AVEVA™ Gateway for 2D Data

6.0

aveva.com

Publication date: Monday, April 14, 2025

Publication ID: 1526310

Contact information

AVEVA Group Limited
High Cross
Madingley 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 2D Data?

These release notes explain the feature updates, new enhancements and known issues of *AVEVA Gateway for 2D Data*. They are usually released as part of a major product release.

This section describes the release rotes for the *AVEVA Gateway for 2D Data*.

Gateway for 2D Data 6.0

A full release of *AVEVA Gateway for 2D Data 6.0.0*, which replaces the full releases of:

- *AVEVA Gateway for AutoCAD 2D 5.1.5*
- *AVEVA Gateway for MicroStation 2D* in the *Gateway Configuration Tool 5.0.11* and *5.0.11.6*
- *AVEVA Gateway for DEXPI 5.1.4*

The products or product components installable from or upgraded by this release are as follows:

Product Code	**Product Name**	**Version Number**
FLEX-AIM-107	*AVEVA Gateway for 2D Data 6.0.0*	6.0.0

Introduction

This Release Note about *AVEVA Gateway for 2D Data 6.0.0* 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)

Prerequisites	**Version No.**
	Microsoft Windows Server 2022 (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, 2022 or 2025 (64-bit), except when using the Smart P&ID extractor which is supported only on the Windows systems supported by Hexagon's SmartSketch 2018, 11 and 11.1
- **.NET Framework 4.8**
- Microsoft Visual C++ Redistributable 2015-2019 (64-bit)
- Microsoft.ACE.OLEDB.12.0 (if lookups are being made to Excel files)

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

Works (is compatible) with:

A full release of *AVEVA Gateway for 2D Data 6.0.0* works (is compatible) with:

- *AVEVA Asset Information Management* (*AIM*) (previously known as AVEVA NET)
  - On-premise version 5.1.11
- Input 2D files in the following formats:
  - Autodesk AutoCAD 2D .DWG and .DXF files up to version 2018 (version AC1032), produced by all versions of AutoCAD up to AutoCAD 2024
  - Bentley MicroStation .DGN files V8 XM (aka V8.9) and V8i (aka V8.11)
    - Most V7 files can also be processed, but if not then they should be opened in MicroStation and saved to a V8 copy
  - DEXPI .xml files conforming to schemas 3.3.3 or 4.0.1, including DEXPI files exported from *AVEVA P&ID 12.2.2.4*
  - Hexagon's Smart P&ID .IGR and .PID files supported by Hexagon's SmartSketch Versions 2018, 11 and 11.1
- Access Database Engine 2010 or 2016 (if lookups to Excel files)
- Microsoft SQL Server (64-bit) 2019 (if lookups to a SQL Server database)
- Oracle database 12c Release 3 (12.1.0.2.0) (if lookups to an Oracle database)

Supersedes

This release supersedes the previous full releases:

- *AVEVA Gateway for AutoCAD 2D 5.1.5* (release 71447)

- *Gateway for MicroStation 2D* in the *AVEVA NET Gateway Configuration Tool 5.0.11* (release 71332)
- *Gateway for MicroStation 2D* in the *AVEVA NET Gateway Configuration Tool 5.0.11.6* (release 71455)
- *AVEVA Gateway for DEXPI 5.1.4* (release 71396)

To Install Product Release

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

Note: Before this *AVEVA Gateway for 2D Data 6.0.0* is installed, please ensure that any previous version of *AVEVA Gateway for 2D Data* is uninstalled.

First released on:

This product release is first available on AVEVA's Knowledge and Support web site as '*AVEVA Gateway for 2D Data 6.0.0*'.

