Remove:

The simplest form of an attribute mapping entry to remove an existing attribute is shown below:

The following mapping entry removes the attribute `Tag` from an object, if the attribute `Tag` is present in the source system.

```
<Object>
...
<Attributes>
  <Attribute name="Tag" >
    <Remove />
  </Attribute>
</Attributes>
...
</Object>
```

The following mapping entry removes the attribute `Tag` from an object, if the attribute `Tag` is present in the source system and also matches the pattern.

```
<Object>
...
<Attributes>
  <Attribute name="Tag" pattern="^[A-Za-z0-9]+$" >
    <Remove />
  </Attribute>
</Attributes>
...
</Object>
```

The following mapping entry removes all the attributes matching the pattern from an object.

```
<Object>
...
<Attributes>
  <Attribute name="*" pattern="^[A-Za-z0-9]+$" >
    <Remove />
  </Attribute>
</Attributes>
```

The following mapping entry removes all the attributes present in an object.

```
<Object>
...
<Attributes>
  <Attribute name="*" >
    <Remove />
  </Attribute>
...
</Object>
```

keepUnmappedAttributes

This setting determines whether attributes that are not matched are included in the output EIWM file.

If this setting is true, then all the attributes are included in the output EIWM file.

If this setting is false, then only those attributes that have a matching mapping entry are included in the output EIWM file.

```
<Object>
...
  <Attributes keepUnmappedAttributes="true">
    <Attribute name="Tag">
      <Value value="ID#[Tag]" />
    </Attribute>
  </Attributes>
...
</Object>
```

Notes:

The keepUnmappedAttributes setting is optional. By default, the keepUnmappedAttributes attribute value is true if the setting is not provided.

The following attributes are included in the EIWM file regardless of their mapping or of the value of keepUnmappedAttributes attribute being set to false:

- GlobalID
- Name
- ObjectID
- ObjectName
- ClassID
- Context
- RevisionID
- TemplateID
- IncidentalClassification

Excluding Objects from Output Files

You can add the following special attributes to specific objects to exclude them from the various output files:

- **#EXCLUDE_FROM_EIWM#** - excludes the objects from the EIWM file (but not associations to those objects from other objects).
- **#EXCLUDE_FROM_CSV#** - excludes the objects from CSV files, both CSV reports and CSV load files.
- **#EXCLUDE_DOCUMENT_ASSOCIATIONS#** - excludes the references to Document objects (usually represented by 'is referenced in' associations) in the EIWM file.
- **#EXCLUDE_FROM_XGL#** - excludes the object's hotspot (**SELECTIONID**) tagging information from XGL/ZGL the file, if produced.

It's possible to create multiple **EXCLUDE** attributes of different types on the same object to exclude it from the relevant multiple output files. If you do not want the **EXCLUDE** attribute for one type of output file appearing as an attribute in another type of output file, then set the attribute's system parameter to "true".

For example, the following configuration will exclude from a CSV file all objects that do not have a **Class ID** defined and the **#EXCLUDE_FROM_CSV#** attribute won't be included in the EIWM file:

```
<Object>
  <Conditions>
    <Attribute name="ClassID" pattern="" />
  </Conditions>
  <Attributes>
    <Attribute system="true" >
      <Name value="#EXCLUDE_FROM_CSV#" />
      <Value value="" />
    </Attribute>
  </Attributes>
</Object>
```

Dataset Mapping

You can create a new dataset object from one or more attributes of a source object.

Attribute mapping may be configured to create Datasets associated with the source object. Each attribute present in the source object may be attached to a Dataset.

```
<Datasets>
  <Dataset id="Dataset1" >
    <DatasetID>
      <Transforms>
        <InsertAfter pattern="^.*$" value=" Dataset" />
      </Transforms>
    </DatasetID>
    <ContextID value="IPE" />
    <ClassID value="DATASET" />
  </Dataset>
</Datasets>

<ObjectMappings regExTimeoutSecs="10">
  <Object>
    <Conditions>
      <Attribute name="ClassName" pattern="Door" />
    </Conditions>
    <Attributes>
      <Attribute name="Name">
        <Dataset id="Dataset1" />
        <Name value="Door Name" />
        <Value value="[Name]" />
      </Attribute>
      <Attribute name="Description" pattern="^[A-Za-z]+$">
        <Dataset id="Dataset1" />
        <Name value="Door Description" />
        <Value value="[Description]" />
      </Attribute>
    </Attributes>
  </Object>
</ObjectMappings>
```

The above example defines a Dataset in the **IPE** context whose **ID** is based upon the **ID** of the source object with "Dataset" appended to it. The **<DatasetID>**, **<ContextID>** and **<ClassID>** elements support attribute references, transforms and lookups. The above example creates the attributes **Door Name** and **Door Description** on the Dataset.

As you move some or all the attributes from a source object into the dataset object, you may want to delete these objects from the source object. You can do this using the following syntax:

```
<ObjectMappings regExTimeoutSecs="10">
  <Object>
    <Conditions>
      <Attribute name="ClassName" pattern="PUMP" />
    </Conditions>
    <Attributes>
      <Attribute attribute="*">
        <Dataset id="Dataset1"/>
        <Remove />
      </Attribute>
    </Attributes>
  </Object>
  <Object>
    <Conditions>
      <Attribute name="ClassName" pattern="EQUIPMENT" />
    </Conditions>
    <Attributes>
      <Attribute attribute="Vendor">
        <Dataset id="Dataset1"/>
        <Remove />
      </Attribute>
    </Attributes>
  </Object>
</ObjectMappings>
```

Note: The Dataset is assigned a [ClassID](#) of 'DATASET' if not defined in the mapping.

Association Mapping

Association Mapping allows extracted associations to be customized in the output. It also permits new associations to be created in the output by transforming attributes from the source system.

A mapping entry may apply to a relationship from the source, or an attribute from the source depending upon whether it is customizing an association or generating a new one.

For each object from the source system, each mapping entry in the configuration is examined and compared against the object attributes. Each of the source object's relationships are then examined and compared to the mapping entries.

Note: Including both 'attribute' and 'relationship' qualifiers on a mapping entry generates an error.

A sample Association mapping is as follows:

```
<Object>
  ...
  <Associations>
<Association>
  <Conditions>
    <Attribute name="XYZ" pattern="^.*$"/>
  </Conditions>
  <Type value="ABC" />
</Association>
</Associations>
  ...
```

</Object>

Both forms of Association Mapping entries permit four values to be specified; the type of the association to generate, the **ID** of the object targeted by the association, the **Context ID** of the object targeted by the association and the **Class ID** of the object targeted by the association.

Association Mapping Entry Type	Description
Association Type	The <AssociationType> element specifies the association type to assign. This value supports attribute references, lookups and transforms.
Target ID	The <TargetID> element specifies the ID of the object to associate with the source object. This value is typically taken from an attribute value. However, a fixed value might be used to add an association to a specific object. This value supports attribute references, lookups and transforms.
Target Context	The <ContextID> element specifies the context of the object to associate the source object with. This value supports attribute references, lookups and transforms.
Target Class	The <TargetClassID> element specifies the class of the object to associate the source object with. This value supports attribute references, lookups and transforms.

Multi-Value Attributes

The mapping entry below demonstrates how to create multiple associations in the output from an attribute that holds multiple values. Multi-value attributes are assumed to store multiple values delimited with a separator (for example, comma, semi-colon and so on). Each value held in the attribute is assumed to be an object ID, and a separate association is created in the output for each value stored in the attribute.

Note: All associated objects must exist in the same context and have the same classification. Each generated association has the same association type.

<Object>

```

...
  <Associations>
<Association>
  <Conditions>
    <Attribute name="DecomposedOf" pattern="^.*$"/>
  </Conditions>
  <Type value="mapped_association" />
  <TargetID value="P1;P2;P3;P4" >
    <MultiValue separator=";" />
  </TargetID>
  <TargetClassID value="EQUIPMENT" />
</Association>

```

```
</Associations>
...
</Object>
```

Note: The `<MultiValue>` tag that designates the `'part_codes'` attribute as being a multivalued attribute whose values are separated with a semi-colon. The `<MultiValue>` tag is only supported for mapping attributes to associations. If specified for a mapping entry that is mapping relationships, it is silently ignored.

keepUnmappedAssociations Setting

This setting determines whether associations that are not matched are included in the output EIWM file.

If this setting is true, all the associations are included in output EIWM.

If the setting is false, then only associations that have a matching mapping entry are included in the output EIWM.

```
<Object>
...
<Associations keepUnmappedAssociations="false">
<Association>
<Conditions>
<Attribute name="DecomposedOf" pattern="^.*$"/>
</Conditions>
<Type value="mapped_association" />
<TargetID value="P1;P2;P3;P4" >
<MultiValue separator=";" />
</TargetID>
<TargetClassID value="EQUIPMENT" />
</Association>
</Associations>
...
</Object>
```

Notes:

- The `keepUnmappedAssociations` setting is **optional**. By default, the `keepUnmappedAssociations` attribute value is **true** if the setting is not provided.
- Associations containing association types, `is referenced in` and `is identified by`, are considered as **mandatory**.
- The following Object mapping can be used for Associations of a specific `Type` to be exported to the EIWM file for all objects.

```
<Object>
...
<TemplateID id="default" />
<ObjectID value="[GlobalId]"/>
<ClassID value="[ClassName]"/>
<Associations keepUnmappedAssociations="false">
  <Association relationship="FillsVoids" >
    <Type value="FillsVoids" />
  </Association>
</Associations>
...
</Object>
```


The `keepUnmappedAssociations` value must be set to **false**; the `Association relationship` and `Type` values should be same to achieve this functionality.

- The following Object mapping can be used for Associations of a specific `target` to be exported to the EIWM file for all objects.

```
<Object>
...
<TemplateID id ="default" />
<ObjectID value="[GlobalId]"/>
<ClassID value="[ClassName]"/>
<Associations keepUnmappedAssociations="false">
  <Association relationship="FillsVoids" >
    <Type value="FillsVoids" />
    <Conditions>
      <Attribute name="ClassName" pattern="OPENINGELEMENT" />
      <Attribute name="Name" pattern="M_Single-Flush:0915 x 2134mm:197506:1" />
    </Conditions>
  </Association>
</Associations>
...
```

The `keepUnmappedAssociations` value must be set to false; the `Association relationship` and `Type` values should be same and its conditions determine the target.

Expand and Interpolate

The `Expand` feature lets you create multiple associated Tags or Objects in the EIWM output from an attribute value containing text delimited with a separator (such as comma or semicolon). You can use any number of Expands to build the tags incrementally.

A separate association is created in the output for each expanded tag and the value is used as the `ObjectID` of the associated object.

Notes:

- The Expand mapping pattern must contain the expansion character to consider the attribute value for the expansion.
- If an Expand setting is defined along with a `MultiValue` setting, the `MultiValue` setting is applied first on the attribute value. Expand settings are then applied on individual values derived from the `MultiValue` setting to generate expanded/interpolated values.

Examples:

```
<Association attribute="Name" attributePattern="^[A-Z]{1}[a-z]{5}$" >
<Type value="is a part of" />
<TargetID value="P-101A/D" >
<Expand interpolate = "false">
<Pattern>[A-Z]/[A-Z]</Pattern>
<Char>/</Char>
</Expand>
</TargetID>
</Association>
```

In the above example, two tags named `P- 101A` and `P-101D` are created as the `interpolate` setting is false.

```
<Association attribute="Name" attributePattern="^[A-Z]{1}[a-z]{5}$" >
```

```
<Type value="is a part of" />
<TargetID value="P-101A/D" >
<Expand interpolate = "true">
<Pattern>[A-Z]/[A-Z]</Pattern>
<Char></Char>
</Expand>
</TargetID>
</Association>
```

In the above example, four tags named P- 101A, P- 101B, P- 101C and P-101D are created as the [interpolate](#) setting is true.

```
<Association attribute="HotSide" >
<Type value="is a part of" />
<ContextID value="IPE" />
<TargetID value="07001-QE/QT-003;07001-FE-004A/B/C" >
<MultiValue separator=";" />
<Expand interpolate = "false">
<Pattern>[A-Z][A-Z]/[A-Z][A-Z]</Pattern>
<Char></Char>
</Expand>
<Expand interpolate = "false">
<Pattern>[A-Z]/[A-Z]/[A-Z]</Pattern>
<Char></Char>
</Expand>
</TargetID>
</Association>
```

In the above example, tags named "07001-QE-003", "07001-QT-003", "07001-FE-004A", "07001-FE-004B", "07001-FE-004C" are created.

```
<Association attribute="HotSide" >
<Type value="is a part of" />
<ContextID value="IPE" />
<TargetID value="07001-QE/GT-003;07001-FE-004A/B/C" >
<MultiValue separator=";" />
<Expand interpolate = "true">
<Pattern>[A-Z]/[A-Z]</Pattern>
<Char></Char>
</Expand>
<Expand interpolate = "false">
<Pattern>[A-Z]/[A-Z]</Pattern>
<Char></Char>
</Expand>
</TargetID>
</Association>
```

In the above example, the first [Expand](#) block creates the following tags "07001-QET-003", "07001-QFT-003", "07001-QGT-003", "07001-FE-004A/B" and "07001-FE-004A/C". The second [Expand](#) block further splits tags "07001-FE-004A/B" and "07001-FE-004A/C" into "07001-FE-004A", "07001-FE-004B" and "07001-FE-004C". Finally, six tags named "07001-QET-003", "07001-QFT-003", "07001-QGT-003", "07001-FE-004A", "07001-FE-004B" and "07001-FE-004C" are created.

Association Mapping entries can be broadly classified into the following:

Create: An example of an Association mapping entry that generates an association from an attribute in the source is shown below:

```
<Associations>
<Association>
```

```
<Conditions>
<Attribute name="DTOption" pattern="^.*$"/>
</Conditions>
<Type value="is DTOption of" />
<TargetID value="ID123" />
<TargetClassID value="EQUIPMENT" />
<ContextID value="IPE" />
</Association>
</Associations>
```

Note: Conditions may hold multiple attribute elements. All conditions must be met for the mapping to be applied.

A **pattern** property can be specified to match only attributes with a value that conforms to the regular expression.

- **Update:** An example of an Association mapping entry that modifies an existing association in the source is shown below:

```
<Associations>
  <Association relationship="TestAssociation" >
    <Type value="mapped association" />
    <TargetID value="abcd1234" />
    <TargetClassID value="EQUIPMENT" />
  </Association>
</Associations>
```

- **Remove:** An example of an Association mapping entry to remove an association from the source object is shown below:

```
<Associations>
  <Association relationship="TestAssociation">
    <Remove />
  </Association>
</Associations>
```

An example of an **Association mapping** entry to remove all associations from the source object is shown below:

```
<Associations>
  <Association relationship="*" >
    <Remove />
  </Association>
</Associations>
```

Optionally, to avoid including orphaned associated objects, you can also remove the target object by using the keyword '**includeTarget**'. An example of an Association mapping entry that removes an association from the source object and associated object is shown below:

```
<Associations>
  <Association relationship="TestAssociation">
    <Remove includeTarget="true"/>
  </Association>
</Associations>
```

Using the Parent Object's Attributes When Updating Associated Objects

When updating an associated object, for example, in setting the **ObjectID** of a dataset, it might be necessary to use one of the parent object's attribute values. This is done using transformer syntax like

[parent:attributeName] to resolve attribute values from the parent object. This example assigns a parent attribute in the associated dataset object when assigning a **targetID**:

```
<Association relationship="IsDefinedBy">
  <Conditions>
    <Attribute name="ClassName" pattern="DATASET" />
  </Conditions>
  <Type value="has dataset" />
  <TargetID value="[parent:ObjectID]@[GlobalID] Dataset" />
  <TargetClassID value="Dataset"/>
</Association>
```

The prefix "parent:" can be applied when resolving values of **TargetID**, **TargetClassID**, **TargetName** and **ContextID** elements that are sub elements to the **Association** object.

The relationships between block references, nested block references and drawing items (primitive graphics) are represented in the output model by their associations of type "is a part of". To maintain these relationships, you must not remove these associations during mapping, for example, by defining mappings for other types of associations and then setting "keep unmapped associations" to **false**.

If you use unique identifiers when defining tags for structures with nested objects during mapping, you can highlight selected objects in *Workhub* and *Dashboard* on any level of the structure by selecting the objects in the Explorer's tree.

ObjectID Mapping

Object ID Mapping, also known as Identification, determines the ID assigned to an object in the output. Each mapping entry in the configuration is examined in turn and the first matching entry is used to generate the object's **ObjectID**.

The simplest form of **ObjectID** map entry to derive **ObjectID** for an object is shown below:

```
<Object>
  ...
  <Conditions>
    <Attribute name="object_class" pattern="Door" />
  </Conditions>
  <ObjectID value="[GlobalId]">
  </ObjectID>
  ...
</Object>
```

Note: Conditions may hold multiple attribute elements. All conditions must be met for the mapping to be applied.

Lookups may be used to obtain the **objectID** from an external data source.

```
<Object>
  ...
  <Conditions>
    <Attribute name="object_class" pattern="Door" />
    <Attribute name="Name" pattern="Door 100X40" />
  </Conditions>
  <ObjectID value="[GlobalId]">
    <Lookup id="ExcelLookup" >
      <FailAction action = "DiscardObject" />
    </Lookup>
  </ObjectID>
```

```
...
</Object>
```

In this case, the value of the `GlobalId` attribute is used as a key to look up a value for the `ObjectID`. If no match is found, value is derived from its fail action.

You can use `Transforms` to modify the value of an attribute, as shown below:

```
<Object>
...
<Conditions>
  <Attribute name="object_class" pattern="Door" />
</Conditions>
<ObjectID value="[GlobalId]_[object_name]">
  <Transforms>
    <Keep pattern="pump-\d{2}[A-Z]" />
    <InsertAfter pattern="^.*$" value="InsertAfter" />
    <InsertBefore pattern="^.*$" value="InsertBefore " />
  </Transforms>
</ObjectID>
...
</Object>
```

For example, if the `ObjectID` value is "3\$pEhtFpv31QFndkpGikoC_this pump-01A is new", the final tag will be "InsertBeforepump-01AInsertAfter".

You can also use attributes from associated objects to modify the value of `ObjectID` as shown below:

```
<Object>
...
<IncludeAssociatedObjectAttributes>
<AssociationType pattern="^FillsVoids$" />
<Conditions>
  <Attribute name = "ClassName" pattern = "^OPENINGELEMENT" />
  <Attribute name = "Name" pattern = "^element1$" />
</Conditions>
</IncludeAssociatedObjectAttributes>
< ObjectID value="[GlobalId]_[associated:GlobalId]" />
...
</Object>
```

In this case above, the value of the `GlobalId` attribute along with the value of `GlobalId` of the first associated object matching the condition is used in deriving `ObjectID`.

Note: The prefix "`associated:`" is used when referring to the associated object's attribute.

If an object is not associated with an `ObjectID`, it is not included in the output. This provides a means of filtering out unwanted objects. To avoid this, an entry should be created at the bottom of the mapping file that matches any object. For example:

```
<Object>
  <ObjectID value="[GlobalId]"/>
</Object>
```

Note: The log files indicate a warning message whenever multiple objects in the EIWM have a common `Object ID` value. You can then decide if these objects must be unique, so that you must modify the relevant value in the source system or choose to use another method to determine the `Object IDs`.

The following example shows the warning message in the log file:

Object with [ObjectID]-[ClassID]-[Context]: "<ObjectID value>-<ClassID value>-<Context value>" is duplicated. Please revisit the mapping configuration.

ClassID Mapping

ClassID Mapping, also known as Classification, determines the class assigned to the object in the output. Each mapping entry in the configuration is examined in turn and the first matching entry is used to generate the object's Class ID.

The simplest form of **ClassID** map entry to derive **ClassID** for an object is shown below:

```
<Object>
...
<Conditions>
<Attribute name="object_class" pattern="Door" />
</Conditions>
<ClassID value="[object_class]">
</ClassID>
...
</Object>
```

Note: Conditions may hold multiple attribute elements. All conditions must be met for the mapping to be applied.

Lookups may be used to obtain the **Class ID** from an external data source.

```
<Object>
...
<Conditions>
  <Attribute name="object_class" pattern="Door" />
  <Attribute name="Name" pattern="Door 100X40" />
</Conditions>
<ClassID value="[object_class]">
  <Lookup id="ExcelLookup" >
    <FailAction action = "Fixedvalue" value="UNKNOWN" />
  </Lookup>
</ClassID>
...
</Object>
```

In this case, the value of the **object_class** attribute is used as a key to look up a value for the **ClassID**. If no match is found, value is derived from its fail action.

You can use Transforms to modify the value of an attribute, as shown below:

```
<Object>
...
<Conditions>
  <Attribute name="object_class" pattern="Door" />
  <Attribute name="Name" pattern="Door 100X40" />
</Conditions>
<ClassID value="[object_class]">
  <Transforms>
    <Remove pattern="^[A-Za-z0-9]{3}" />
  </Transforms>
</ClassID>
...
</Object>
```

```

    </Transforms>
  </ClassID>
  ...
</Object>

```

You can also use attributes from associated objects to modify the value of **ClassID** as shown below:

```

<Object>
  ...
  <IncludeAssociatedObjectAttributes>
    <AssociationType pattern="^FillsVoids$" />
    <Conditions>
      <Attribute name = "ClassName" pattern = "^OPENINGELEMENT" />
      <Attribute name = "Name" pattern = "^element1$" />
    </Conditions>
  </IncludeAssociatedObjectAttributes>
  <ClassID value="[ClassName]_[associated:ClassName]" />
  ...
</Object>

```

In the above example, the value of the **ClassName** attribute along with the value of **ClassName** of the first associated object matching the condition is used in deriving **ClassID**.

Notes:

- The prefix "**associated:**" is used when referring to the associated object's attribute.
- All conditions must be met for the **ClassID** mapping to be applied.

If an object does not match any of the mapping entries, it is assigned a class ID of **UNKNOWN**.

You can use the following mapping to create an object where all tags similar to **V-101** are created with a **ClassID** of **VESSEL**:

```

<Object>
  ...
  <Conditions>
    <Attribute name = "block name" pattern="V-\d{3}" />
  </Conditions>
  <ObjectID value = "[block name]" />
  <ClassID value="VESSEL" />
  ...
</Object>

```

Context Mapping

Context Mapping determines the ID of the context within which an Object ID resides.

Each mapping entry in the configuration is examined in sequence and the first mapping entry to match is used to generate the object's context ID.