List of Enhancements

This release includes the following functional enhancements with respect to the previous version of the Gateways:

- *AVEVA Gateway for AutoCAD 2D 5.1.5*
- *AVEVA Gateway for MicroStation 2D* in the Gateway Configuration Tool 5.0.11 and 5.0.11.6
- *AVEVA Gateway for DEXPI 5.1.4*

Incident Number	**IMS Number**	**Story Number**	**Description**
		946897	All: Improved memory usage when processing multiple files
		1660009	All: Option for CSV Reporter to append results to an existing CSV file
		1834069	All: Improve speed of Base Mapping processing with 100's of patterns
		2336324	All: Timestamps should not be included in output files
		2372869	All: CSV annotations and reporting of matched patterns
		2372865	All: Text Mapping
		2626025	All: Optimize size of Gateway for 2D Data
960147917	1734650	1735388	MS2D: line tag represented in several boxes not recognized
960170431	1888972	1942787	MS2D: Gateway for MS 2D errors
960218067	2105064	2126134	MS2D: Reg AVEVA NET GCT Error - Excel Embedded
960238391	2181020	2186140	MS2D: Title block information not exported as characteristics

Incident Number	IMS Number	Story Number	Description
		2201022	MS2D: Embedded excel support
		2397144	MS2D: Support for Raster/Embedded images
		2201064	MS2D: Support Wipe out graphics
		2373027	MS2D: Remove construction elements
		2489939	MS2D: Add Extents to Text objects in Extractor
		2201008	MS2D: Enable Json logging for MS2D Gateway
		2201069	MS2D: Fill Pattern mapping in extractor for Hatches
		2491516	MS2D: Support fonts: RSC, SHX, TTF
		2507710	MS2D: Support Notes
		2511088	MS2D: Group tags with associated graphics
		2507144	MS2D: Text Node should be extracted as one object
		2489805	MS2D: Support Complex Shapes
		2199032	MS2D: Pattern 2D mapping
		2373030	MS2D: Remove hidden layers in extractor
		1932057	MS2D: Font Mapping
		2200795	MS2D: Handling reference cell and Xref
		2336325	MS2D: Support for "Complex Chains"
		2352589	MS2D: Support automatic upgrade from GCT configs to new configs
		2352602	MS2D: support Patterns2D to avoid need for explodeCells
		2201050	MS2D: Option for the zoom to extents feature
		2509823	MS2D: Upgrade configurations from GCT Gateway for MicroStation
		1968116	AC2D: Filter for views/layouts to Patterns 2D extension
		615485	AC2D: AVEVA Cloud Storage support
		1804105	AC2D: Add CLIPs to viewports
		1834130	AC2D: Support extents for some text objects under proxy objects

Incident Number	IMS Number	Story Number	Description
		2510552	AC2D: Upgrade configurations from *Gateway for AutoCAD 2D 5.1.5*
		2510552	AC2D: Upgrade configurations from GCT *Gateway for AutoCAD 2D*
		2510536	DEXPI: Upgrade configurations from *Gateway for DEXPI*

List of Fault Corrections

This release addresses defects arising from the following support incidents in the previous version of the *AVEVA Gateway for AutoCAD 2D 5.1.5*:

Incident number	IMS number	Defect number	Description
960278045	2403571	2415865	AC2D: SVG too small due to proxy objects with large X,Y positions
960283091	2419394	2429200	AC2D: Gateway error when processing selected dwg files
960289568	2444339	2444473	AC2D: Upgrading GCT projects does not work
960283004	2453346	2491841	AC2D: Getting gateway error after reprocessing
960283004	2453347	2491897	AC2D: Getting gateway error after reprocessing (Failed in Extractor)
960308844	2533031	2546430	AC2D: File Not Found-Gateway Error (Failed in Transformer)
960317800	2561145	2561378	AC2D: Migration issue from GCT to Gateway for AutoCAD
960304981	2507416	2564044	AC2D: Multiple instances of same object ID lost using Search Criteria
960318384	2566573	2566611	AC2D: How do I remove layers from the svg output file?
960315966	2567959	2573158	AC2D: Data Processing Errors for AVEVA AIM-A
960310079	2542831	2586328	AC2D: Gateway & Data import error
960282604	2525774	2612871	AC2D: Data Import Error for CAD files
	2419394	2429200	AC2D: Load Reporting not updating status of files reloaded successfull

Product Quality Statement

The development, maintenance and testing of AVEVA Group's software products is 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

Limited independent system testing has been carried out on this product, as released, to verify that it installs correctly in all supported configurations, that any new functionalities have been correctly implemented and identified defects have been corrected.

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 on AVEVA's Knowledge and Support web site .

Exceptions - Significant Defects and Recommended Workarounds

When the input DWG files are stored in an S3 bucket support for multiple layouts is limited to only the "Active" layout. If other layouts need to be extracted then the DWG files should first be copied to a server folder and then processed by the Gateway.

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

Known Issues

None.

Chapter 2

Gateway for 2D Data 6.0

The Gateway extracts information from ‘from AutoCAD 2D .DWG and .DXF files, MicroStation 2D .DGN files and DEXPI format .XML files and transforms them into output EIWM and .SVG file formats to be loaded into *AIM*. The Gateway is a stand-alone application that can be deployed independently or as part of the AIM pipeline accessed via the Gateway Data Publisher. The Gateway can be run either from its graphical user interface or from a command line interface. The output data can also be stored on a Windows file system, on an Amazon Web Services (AWS) S3 bucket, or on AVEVA Cloud Storage.

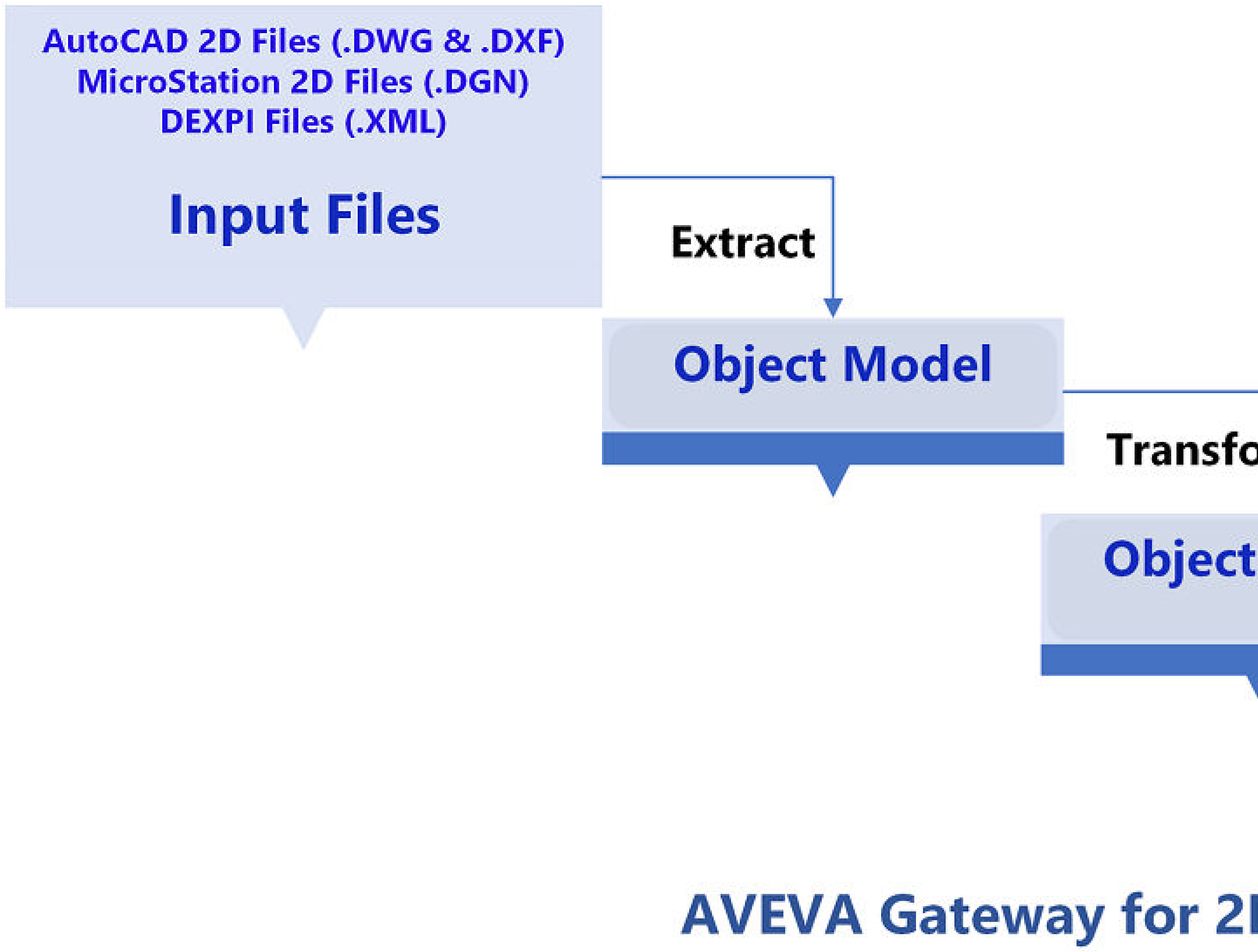
Characteristics of the Gateway

The following are the characteristics of the Gateway:

- **Common User Experience**: All new gateways built with common components ensures consistent configurations and operations.
- **Extract data from AutoCAD, MicroStation or DEXPI files,** where separate projects for each type of extraction are required due to differences in Transformation operations.
- **Transformation of Data**: Supports transformation of extracted data from input files through XML-based mappings. Results of each transformation can be exported to a .CSV file for inspection.
- **Load the transformed data** into output files suitable for AIM:
    - Engineering objects, attributes and their associations into EIWM files.
    - 2D renditions of the drawing graphics and object (tag) hotspots in .SVG files.
- **Summary Report**: Provides a detailed log and short summary report describing general information about processed data.
- **Standalone**: As it is a standalone application, deploy it independently either from its graphical user interface or from a command line interface.