The simplest form of **ContextID** map entry to derive Context for an object is shown below:

```

<Object>
  ...
  <Conditions>
    <Attribute name="object_context" pattern="ROOT" />
  </Conditions>
  <ContextID value="[object_context]" />

```

```
...
</Object>
```

You can specify multiple levels of context using the '|' character to separate levels:

```
<Object>
...
<Conditions>
  <Attribute name="object_context" pattern="ROOT" />
  <Attribute name="object_subcontext" pattern="PARENT" />
  <Attribute name="object_Childcontext" pattern="CHILD" />
</Conditions>
<ContextID value="ROOT|PARENT|CHILD">
</ContextID>
...
</Object>
<Object>
...
<Conditions>
  <Attribute name="object_context" pattern="ROOT" />
  <Attribute name="object_subcontext" pattern="PARENT" />
  <Attribute name="object_Childcontext" pattern="CHILD" />
</Conditions>
<ContextID value="[object_context]|[object_subcontext]|[object_Childcontext]">
</ContextID>
...
</Object>
```

Note: Conditions may hold multiple attribute elements. All conditions must be met for the mapping to be applied.

You can use Lookups and transforms to modify context value.

```
<Object>
...
<Conditions>
  <Attribute name="object_context" pattern="ROOT" />
</Conditions>
<ContextID value="[object_context]">
  <Transforms>
    <Replace pattern="\d{2}" value="yyy" />
  </Transforms>
</ContextID>
...
</Object>
<Object>
...
<Conditions>
  <Attribute name="object_context" pattern="ROOT" />
</Conditions>
<ContextID value="[object_context]">
  <Lookup id="csv Map" >
    <FailAction action = "EmptyValue" />
  </Lookup>
</ContextID>
...
</Object>
```


You can also use objects from associated objects to modify the value of [ContextID](#), as shown below:

```
<Object>
...
<IncludeAssociatedObjectAttributes>
<AssociationType pattern="^FillsVoids$" />
<Conditions>
<Attribute name = "ClassName" pattern = "^OPENINGELEMENT" />
<Attribute name = "Name" pattern = "^element1$" />
</Conditions>
</IncludeAssociatedObjectAttributes>
<ContextID value="[ClassName]_[associated:ClassName]" />
...
</Object>
```

In this case, the value of the `ClassName` attribute along with the value of `ClassName` of the first associated object matching the condition is used in deriving `ContextID`. The prefix "[associated:](#)" is used when referring to the associated object's attribute.

If an object does not match any of the mapping entries, it is not assigned any context. To set a default context that is used if none of the other mapping entries match, an entry with no qualifiers should be created at the bottom of the mapping file:

```
<Object>
...
<ContextID value="AVNGATE" />
...
</Object>
```

ObjectName Mapping

ObjectName Mapping determines the name of the object. Each mapping entry in the configuration is examined in sequence and the first mapping entry to match is used to generate the object's context ID.

The simplest form of ObjectName map entry to derive Object Name for an object is shown below:

```
<Object>
...
<Conditions>
<Attribute name="object_class" pattern="Door" />
</Conditions>
<ObjectName value="[object_name]">
</ObjectName>
...
</Object>
```

Note: Conditions may hold multiple attribute elements. All conditions must be met for the mapping to be applied.

You can also use Lookups and transforms to modify the [ObjectName](#) value.

```
<Object>
...
<Conditions>
<Attribute name="object_class" pattern="Door" />
</Conditions>
<ObjectName value="[object_class]">
<Transforms>
<Replace pattern="\d{2}" value="yyy" />
</Transforms>
</ObjectName>
...
</Object>
```

```

    </Transforms>
  </ObjectName>
  ...
</Object>
<Object>
  ...
  <Conditions>
    <Attribute name="object_class" pattern="Door" />
  </Conditions>
  <ObjectName value="[object_class]">
    <Lookup id="ExcelLookup" >
      <FailAction action = "DiscardElement" />
    </Lookup>
  </ObjectName>
  ...
</Object>

```

You can also use objects from associated objects to modify the value of **ObjectName** as shown below:

```

<Object>
  ...
  <IncludeAssociatedObjectAttributes>
    <AssociationType pattern="^FillsVoids$" />
    <Conditions>
      <Attribute name = "ClassName" pattern = "^OPENINGELEMENT" />
      <Attribute name = "Name" pattern = "^element1$" />
    </Conditions>
  </IncludeAssociatedObjectAttributes>
  <ObjectName value="[Name]_[associated:Name]" />
  ...
</Object>

```

In the above case, the value of the **Name** attribute along with the value of **Name** of the first associated object matching the condition is used in deriving **ObjectName**. The prefix "**associated:**" is used when referring to the associated object's attribute.

If an object does not match any of the mapping entries, it is not assigned any **ObjectName**.

RevisionID Mapping

RevisionID Mapping can be used to derive the revision of a document. Each mapping entry in the configuration is examined in sequence and the first mapping entry to match is used to generate the object's context ID.

The simplest form of **RevisionID** mapping entry to derive Revision or an object is shown below:

```

<Object>
  ...
  <Conditions>
    <Attribute name="object_class" pattern="Door" />
  </Conditions>
  <Revision value="[object_name]" />
  ...
</Object>

```

Note: Conditions may hold multiple attribute elements. All conditions must be met for the mapping to be applied.

You can also use Lookups and transforms to modify the Revision value.

```
<Object>
...
<Conditions>
  <Attribute name="object_class" pattern="Door" />
</Conditions>
<Revision value="[object_name]">
  <Transforms>
    <Replace pattern="^[a-z]{3}" value="OBJ" />
  </Transforms>
</Revision>
...
</Object>
<Object>
...
<Conditions>
  <Attribute name="object_class" pattern="Door" />
</Conditions>
<Revision value="[object_name]">
  <Lookup id="ExcelLookup" >
    <FailAction action = "DiscardElement" />
  </Lookup>
</Revision>
...
</Object>
```

You can also use attributes from associated objects to modify the value of [RevisionID](#) as shown below:

```
<Object>
...
<IncludeAssociatedObjectAttributes>
  <AssociationType pattern="^FillsVoids$" />
  <Conditions>
    <Attribute name = "ClassName" pattern = "^OPENINGELEMENT" />
    <Attribute name = "Name" pattern = "^element1$" />
  </Conditions>
</IncludeAssociatedObjectAttributes>
<ObjectName value="[Tag]_[associated:Tag]" />
...
</Object>
```

In this case above, the value of the [Tag](#) attribute along with the value of [Tag](#) of the first associated object matching the condition is used in deriving [RevisionID](#). The prefix "[associated:](#)" is used when referring to the associated object's attribute.

If an object does not match any of the mapping entries, it is not assigned any [RevisionID](#).

Incidental Classification Mapping

Incidental classification mapping determines the state of an object in AIM. Each mapping entry in the configuration is examined in turn and the first entry to match is used to generate the Incidental classification.

A simple example to define Incidental classifications for a group of objects matching the condition on source system is as follows:

```
<Object>
```

```

...
<Conditions>
  <Attribute name="Class" pattern="ELEMENT" />
  <Attribute name="object_name" pattern="test" />
</Conditions>
<IncidentalClassifications>
  <IncidentalClassification value="void" />
</IncidentalClassifications>
...
</Object>

```

An object may be associated to multiple Incidental classifications. You can also use Lookups and transforms to modify the Incidental classification value.

```

<Object>
  ...
  <Conditions>
    <Attribute name="Class" pattern="ELEMENT" />
    <Attribute name="object_name" pattern="test" />
  </Conditions>
  <IncidentalClassifications>
    <IncidentalClassification value="Deleted" >
      <Transforms>
        <Remove pattern="_[a-z]{3}$" />
        <LowerCase />
      </Transforms>
    </IncidentalClassification>
    <IncidentalClassification value="void" />
  </IncidentalClassifications>
  ...
</Object>

```

Search Criteria Mapping

Search Criteria Mapping applies pattern matching recursively so that all matching tags can be identified from an attribute's value and multiple tags can be created.

The search criteria mapping entry to create new tags is shown below:

```

<SearchCriteria attribute = "Content">
<Search>
<Patterns>
<Pattern value = "[A-Z]-\d{3}"/>
<Pattern value = "[A-Z]-\d{3}[A-Z]"/>
</Patterns>
<ClassID value = "Document" />
<ContextID value = "Test"/>
<TemplateID id = "default" />
<Attributes keepUnmappedAttributes = "true">
<Attribute>
<Name value = "Description" />
<Value value = "[Content]" />
</Attribute>
</Attributes>
<Associations>
<Association>

```

```

<Type value = "Modified association" />
<TargetID value = "abc123" />
<TargetClassID value = "Document" />
</Association>
</Associations>
</Search>
</SearchCriteria>

```

This mapping entry searches for matching patterns on the value of the attribute and creates the new tags if any patterns are matched.

SearchCriteria is the main block where you can add one or more search blocks. Search is a sub-block where you can add Patterns, containing one or more Pattern values. These Patterns will be applied on the attribute's value. The entries for ClassID mapping, Context mapping, Template mapping, Attribute mapping and Association mapping are to be applied for each newly identified tag from the given Pattern value. For each identified tag, a new object will be created and its object ID is by default assigned with the identified tag.

Once a text matches a pattern, the relevant tag is created and then no other patterns in the mapping sequence of a search block will be applied to that tag's text. If multiple Search blocks are configured within the **SearchCriteria**, then all unmatched parts of the text will be subject to pattern mapping of the subsequent Search blocks.

For example, if an extracted object contains the text "P-101ASDFP-102ASDFP-103" and tags to be identified are P-101A and P-102A, then the Search Pattern will be:

```

<Search>
<Patterns>
  <Pattern value = "[A-Z]-\d{3}[A-Z]"/>
</Patterns>
</Search>

```

If two Patterns are defined:

```

<Search>
  <Patterns>
    <Pattern value = "[A-Z]-\d{3}[A-Z]"/>
    <Pattern value = "[A-Z]-\d{3}"/>
  </Patterns>
</Search>

```

The first Pattern successfully identifies the tags P-101A and P-102A, so the second Pattern will not be applied on the remaining part of the text "SDF" and "SDFP-103".

If the order of the Patterns is reversed:

```

<Search>
  <Patterns>
    <Pattern value = "[A-Z]-\d{3}"/>
    <Pattern value = "[A-Z]-\d{3}[A-Z]"/>
  </Patterns>
</Search>

```

This Search results in the tags P-101 and P-102 and P-103.

If you need to identify P-101A, P-102A and P-103 tags, then use two Search blocks as:

```

<SearchCriteria attribute = "Content">
<Search>
<Patterns>
<Pattern value = "[A-Z]-\d{3}[A-Z]"/>

```

```

</Patterns>
</Search>
<Search>
  <Patterns>
    <Pattern value = "[A-Z]-\d{3}"/>
  </Patterns>
</Search>
</SearchCriteria>

```

Here, the identified tags will be **P-101A** and **P-102A** from the first search block and **P-103** from the remaining part of the text "**SDF**" and "**SDFP-103**" that is searched in the second Search block.

Expand Attribute Mapping

Expand Attributes mapping treats a separator-split string in an attribute as a list with syntax to define the separation character and to identify each element of the list by its index value.

The **ExpandAttributes** element can have two attributes, **name** and **separator**. The default separator is comma. You can have multiple attribute blocks, which permit the **Name** and **Value** to be specified. The **Name** attribute cannot be empty.

For example, suppose an input attribute called **Content** has the value **Material, Steel**, you can create a new attribute called **Material** with a Value of **Steel** using the following mapping.