Note: You need not install AutoCAD or MicroStation for the Gateway to function. However, the Gateway may be configured to use the AutoCAD and MicroStation fonts and support files, which are available in the settings of a project.

Like all of the new Gateways, it consists of an Extract, Transform and Load architecture:


AutoCAD 2D Files (.DWG & .DXF)
MicroStation 2D Files (.DGN)
DEXPI Files (.XML)
Input Files
Extract
Object Model
Transfo
Object
AVEVA Gateway for 2I

Gateways Copyright Information

Chapter 1 Get Started

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

Hardware Requirements

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

Component	**Minimum**	**Recommended**
Memory	8GB	32GB
Hard Disk Drive	80GB	
CPU	Intel® Xeon® processor, 3GHz, Dual core	Intel® Xeon® processor, 3GHz, 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 16GB of RAM, then the Gateway process consumes less than 12.8GB 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%.

Software Requirements

The Gateway server must meet the following software requirements before you install the Gateway:

Application	**Supported Version**
Microsoft .NET Framework	4.8
Windows Server	2016 (64-bit), 2019 (64-bit), 2022 (64-bit), 2025 (64-bit)*
Visual C++ Redistributable Packages	Microsoft Visual C++ Redistributable 2015-2022 (64 bit)
Access Database Engine (ADE)	2010 (64-bit), 2016 (64-bit) **Notes**: • When you install ADE, this installs required default drivers for ODBC. • ADE is only needed for using lookups in mapping.

Notes:

- The Gateway supports *AVEVA Asset Information Management (AIM)* version 5.1.11.
- AVEVA Licensing System (ALS) and Licensing as a Service (LaaS) are no longer needed.

- * Except when using the Smart P&ID extractor which is supported only on the Windows systems supported by Hexagon's SmartSketch 2018, 11 and 11.1.
- The Gateway supports:
  - Autodesk AutoCAD 2D .DWG and .DXF files up to version 2018 (version AC1032), as produced by all versions of AutoCAD up to AutoCAD 2024.
  - Bentley MicroStation 2D V8 XM (aka V8.9),V8i (aka V8.11) files (some V7 .dgn files that can't be read should be opened in MicroStation and saved as V8 files).
  - DEXPI format `.XML` files conforming to schema 3.3.3 or 4.0.1.
  - Hexagon's Smart P&ID .IGR and .PID files supported by Hexagon SmartSketch Versions 2018, 11 and 11.1.