Note: As the separator is not defined, the default value of comma is assumed.

```

<ExpandAttributes name = "Content">
  <Attribute>
    <Name index = "1" />
    <Value index = "2" />
  </Attribute>
</ExpandAttributes>

```

If the input **Content** attribute has the value "**Pump1 : 10 : kPa**", then a new property can be created with the below mapping.

```

<ExpandAttributes name = "Content" separator = ":" >
  <Attribute>
    <Name index = "1" />
    <Value index = "2" />
    <Units index = "3" />
  </Attribute>
</ExpandAttributes>

```

The values of **Name** element, **Value** element and **Units** element are taken directly from the index positions 1-3 of the split list, to create a property with **Name** as **Pump** and **Value** as 10 and Units as kPa in the output EIWM.

New attributes can also be created using the defined value. Consider the following Expand Attribute mapping:

```

<ExpandAttributes name = "Content" separator = ":">
  <Attribute>
    <Name value = "Pump" />
    <Value value = "Sequence" />
  </Attribute>
</ExpandAttributes>

```

This mapping entry adds a new attribute whose **Name** will be **Pump** and **Value** will be **Sequence**.

Either value or index attribute can be used with [Name](#), [Value](#) and [Units](#) elements, but not both for the same element.

3D Model Hierarchies

Identifying the graphical structure of branch entities, for example, having all the pipes and connectors in a pipeline highlighted when selecting a pipeline in AIM, can be done by creating "is a part of" associations between the relevant child engineering objects (for example, pipes) and the engineering object higher up in the hierarchy that 'owns' them (for example, pipeline).

In Smart 3D models, entities can have multiple nested hierarchies where the child of one entity may be a parent for another entity that can contain children. The Gateway automatically creates the relevant "is a part of" associations between these child objects and their parents.

When viewed in AIM, the hierarchy is reflected in the object explorer tree structure.

Presentation Mapping

Presentation Mapping is used to adjust the output XGL and EIWM properties such as [materials](#) and [colours](#), or to segment the input model to be included in the XGL rendition. Using Presentation mapping extension in the main Transform Configuration file is optional.

The following example shows all sections of presentation mapping configuration with description of available features. Presentation Mapping contains three sections: Materials, Colours and Segments.

Section with Materials Mapping

This section is used to map the material used in the model in *AIM Dashboard*. These values can then be used with the 3D Materials functionality in *AIM Dashboard*, for example, to highlight or hide all objects in the model that have the same material value. The syntax allows you to map any attribute to any [toMaterial](#) value.

Example:

```
<materials>
<material fromAttribute="ClassId" fromValue="Wall" toMaterial="Walls"/>
<material fromAttribute="ClassId" fromValue="OPENINGELEMENT" toMaterial="OPENINGS"/>
<material fromAttribute="ClassId" fromValue="DOOR" toMaterial="DOORS"/>
<material fromAttribute="ClassId" fromValue="WINDOW" toMaterial="WINDOWS"/>
</materials>
```

Section with Colours Mapping

The section is used to model colours in *AIM Dashboard*. The syntax contains two abilities to map colours: [fromAttribute](#) and [fromColour](#). It is possible to map colours from any attribute value or specific [colour](#) attribute value to a [toColour](#) attribute.

Example:

```
<colours>
  <colour fromAttribute="Name" fromValue="P100" toColour="green"/>
  <colour fromAttribute="ClassId" fromValue="PART" toColour="#008000"/>
  <colour fromAttribute="autocad color index" fromValue="71" toColour="red"/>
  <colour fromColour="magenta" toColour="130,100,130"/>
  <colour fromColour="#455050" toColour="blue"/>
  <colour fromColour="255,10,0" toColour="#F3F3F3"/>
</colours>
```

The `fromColour` method maps the input colour to a new colour. You can use following methods of coloring:

- RGB, for example, 128, 0, 128.
- Known colours (HTML colour standard), for example, Red.
- HTML hexadecimal: `#E3D3D3`.
- Wildcard "*" in `fromColour` attribute to map all colours not affected by previous colour mapping items.

Section with Segmentation of Data:

This section is used to segment the model to produce a subset or subsets of graphical objects into XGL files.

You can segment a model in four ways:

- According to any attribute value:

This segments the model according to any attribute value. For example, `fromValue="*" fromAttribute="Pressure"` generates a model that contained only the objects with a `Pressure` attribute of any value.

- According to extents of the geometric model (min and max points and additional `includeParts` attributes):

This segments the model by its extents represented by two points max and min. The `includeParts="true"` (or false) attribute determines whether objects that cross this border are included. By default, the attribute value is false.

- According to any association via three types of attributes:

- `fromAssociation`: The value provides the associated object's name.
- `fromTargetAttribute`: The value provides the associated object's attribute name.
- `fromTargetValue`: The value provides the associated object's attribute value.

- When an object is not handled by any segment node:

`<segment toSuffix="<nameOfSuffixForModel>" />` handles all graphical objects not handled by any of the previously selected segments.

Note: If this segment method is used first, then it exports the complete model. If it is used as the last segment method, then it exports the remainder of objects not handled by any other segment node.

- Segment the output file into pieces up to a given size in MB using following syntax:

```
<segment
  maxOutputSize="100"
  outputType="xgl"
  separator="_"
  suffixNumberFormat="DDD"
/>
```

- `maxOutputSize` is required and specifies the maximum output size in MB.
- `outputType` is required and must be set as 'xgl'.
- `separator` is optional, it represents a text string to be inserted between the main name and the numbered suffix. Use characters that are valid for file names in your operating system.
- `suffixNumberFormat` is optional. It formats the numbering representation at the end of the file name. Defines how many decimal places are always displayed. Only 'D' characters allowed.

Example:

```
<segments>
  <segment toSuffix="" />
  <segment fromAttribute="#MODEL_NAME#" fromValue="*" to="#MODEL_NAME#" />
  <segment fromAttribute="ClassId" fromValue="*" toSuffix="" />
  <segment fromAttribute="ClassId" fromValue="WALLSTANDARDCASE"
toSuffix="_Only_WALLSTANDARDCASE"/>
  <segment min="2333.223,0.232,100000" max="50000,1000,200000" toSuffix="_SegmentA"/>
  <segment fromAssociation="IsPartof" fromTargetAttribute="Name"
fromTargetValue="Level 1" toSuffix="SegmentB"/>
  <segment min="50000,1000,200000" max="150000,11000,250000" includeParts="true"
toSuffix="_SegmentC"/>
  <segment maxOutputSize="100" outputType="xgl" separator="_"
suffixNumberFormat="DDD"/>
</segments>
```

Limitations: The segmentation size is relative to XGL output, so if the XGL Loader configuration is set to ZGL, then each output file will be a compressed version of that XGL content and their sizes will be much less than `maxOutputSize`.

Example of segmented output with no separator and suffixNumberFormat:

```
Building1.xgl
Building2.xgl
Building3.xgl
```

Example of segmented output with separator=" " and suffixNumberFormat="DDD":

```
Building_001.xgl
Building_002.xgl
Building_003.xgl
```

You can name the segment by using attributes:

- **to:**
For example, `to="#MODEL_NAME#"` - Sets the segment to value of attribute `'#MODEL_NAME#'`.
- **toSuffix:**
For example, `toSuffix="_seg1"` - Adds suffix to current name. For example, if input model is `'house'` then segment name will be `'house_seg1'`.

Chapter 5 Appendix B: CSV Reporting

The extracted data and the way they are transformed can be inspected at any point during the processing by creating a CSV report from the selected data. The selection of data to be included in CSV reporting is defined by the Transform and Loader configuration extensions. CSV files may include attributes or associations, which are selected in `columns` keyword in the configuration file. Each transformation configuration extension represents a step in the transformation process, so you should position this CSV reporting configuration at the appropriate transformation step to access the state of the data due to all previous transformation steps. A CSV report of the final transformed data that is converted to EIWM format can also be generated in the loader configuration.

- **Reporting during transformation configured in Transform configuration file**

The Gateway generates the CSV report during transformation and customizes the type of the information written to the CSV file. You can also generate multiple reports by using different suffixes and output paths for CSV files.

Example 1: CSV Reporting section with basic attributes, from Transform configuration file

```
<extension name="CSVReporting">
  <csvReport
    suffix="_AfterTransformation"
    addHeaderRow="true"
    columns="ClassID, ObjectID, ClassName, Name, #Association:Referenced"
  />
</extension>
```

Example 2: CSV Reporting section with tracking attributes, from Transform configuration file:

```
<extension name="CSVReporting">
  <csvReport
    suffix="_AfterTransformation"
    addHeaderRow="true"
    columns="#InternalId#, ClassName">
    <column value="#Tracking:
    DateTime
    ModuleName
    EventType
    ChangedMember
    PrevAttributeName
    AttributeName
    PrevAttributeValue
    AttributeValue
    PrevAssociationType
    AssociationType
    ChangedMember
    PrevTargetInternalId
    TargetInternalId
    #Filter: EventType: AttributeChanged"
  />
  </csvReport>
</extension>
```

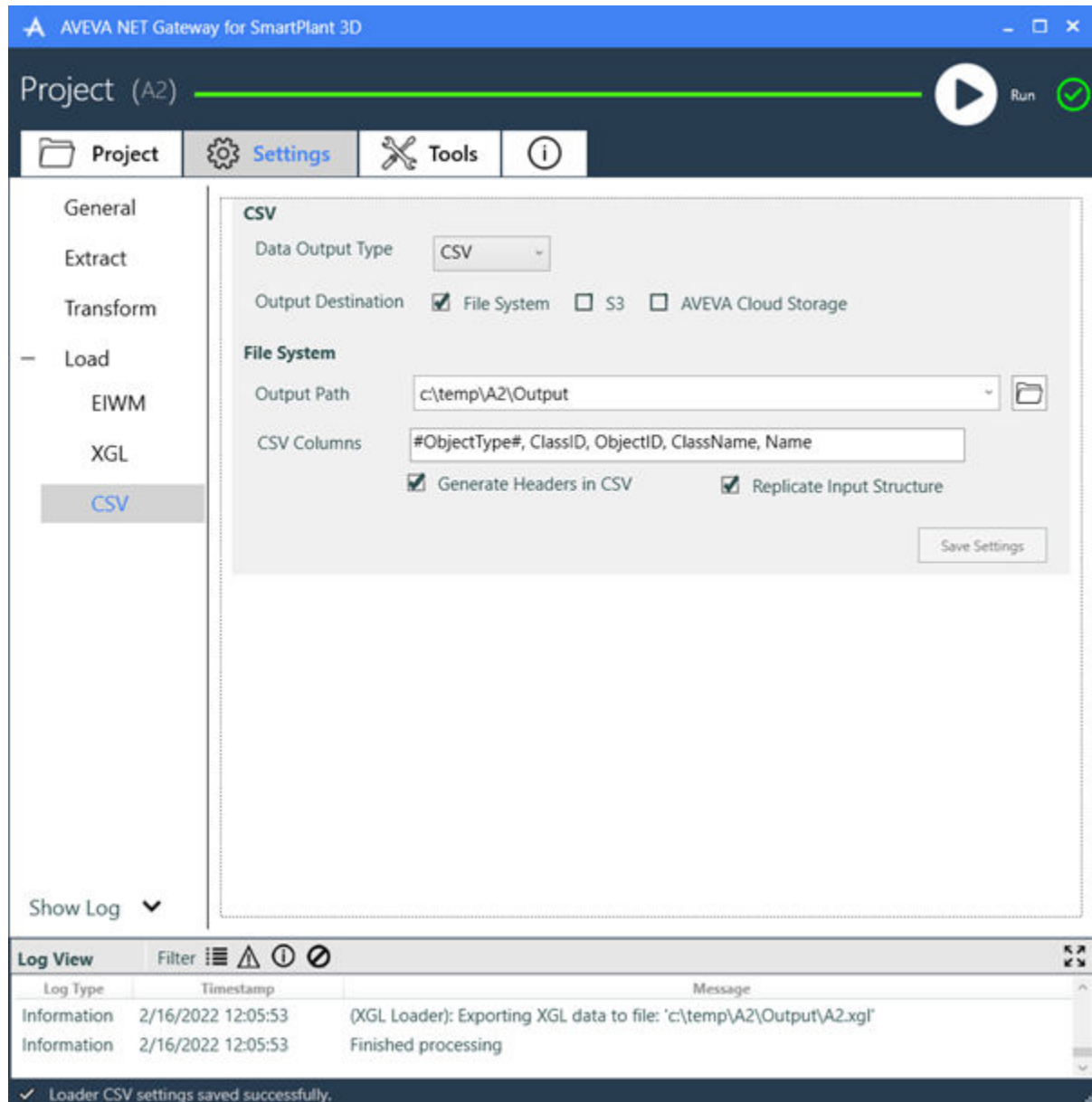
Note: The optional 'annotations' feature described in annotation provides additional information which also can be reported. In the above example, after keyword `#Tracking:` there are listed tracking attributes that

store data about creation, modification and deletion operations and can be inspected via CSV reporting as any other information.

- **Reporting after transformation configured from GUI**

This CSV report is generated when **CSV** is selected in **Data Output Type** combo list in the **Load>CSV** tab. This report reflects the extracted data state after all transformations. A report file with a **.CSV** extension is generated in the location set in the **Output Path** field.

- **GUI Settings in the Load > CSV tab:**



Settings

The following list details about the CSV Report features for the **Transform** and the **Load** configurations:

- `columns = "<keywords, names...>":` (**Mandatory**) Sets number of columns, each separated by a comma.

- **#Attribute: <name>** - (**Optional**) Sets the name of the attribute to select the attribute values that will be shown in the CSV column for all extracted data entities. Name of the attribute can also be written without **#Attribute**. **#Attribute** is added to maintain consistency with other keywords. Besides standard objects' attributes, you can also collect different data using keywords:
 - **#InternalID#** - Specifies the Internal Globally Unique Identifier (GUID) of the object.
 - **#SourceID#** - If available, the source ID of the object specified in the source data, for example, the STEP line number in IFC files.
 - **#ObjectType#** - Specifies the internal type of the object, for example: **EngineeringObject**, **GraphicalObject2D**.
- These attributes cannot be changed but may be helpful in tracking the history of the objects or filtering the CSV report by Object Type.
- **#Association: <type>** - Checks if specified association type is available in objects. CSV cells indicate **AVAILABLE** status if the corresponding object contains selected association type.
- **#Associations** - Checks if associations are available in objects. All types of associations are present in the extracted data and each CSV column represents one type of association. CSV cells indicate **AVAILABLE** status if the corresponding object contains association type defined in the CSV column.
- **#Association: <type>** and **#TargetAttribute: <name>** - Used together to first set the name of the association and then the name of the attribute whose value should be shown in the CSV column.

Note: An association may point to many referenced objects and in this case values of attributes of all these objects are added to the cell in the CSV column, each separated by a comma.

- **#Unit** - Used only with one of the three keywords: **#Attribute: <name>**, **#Attributes** or **#TargetAttribute: <name>**. If attribute contains unit, it is added to the value after putting a single space.
- **#Tracking: <tracking attributes names>** - This keyword can be used with the list of Tracking attributes' names separated by a blank character (for example, space, tabulation or a new line), where each blank character represents data in a specific column. All recorded changes (**Tracking entries**) of extracted data are listed in a chronological order in the CSV cell (see **Example 2** of CSV Document).
- **#Tracking: <tracking attributes names> #Filter: <tracking attribute name> : <RegEx expression>** - The **#Filter** keyword includes only those changes in the CSV report where selected tracking attribute name matches the **RegEx**.
- **column = "<keywords, names...>"**: Separates long and complex descriptions of columns and defines each description in an XML element (see **Example 2**). The separate **"column"** element adds more clarity.
- The **column** element contains a **#Filter** with a regular expression.

Note: **Columns** and **column** can be used interchangeably or both at the same time.

The following list details about the settings applicable for the **CSV Reporting** configuration:

- **suffix = "<file name suffix>":** (**Optional**) Defines a suffix to add to the CSV file name. It is used only in Transform configuration.
- **outputFolder = "<output folder path>":** (**Optional**) Stores the CSV files in the location indicated by this setting. Otherwise, files are passed to the output folder set in Load settings. It is used only in Transform configuration.

- addHeaderRow = "<true/false>" (in **Generate Headers in CSV** checkbox): (Optional) Sets to "True" to write the names of the attributes or associations to the first row of the CSV file. This setting is enabled by default.

Tracking attributes names which can be used with keyword #Tracking:

Tracking Attribute	Description
DateTime	Date and time of the change
ModuleName	Name of the component which introduces change
EventType	Type of the introduced change
ChangedMember	Affected type of the data: attribute's name, value, unit; or association's type or referenced object
PrevAttributeName	Name of the attribute before the change
AttributeName	Name of the attribute after the change
PrevAttributeValue	Value of the attribute before change
AttributeValue	Name of the attribute after the change
PrevAssociationType	Type of the association before the change
AssociationType	Type of the association after the change
PrevTargetInternalId	Internal ID of the associated object before replacing to new one
TargetInternalId	Internal ID of new associated object

List of the EventType types which can appear in CSV in the EventType column:

- [ObjectModified](#), [ObjectAdded](#) and [ObjectRemoved](#)
- [AssociationAdded](#), [AssociationChanged](#), [AssociationReplaced](#), [AssociationRemoved](#), [AssociationMoved](#) and [AssociationsCleared](#)
- [AttributeAdded](#), [AttributeChanged](#), [AttributeReplaced](#), [AttributeRemoved](#), [AttributeMoved](#) and [AttributesCleared](#)

Example 1 of CSV document viewed in Excel

	A	B	C	D	E	F	G	H
1	ClassID	ObjectID	ClassID	Name	#CURRENT_USER#	#DATE_TIME#	autocad type	block name
2	3D Model	X1	3D Model	N/A	USER123	16/05/2018 20:46	N/A	N/A
3	N/A	fdd8bc95-c49f-40de-aeae-39738e3ba18b	EQUIPMENT	N/A	N/A	N/A	BlockReference	EQ1
4	N/A	58044675-12f5-4dd8-9a64-bf53f1cba695	EQUIPMENT	N/A	N/A	N/A	BlockReference	EQ2
5	N/A	a0da42b7-7b3c-484f-98ce-49aaebca0052	UNKOWN	N/A	N/A	N/A	BlockReference	DESC
6	N/A	8a3a1928-f5f4-44de-aad7-4ff169503e72	UNKOWN	N/A	N/A	N/A	BlockReference	DESC
7	N/A	181f2502-df43-47c0-9abc-cfe3c187db88	EQUIPMENT	N/A	N/A	N/A	3dSolid	EQ1
8	N/A	03931bd2-9082-4db2-8a56-326e01ef61a8	EQUIPMENT	N/A	N/A	N/A	3dSolid	EQ1

Example 2 of CSV document with history of changes viewed in Excel

	A	B	C	D	E
1	#InternalID#	#Tracking:ModuleName	#Tracking:EventType	#Tracking:AttributeName	#Tracking:PrevAssociationType
		BaseMapping	AttributeAdded	TemplateID	N/A
		BaseMapping	AttributeAdded	ClassID	N/A
		BaseMapping	AttributeAdded	ObjectID	N/A
2	ca532181-e71f-4846-97cf-0394596dd3e3	BaseMapping	AttributeAdded	keepUnmappedAttributes	N/A
		BaseMapping	AttributeAdded	TemplateID	N/A
		BaseMapping	AttributeAdded	ClassID	N/A
		BaseMapping	AttributeAdded	ObjectID	N/A
3	936b0f8d-6b66-4776-b328-7456e7c6326f	BaseMapping	AttributeAdded	keepUnmappedAttributes	N/A
		BaseMapping	AttributeAdded	TemplateID	N/A
		BaseMapping	AttributeAdded	ClassID	N/A
		BaseMapping	AttributeAdded	ObjectID	N/A
4	bcbc8abe-5011-41a5-802a-76e85b3167ed	BaseMapping	AttributeAdded	keepUnmappedAttributes	N/A
		BaseMapping	AttributeAdded	TemplateID	N/A
		BaseMapping	AttributeAdded	ClassID	N/A
		BaseMapping	AttributeAdded	ObjectID	N/A
5	40fb8a58-9eed-42ba-a906-31b021832317	BaseMapping	AttributeAdded	keepUnmappedAttributes	N/A
		BaseMapping	AttributeAdded	TemplateID	N/A
		BaseMapping	AttributeAdded	ClassID	N/A
		BaseMapping	AttributeAdded	ObjectID	N/A
6	f7a6b4b6-2cf4-4579-b4bb-1fc3ecb94f7d	BaseMapping	AttributeAdded	keepUnmappedAttributes	N/A
		BaseMapping	AttributeAdded	TemplateID	N/A
		BaseMapping	AttributeAdded	ClassID	N/A
		BaseMapping	AttributeAdded	ObjectID	N/A
7	aff516ca-e8b3-41ee-82c1-ff8a285babc1	BaseMapping	AttributeAdded	keepUnmappedAttributes	N/A
		BaseMapping	AttributeAdded	TemplateID	N/A
		BaseMapping	AttributeAdded	ClassID	N/A
		BaseMapping	AttributeAdded	ObjectID	N/A
8	51431b16-959d-406e-9633-ea081048dce8	BaseMapping	AttributeAdded	keepUnmappedAttributes	N/A

Note: If you want to open CSV reporting in a spreadsheet form, you must extend rows and columns of cells to see complete content.

Object Parameters Available for the CSV Report

The data available to select as columns in the CSV Report are as follows:

- [ObjectID](#)
- [ClassID](#)
- [ClassName](#)
- [#Attributes](#) (selects all attributes)
- [#Associations](#) (selects all associations)
- [#InternalID#](#)
- [#SourceID#](#)
- [#ObjectType#](#)
- any attribute and property name defined for the entities, defined in property sets.

When a relationship is used, the property names from each relationship get concatenated, for example, [HasAssociations:Materials:Name](#).

Note: Objects containing an attribute "[#EXCLUDE_FROM_CSV#](#)" will not be included in the output .csv file. For more information, see [Attribute Mapping](#) ().

Chapter 6 Appendix C: Tracking Changes in Data

The optional '**annotations**' feature allows you to store additional data about how each object, attribute or association have been changed due to processing of data under Extract, Transform and Load components.

These tracking attributes store data about creation, modification and deletion operations and can be inspected via CSV reporting. You can collect this information for the CSV report by setting XML `<annotations>` node to the respective module configuration in the Extract or Load settings or to extension configuration for the Transformer. The available annotations level values are '**None**' and '**Basic**'. When the annotation level is set to **None**, no tracking information is collected. When the annotation level is set to **Basic**, then all tracking information connected with creating, updating and deleting performed on objects, attributes and associations are recorded.

Example for Extractor and Loaders:

```
<configuration ...>
...
<annotations level = "None" />
</configuration>
```

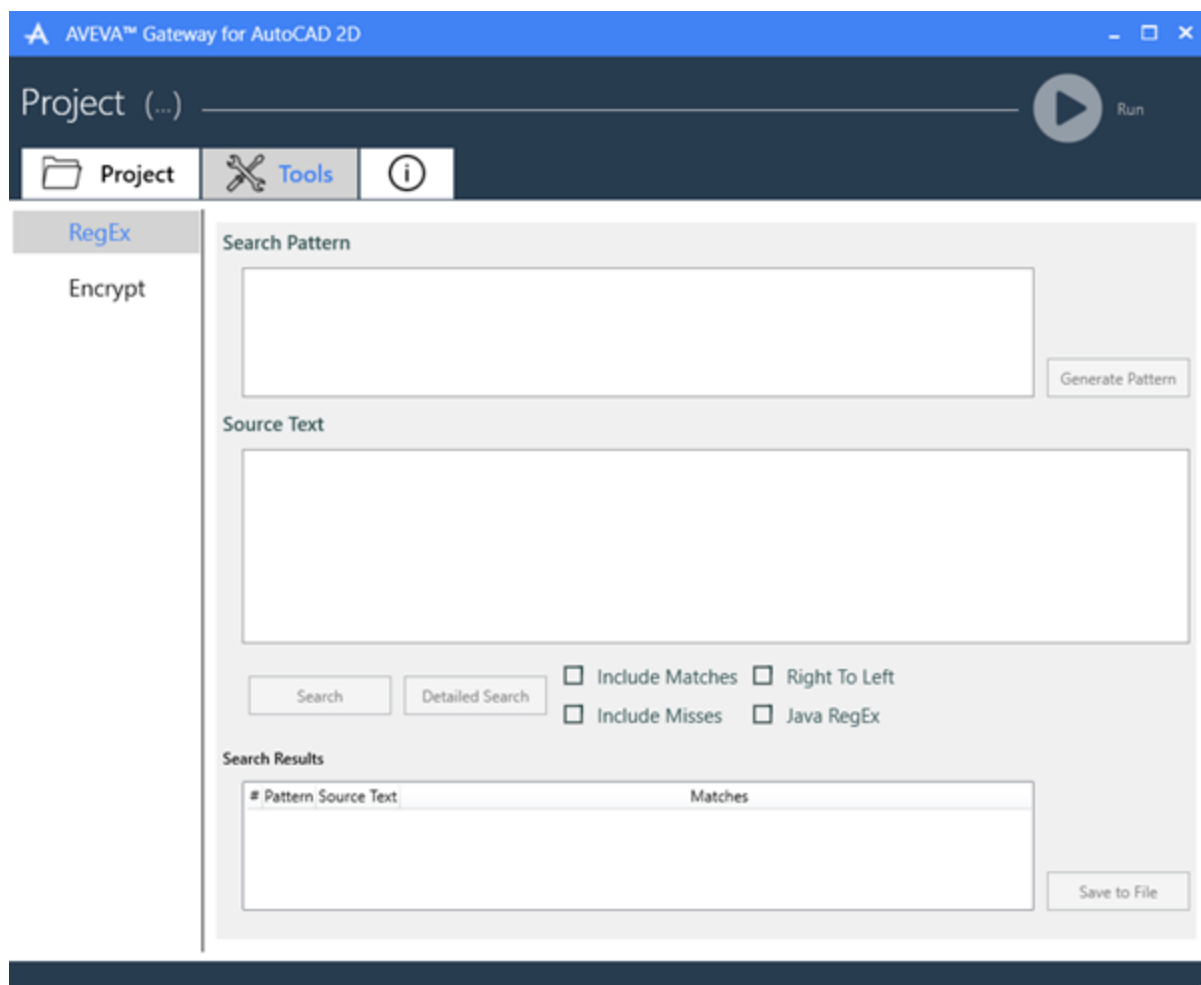
Example for Transformer's Extensions:

```
<extension ...>
...
<annotations level = "Basic" />
</extension>
```

For more information about passing report of tracking data to CSV, see [Appendix B: CSV Reporting](#) ().

Appendix D: RegEx Utility

A regular expression (RegEx) is a special text string to describe a search pattern.



The following table lists the various RegEx utility options and their various user actions.

RegEx Utility Options	User Action
Search Pattern	Select this to specify a regular expression search pattern or patterns. Multiple patterns can be specified by putting each pattern on a new line.
Generate Pattern	Select this to generate a regular expression search pattern.
Source Text	Type or paste the source text in the source text section. This activates the Generate Patterns button and generates Regular Expression patterns in the Search Pattern box that match the lines in the Source Text box.
Import Source	Click this to import the source text.
Search	Select this to apply the patterns to the source text. The summary of searches is listed in the Search Results box.
Detailed Search	Select this to search each line of the source text in each pattern. If a pattern finds a match on a line that line is no longer considered by subsequent patterns.

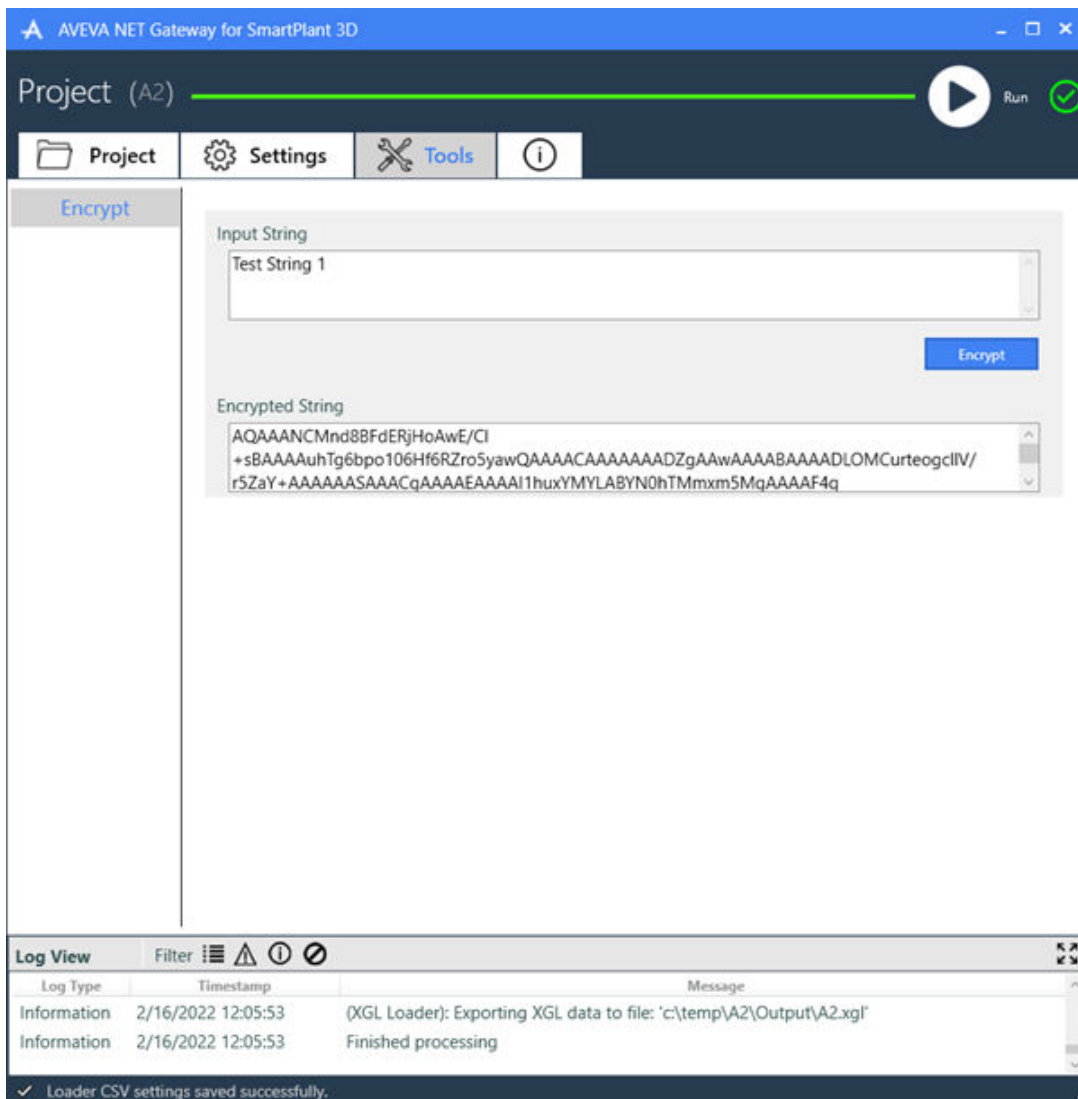
RegEx Utility Options	User Action
Include Misses	Select this to include the lines where a match is not found for specified pattern(s) in the search results.
Include Matches	Select this option to include the lines where a match is found for specified pattern(s) in the search results.
Right to Left	Select this option to allow the patterns to start searching from the end of lines in the source text and not the start.
Java RegEx	Select this option to employ the Java flavor of RegEx.
Search Results	Use the search results that is displayed indicating the patterns, source text, total source lines and matches.
Save to File	Click this to save the search results file.

Chapter 7 Appendix E: Encrypt Utility

For the lookup data sources that have connection strings, you can store the connection string encrypted in the configuration file, as the connection string may have sensitive information such as a **user name** and **password**.

To encrypt the connection strings:

1. Specify the connection string.
2. Select the option **Tools > Encrypt** and enter the connection string in the Input String box.
3. Click **Encrypt**.
4. The encrypted string appears below.
5. Copy the encrypted string and paste it in place of the connection string in all the supported mapping configuration sections.



Note: As the encrypted password is associated with the local machine where the Gateway is running, other machines cannot use the configuration file directly. You will have to encrypt the password before using the configuration created in another machine. You can obtain an encrypted password using the encryption utility.

Appendix F: Upgrade from the Gateway Configuration Tool Project Settings

Available upgrades for the Gateway:

- GCT MicroStation 2D → Gateway for 2D Data / MicroStation 2D
- GCT AutoCAD 2D → Gateway for 2D Data / AutoCAD 2D
- Gateway for DEXPI 5.1.4 → Gateway for 2D Data / DEXPI

The following table lists the full upgrade information with respect to **MicroStation 2D** syntax:

GCT Project - Common Setting	New Gateway Target	Comment
logFolder path	General Settings - Log Path	Copy value
logFolder debug	General Settings - Log Level	If False -> Log Level: Warning If True -> Log Level: Verbose
environmentVariable	Extract	Imported into 'Fonts and Support Folders'
inputOutputFolders	N/A	No action
exampleData	N/A	No action
moveProcessedUnprocessedFiles	N/A	Log Information: "Setting not imported as obsolete"
objectIdFromVnetFile	N/A	Log Warning: In the <i>Gateway for 2D Data</i> , the functionality Object ID From Vnet File used in Gateway Configuration Tool's project is not supported. This feature is replaced by LookupDataSource feature in Base Mapping which can be used to map Object ID and Class ID from an external data source. In order to set it, see LookupDataSource.
restrictProcess	Project Config	Copy value
GCT Project - MS2D Settings	New Gateway Target	Comment
inputs path	Extract	Input Path = copy value, Type = File System
outputs path	Load EIWM	Output Path = copy value, Type = File System

outputs path	Load SVG	Output Path = copy value, Type = File System
outputs path	Load CSV	Output Path = copy value
mappings patterns path	see table: GCT Pattern Mapping Settings	See table: GCT Pattern Mapping Settings
mappings patterns defaultContext	Load EIWM - Project Context	Output Path = copy value, Type = File System
mappings presentation path	see table: GCT Presentation Mapping Settings	See table: GCT Presentation Mapping Settings
mappings attributes path	see table: GCT Attribute Mapping Settings	See table: GCT Attribute Mapping Settings
mappings classes path	see table: GCT Class Mapping Settings	See table: GCT Class Mapping Settings
backgroundColour	Load SVG - Background Color	Copy value
backgroundColour	Load SVG - Override Color	True
regularExpressions	DefaultBaseMapping.xml	Resolved under Pattern Mapping translation
unmappedAttributes	AttributeMapping.xml	Resolved under Attribute Mapping translation
svgScale	Rescale2dUnits Def.xml - drawingResolution Rescale2dUnits Def.xml - OLEResolution	Copy value
digitsInFloatValues	N/A	Log Warning: Setting: "Float Point Digits" is obsolete.
Use Pattern Mapping Based on Cell Extents	Patterns2d.xml	Conditions ObjectType=symbol, group, IncludeArea

Exclude blocks from pattern matching with area more than	Patterns2d.xml	Conditions, ObjectType=symbol, Size min, max
Increase Extents	Patterns2d.xml	IncludeArea
Cell	Patterns2d.xml	IncludeArea
Circle	Patterns2d.xml	IncludeArea
Closed Polyline	Patterns2d.xml	IncludeArea
Increase By Width	Patterns2d.xml	IncludeArea, right, left
Increase By Height	Patterns2d.xml	IncludeArea, top, bottom
Match Closed Objects Up To	Patterns2d.xml	Conditions ObjectType=polyline_closed, ovals, Size min, max
Multiline Text Tagging	Patterns2d.xml	Ungroup, ObjectType=mtext
clippingBox	N/A	Log Warning: Setting: Clipping Box - not imported as Segmentation is not supported.
lineWeight	Rescale2dUnits Def.xml - lineResolution type="lines"	Copy value
rscFonts	ExampleTextMa pping.xml	True -> add font map from GCTPresentationFile
zoomToExtent	N/A	If False Log Warning: Setting: "Zoom Extents Before Processing" is not upgraded. 'True' is default in Gateway for 2D Data.
v7Upgrade	N/A	If True Log Warning: Setting: "Run MicroStation Upgrade Utility" is not upgraded as not supported in Gateway for 2D Data.
viewPort	N/A	Log Warning: Setting: "viewPort" is obsolete.
hiddenLevels	Extract	Copy value
constructionelements	Extract	Copy value
removelowprioritylines	N/A	If True Log Warning: Setting: "Remove Low Priority Lines" is not upgraded as not supported in Gateway for 2D Data.
timeout	General Settings - Timeout	Output Path = copy value, Type = File System

tagCsvFile	Transform extension "CSVReporting"	See below: CSV Export Conversion Table
GCT Presentation Mapping Settings	New Gateway Target	Comment
removeLevels	DefaultBaseMapping.xml	Resolved by 'Remove' feature for objects with attribute 'layer' set in pattern
colours	ExamplePresentationMapping.xml	Color mapping
cells	DefaultBaseMapping.xml	ObjectId is set to Cell name, ClassId is set to value provided in mapping
levels	DefaultBaseMapping.xml	ObjectId is set to Level name, ClassId is set to value provided in mapping
filenamepatterns	NameMapping.xml	Resolved by adding additional Base Mapping file.
replacecharacters	ExampleTextMapping.xml	Add text mapping
explodeCells	Pattens2d.xml	Ungroup, ObjectType=symbol, numberOfObjects min, max, attribute name=block name
convertCells	N/A	Log Warning: Setting: "Convert Cells" is not upgraded as not supported in Gateway for 2D Data.
excludeCellsFromPatternMapping	DefaultBaseMapping.xml	Resolved by add attributes "#EXCLUDE_FROM_EIWM#", "#EXCLUDE_FROM_SVG#", "#EXCLUDE_FROM_CSV#" for objects with attribute 'block name' set in pattern
defaultfont	ExampleTextMapping.xml	Add map for missing fonts
fonts	ExampleTextMapping.xml	Add font mapping
GCT Pattern Mapping Settings	New Gateway Target	Comment
all mappings	BaseMapping_[original_GCT_pattern_mapping_file_name].xml	Conversion with respect both GCT and new user guides

Supported keys: from, modifier (untag, ignore), entity (block, text), illegal, block, layer, to, expand, context, ~ (reverse), qualifier, separator, modifiers, replacespace, prefix, suffix	BaseMapping_[original_GCT_p attern_mapping _file_name].xm l	Conversion with respect both GCT and new user guides
Not supported keys: prefixqualifierclassific ation, suffixqualifierclassific ation	N/A	Log Warning: feature [prefixqualifierclassification, suffixqualifierclassification] not converted.
GCT Attribute Mapping Settings	New Gateway Target	Comment
all mappings	BaseMapping_[original_GCT_at tribute_mappin g_file_name].x ml	Conversion with respect both GCT and new user guides
Supported keys: class, from, to, separator, retag, output	BaseMapping_[original_GCT_at tribute_mappin g_file_name].x ml	Conversion with respect both GCT and new user guides
GCT Class Mapping Settings	New Gateway Target	Comment
all mappings	BaseMapping_[original_GCT_cl ass_mapping_fi le_name].xml	Conversion with respect both GCT and new user guides
Supported keys: from, to, modifier (untag, ignore)	BaseMapping_[original_GCT_cl ass_mapping_fi le_name].xml	Conversion with respect both GCT and new user guides
CSV Export Conversion Table GCT Column	New Gateway Target	Comment

TAG	Transformer extension: CSVReporting attribute: 'columns'	ObjectID
REGEXON	N/A	N/A
PMEXPRESSION	Transformer extension: CSVReporting xml node: 'column'	CSVReporting add node 'column' #Tracking: AttributeName BMExpression #Filter: AttributeName: ObjectID " BaseMappings annotation level: "ObjectIDTracking"
CLASS	Transformer extension: CSVReporting attribute: 'columns'	ClassID
ALIAS	Transformer extension: CSVReporting attribute: 'columns'	#Association:is identified by #TargetAttribute:ObjectID
[any characteristic]	Transformer extension: CSVReporting attribute: 'columns'	Copy value

The following table lists the full upgrade information with respect to **AutoCAD 2D** syntax:

GCT Project - Common Setting	New Gateway Target	Comment
logFolder path	General Settings - Log Path	copy value
logFolder debug	General Settings - Log Level	False -> Log Level: Warning, True -> Log Level: Verbose
environmentVariable	Extract	Imported into 'Fonts and Support Folders'
<inputOutputFolders create="true" />	N/A	no action
<exampleData install="true" />	N/A	no action

<moveProcessedUnprocessedFiles apply="false" />	N/A	Information
objectIdFromVnetFile	N/A	Warning
generateCategorizedLog	General Settings - Categorized Logs	copy value
GCT Project - AC2D Settings	New Gateway Target	Comment
inputs path	Extract	Input Path = copy value, Type = File System
outputs path	Load EIWM	Output Path = copy value, Type = File System
outputs path	Load SVG	Output Path = copy value, Type = File System
outputs path	Load CSV	Output Path = copy value
mappings patterns path	see table: GCT Pattern Mapping Settings	see table: GCT Pattern Mapping Settings
mappings patterns defaultContext	Load EIWM - Project Context	Output Path = copy value, Type = File System
mappings presentation path	see table: GCT Presentation Mapping Settings	see table: GCT Presentation Mapping Settings
mappings attributes path	see table: GCT Attribute Mapping Settings	see table: GCT Attribute Mapping Settings
mappings classes path	see table: GCT Class Mapping Settings	see table: GCT Class Mapping Settings
backgroundColour	Load SVG - Background Color	copy value
backgroundColour	Load SVG - Override Color	true
regularExpressions	DefaultBaseMapping.xml	resolved under Pattern Mapping translation
unmappedAttributes	AttributeMapping.xml	resolved under Attribute Mapping translation

svgScale	Rescale2dUnitsDef.xml - drawingResolution	copy value
visualStyle2dWireFrame	Extract - Visual Styles - Style	False -> mixed, True -> Wireframe
digitsInFloatValues	N/A	Warning
Change Extents	Patterns2d.xml	IncludeArea
Move Nested Blocks To Root	Patterns2d.xml	ungroup, ObjectType=symbol_nested, keepSingle=true
Explode Blocks With Text Only	Patterns2d.xml	ungroup ObjectType=symbol_nested_textonly, symbol_single_textonly, keepSingle=false
Use Pattern Mapping Based on Block Extents	Patterns2d.xml	conditions ObjectType=symbol, group, IncludeArea
Exclude blocks from pattern matching with area more than	Patterns2d.xml	conditions, ObjectType=symbol, Size min, max
Increase Extents	Patterns2d.xml	IncludeArea
Block	Patterns2d.xml	IncludeArea
Circle	Patterns2d.xml	IncludeArea
Closed Polyline	Patterns2d.xml	IncludeArea
Increase By Width	Patterns2d.xml	IncludeArea, right, left
Increase By Height	Patterns2d.xml	IncludeArea, top, bottom
Match Closed Objects Up To	Patterns2d.xml	conditions ObjectType=polyline_closed, ovals, Size min, max
Multiline Text Tagging	Patterns2d.xml	ungroup, ObjectType=mtext
clippingBox	N/A	Log Warning: Setting: Clipping Box - not imported as Segmentation is not supported.
lineWeight	Rescale2dUnitsDef.xml - lineResolution type="lines"	copy value
shxFonts	ExampleTextMapping.xml	True -> add font map from GCTPresentationFile
strokeWidth (Increase Tagged Text Thickness)	N/A	Warning

changeView	Extract - Layouts Selection - Filter	True Model Space -> #Model#, True Paper Space -> #1stpaper#
processAlternativeViewIfEmpty	Extract - Layouts Selection - Filter	True -> #1stpaper#
hiddenLayers	N/A	True -> Warning
graphicsMode	N/A	Warning
timeout	General Settings - Timeout	Output Path = copy value, Type = File System
tagCsvFile - create	Load CSV	True -> Type = File System, True -> Type = None
exportalllayouts	Extract - Layouts Selection - Filter	True -> #allpaper#
GCT Presentation Mapping Settings	New Gateway Target	Comment
removeLayers	DefaultBaseMapping. xml	Resolved by 'Remove' feature for objects with attribute 'layer' set in pattern
colours	ExamplePresentation Mapping.xml	Color mapping
layers	DefaultBaseMapping. xml	ObjectId is set to Layer name, ClassId is set to value provided in mapping
blocks	DefaultBaseMapping. xml	ObjectId is set to Block name, ClassId is set to value provided in mapping
blocks + extractallblockreferenc etags (project setting)	DefaultBaseMapping. xml	ObjectId is set to Block name with AutoNumber, ClassId is set to value provided in mapping
filenamepatterns	NameMapping.xml	Resolved by adding additional Base Mapping file.
removeBlocks	DefaultBaseMapping. xml	Resolved by 'Remove' feature for objects with attribute 'block name' set in pattern
explodeBlocks	Pattens2d.xml	ungroup, ObjectType=symbol, numberOfObjects min, max, attribute name=block name
convertBlocks	Extract - Blocks - Block	Replace block with image
excludeBlocksFromPatternMapping	DefaultBaseMapping. xml	Resolved by add attributes "#EXCLUDE_FROM_EIWM#", "#EXCLUDE_FROM_SVG#",

		"#EXCLUDE_FROM_CSV#" for objects with attribute 'block name' set in pattern
defaultfont	ExampleTextMapping.xml	add map for missing fonts
fonts	ExampleTextMapping.xml	add font mapping
purge purgeall	N/A	
GCT Pattern Mapping Settings	New Gateway Target	Comment
all mappings	BaseMapping_[original_GCT_pattern_mapping_file_name].xml	conversion with respect both GCT and new user guides
Supported keys: from, modifier (untag, ignore), entity (block, text), illegal, block, layer, to, expand, context, ~ (reverse), qualifier, separator, modifiers, replacespace, prefix, suffix	BaseMapping_[original_GCT_pattern_mapping_file_name].xml	conversion with respect both GCT and new user guides
Not supported keys: prefixqualifierclassification, suffixqualifierclassification	N/A	Log Warning: feature [prefixqualifierclassification, suffixqualifierclassification] not converted.
GCT Attribute Mapping Settings	New Gateway Target	Comment
all mappings	BaseMapping_[original_GCT_attribute_mapping_file_name].xml	conversion with respect both GCT and new user guides
Supported keys: class, from, to, separator, retag, output	BaseMapping_[original_GCT_attribute_mapping_file_name].xml	conversion with respect both GCT and new user guides
GCT Class Mapping Settings	New Gateway Target	Comment

all mappings	BaseMapping_[original_GCT_class_mapping_file_name].xml	conversion with respect both GCT and new user guides
Supported keys: from, to, modifier (untag, ignore)	BaseMapping_[original_GCT_class_mapping_file_name].xml	conversion with respect both GCT and new user guides
CSV Export Conversion Table GCT Column	New Gateway Target	Comment
TAG	Transformer extension: CSVReporting attribute: 'columns'	ObjectID
REGEXON	N/A	N/A
PMEXPRESSION	Transformer extension: CSVReporting xml node: 'column'	CSVReporting add node 'column' #Tracking: AttributeName BMExpression #Filter: AttributeName: ObjectID " BaseMappings annotation level: "ObjectIDTracking"
CLASS	Transformer extension: CSVReporting attribute: 'columns'	ClassID
ALIAS	Transformer extension: CSVReporting attribute: 'columns'	#Association:is identified by #TargetAttribute:ObjectID
[any characteristic]	Transformer extension: CSVReporting attribute: 'columns'	copy value

The following table lists the AVEVA NET Gateway Configuration Tool project settings (for AutoCAD 2D), which are upgraded in the new AVEVA™ Gateway for 2D Data settings:

AVEVA Gateway Configuration Tool General Settings	AVEVA Gateway for 2D Data Target	Comment
inputs path	Extractor Config	
outputs path	SVG, EIWM, CSV Loaders Configs	

type	SVG Loader Config	Setting to set output type SVG
defaultContext	EIWM Loader Config	
backgroundColour	SVG Loader Config	
processXrefs	Base Mapping	Resolved by 'Remove' feature for objects with attribute 'saved path' (specific for Xrefs)
tagCsvFile	CSV Loader Config	See 'CSV Export Conversion Table '
logFolder	Project Config	
debug	Project Config	Resolved by setting Log level to ' Verbose '
environmentVariable	Extractor Config	Imported into ' Fonts and Support Folders
moveProcessedUnprocessedFiles	N/A	Log Information: " Setting not imported as obsolete "
processed path	N/A	
unprocessed path	N/A	
restrictProcess	N/A	Log Information: Setting not imported as obsolete
objectIdFromVnetFile	N/A	Log Warning: In the Gateway for 2D Data, the Object ID From Vnet File functionality used in Gateway Configuration Tool's project is not supported. This feature is replaced by LookupDataSource feature in Base Mapping which can be used to map Object ID and Class ID from an external data source. In order to set it, see LookupDataSource .
inputOutputFolders	N/A	
exampleData	N/A	
lineWeight	N/A	Log Warning: All 2D entities will be exported with default line thickness (in case user selects value different than 1).

The [filenamepatterns](#) setting in the *AVEVA Gateway Configuration Tool* is replaced by the Base Mapping in the new *AVEVA Gateway for 2D Data*, with the following configurations in the [NameMapping.xml](#). This .xml modifies the filename whose value is stored in the `#MODEL_NAME#` attribute:

```
<MappingTemplate componentVersion="2.10.0.0" sourceProductName="AVEVA Gateway for 2D Data"
componentName="BaseMapping">
  <ObjectMappings>
    <!--Mapping imported from Gateway Configuration Tool. Setting: File Name Mapping-->
    <Object>
      <Conditions>
        <Attribute name="#TYPE#" pattern="^manifest$" />
        <Attribute name="#MODEL_NAME#" pattern="^\d{6}_[A-Za-z]+$" />
      </Conditions>
    </Object>
  </ObjectMappings>
</MappingTemplate>
```

```

</Conditions>
<ObjectID value="#MODEL_NAME#">
  <Transforms>
    <Remove pattern="_[A-Za-z]+" />
    <InsertAfter pattern="\d{6}" value="test" />
  </Transforms>
</ObjectID>
<Attributes>
  <Attribute>
    <Conditions>
      <Attribute name="#MODEL_NAME#" pattern="^\d{6}_[A-Za-z]+$" />
    </Conditions>
    <Name value="#MODEL_NAME#" />
    <Value value="#MODEL_NAME#">
      <Transforms>
        <Remove pattern="_[A-Za-z]+" />
        <InsertAfter pattern="\d{6}" value="test" />
      </Transforms>
    </Value>
  </Attribute>
</Attributes>
</Object>
<!--Mapping imported from Gateway Configuration Tool. Setting: File Name Mapping-->
<Object>
  <Conditions>
    <Attribute name="#MODEL_NAME#" pattern="^\d{6}_[A-Za-z]+$" />
  </Conditions>
  <Attributes>
    <Attribute>
      <Conditions>
        <Attribute name="#MODEL_NAME#" pattern="^\d{6}_[A-Za-z]+$" />
      </Conditions>
      <Name value="#MODEL_NAME#" />
      <Value value="#MODEL_NAME#">
        <Transforms>
          <Remove pattern="_[A-Za-z]+" />
          <InsertAfter pattern="\d{6}" value="test" />
        </Transforms>
      </Value>
    </Attribute>
  </Attributes>
</Object>
</ObjectMappings>
</MappingTemplate>

```

Chapter 8 Appendix G: Reserved Attributes Used in the Gateway

The following reserved attributes are used by the Gateway to apply and implement mapping. They should therefore not be present in the extracted source data, unless they are being used for this purpose:

- **ObjectId**: It is used to map Id to the EIWM object. During the transformation phase, the Gateway creates "ObjectId" attribute or overwrites it, if already present.
- **ObjectName**: It is used to map Name to the EIWM object. During the transformation phase, the Gateway creates "ObjectName" attribute or overwrites it, if already present.
- **ClassId**: It is used to map Class Id to the EIWM object. During the transformation phase, the Gateway creates "ClassId" attribute or overwrites it, if already present.
- **ContextId**: It is used to map Context to the EIWM object. During transformation phase, the Gateway creates "ContextId" attribute or overwrites it, if already present.
- **Revision**: It is used to map revision to the EIWM object. During the transformation phase, the Gateway creates "Revision" attribute or overwrites it, if already present.
- **TemplateId**: It is used to map Template Id in which the EIWM object is present. During the transformation phase, the Gateway creates "TemplateId" attribute or overwrites it, if already present.
- **IncidentalClassification**: It is used to map IncidentalClassification association to EIWM object. During the transformation phase, the Gateway creates "IncidentalClassification" association or overwrites it, if already present.

If any of these attributes are present in the source data, then appropriate mapping should be configured to ensure that they do not have unintended effects. For example, if the source data has a "Revision" attribute, then mapping can be used to rename this attribute, for example, "Source_Revision". Otherwise, the EIWM loader interprets this to be the Revision value for the relevant object.

Reporting for Reserved Attributes in Extracted Data

You can generate a CSV report of the extracted data to check if there are any reserved attributes present in the extracted data.

This CSV report should be generated before any other transformation processing. You can insert the following transformation extension node before all other extensions, in the transformation configuration file.

```
<extension name="CSVReporting" >
<csvReport suffix="_BeforeTransformation" columns="ObjectId, ObjectName, ClassId,
ContextId, Revision, TemplateId, IncidentalClassification" addHeaderRow="true" />
</extension>
```

This will generate a .csv file containing information about engineering objects that may contain these attributes. If there are any objects that have a value associated to any of the above attributes, it might cause unexpected results.

Chapter 9 Appendix H: Handle System Attributes

The Gateway creates "System Attributes" that are used to store metadata about the input source and control the behavior of the EIWM Loader. Transformation prevents modification of many of these system attributes. However, transformation can also use these values in filters or update other attributes with their values.

During processing, the Gateway uses these system attributes, as listed in the following table:

Attribute	Properties	Usage		
		Base Mapping	Presentation Mapping	EIWM Loader
#TYPE#	manifest	R	R	-
#MANIFEST#	manifest	R	R	EXPORT
#DATE_TIME#	manifest	R	R	EXPORT
#INPUT_LOCATION#	manifest	R	R	EXPORT
#INPUT_FOLDER#	manifest	R	R	EXPORT
#GRAPHICS_FORMAT#	manifest	R	R	-
#UNITS#	manifest	R	R	EXPORT
#DEFINED_SEGMENTS#	manifest	R	R/W	-
#KEEP_FOLDER_TREE#	manifest	R	R	-
#TARGET_LOCATION#	manifest	R	R	EXPORT
#FILE_TIME_STAMP#	manifest	R	R	-
#UNITS-BORE#	manifest	R	R	EXPORT
#UNITS-CO-ORDS#	manifest	R	R	EXPORT
#UNITS-BOLT-LENGTH#	manifest	R	R	EXPORT
#UNITS-BOLT-DIA#	manifest	R	R	EXPORT
#UNITS-WEIGHT#	manifest	R	R	EXPORT
InfoLocator	manifest	R	R	EXPORT

Attribute	Properties	Usage		
#material_ambient_colour_RGB	Engineering Object	R	R/W	EXPORT
#SEGMENT_NAME#	Manifest and Engineering Object	R	R/W	-
keepUnmappedAssociations	Engineering Object	R/W	R	USED FOR FILTERING
keepUnmappedAttributes	Engineering Object	R/W	R	USED FOR FILTERING

Notes:

- R – The attribute is read only. It is not possible to modify its value.
- R/W – The attribute can be modified in transformer configuration.

Example – Copying of System Attribute to Normal Attribute:

In this example, the attribute #GRAPHICS_FORMAT# is not exportable to EIWM. To include the attribute's value in EIWM, you can create or re-use a non-system attribute and set its value to the value of the system attribute.

```
<Attribute>
  <Conditions>
    <Attribute name="#GRAPHICS_FORMAT#" pattern="^.*$"/>
  </Conditions>
  <Name value="GRAPHICS_FORMAT" />
  <Value value="[#GRAPHICS_FORMAT#]"/>
</Attribute>
```

In this example, the engineering attribute named GRAPHICS_FORMAT is added.

Appendix I: Third-Party Software Dependency

Where third-party software is used in this product release, any notices or terms and conditions that such third-party requires be reproduced in the release are reproduced herein.

Appendix J: Limitations of the Gateway

Supporting Non-English User Locales

When the Gateway is installed on a system where the user has selected a non-English locale that uses a comma as the decimal separator (that is, a number 123.45 in a Windows system with a French locale would be expressed as 123,45), the Gateway does not interpret such numbering formats from input sources and configurations nor does it export them into output files.