Lookups requiring Access Database Engine

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

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

http://www.microsoft.com/downloads/en/details.aspx?FamilyID=C06B8369-60DD-4B64-A44B-84B371EDE16D&displaylang=en

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 that is loaded into *AIM*, therefore, it needs to read, write and modify both files and folders.

The following are the security best practice recommendations:

- Use the **principle of least privilege**:
  - Grant the user account that is used to run the Gateway read-only access. Grant write or update access only to the specific files and folders it needs to modify. For example, within a project folder, grant read only access to Input folder, write access to Log folder and modify access to Output folder, Configuration/ Mappings folder and Project file.
  - 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, restrict the **IAMrole** to access only S3 bucket. Define a bucket policy to restrict access to an S3 bucket.
- 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.

- 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, 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.

If **AVEVA Cloud Storage** (ACS) is used for either input or output file locations, then ensure that the **Transport Layer Security** (TLS 1.2) option is selected. Perform the following recommended steps:

1. Navigate to **Internet Options** and then the **Advanced** tab.
2. Under the **Security** section, select the option "**Use TLS 1.2**".
3. Click **Apply** and **OK**.

If the Gateway's input source contains personal information such as an e-mail address or home address, then this may be passed through into the output EIWM, therefore ensure that it is either filtered out via transformation mappings or that the location and management of the EIWM output file complies with local legislation related to protecting personal information, such as GDPR in EU countries.

Note: 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.

Install the Gateway

Before installing a new version of the Gateway, remove any previously installed versions of the Gateway from your system. You must have Administrator privileges to install the Gateway.

To install the Gateway:

1. From your installation media, double-click the `2DDataGatewayInstaller.msi` file.
2. Click **Next**.
3. Select **I accept**, and then click **Next**.

   Note: To print a copy of the License Agreement, click **Print**.
4. If you want to modify the default installation **Location**, click **Browse**.

   Notes:

   - To find out the available and required disk space for your chosen features and available drives, click **Disk Usage**.
   - If you select to install to a custom directory the installer sets read access only to all **Authenticated Users** to ensure the installed contents can't be changed. If you are an administrator user, you must be careful not to change the installation directory content.
5. Select **Everyone** or **Just me** depending on who you want to be able to access the application.
6. Click **Next**.
7. Click **Install**. The installation process starts.
8. Click **Finish** after the **Setup** wizard completes installing the **Gateway**.

Command Line Installation

To install the **Gateway** from the command line:

1. Type **2DDataGatewayInstaller.MSI /?** to see all options.

Example of quiet installation with passing folder:

```
2DDataGatewayInstaller.MSI INSTALLFOLDER="C:\Program Files\MyFolder" /quiet
```

Uninstall the Gateway

It is possible to uninstall the Gateway by using either of the following two ways:

- Control Panel
- 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 Gateway for 2D Data from the list of programs.
3. Select **Uninstall** to remove the Gateway.

To remove the Gateway using the Gateway installer:

1. From your installation media, double-click the `2DDataGatewayInstaller.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.

Change and Repair the Gateway

It is possible to change or repair the Gateway using the installer for the Gateway.

To change or repair the Gateway using the Gateway installer:

1. From your installation media, double-click the `2DDataGatewayInstaller.msi` file.
2. Click **Next**.
3. Select one of the following:
    a. Click **Change** if you want to modify any optional features installed with the application.
    b. Click **Repair** if you want to repair the errors in the most recent installation, repair any corrupt files, or fix any missing shortcuts or registry entries.
4. 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. It is possible to 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 **Gateway for 2D Data** in the Search box on the taskbar in your computer. Alternatively, click the **AVEVA Gateway for 2D Data** icon in the installed location or the shortcut icon on your desktop, if that option was selected when installing the Gateway.
2. **AVEVA Gateway for 2D Data** 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.

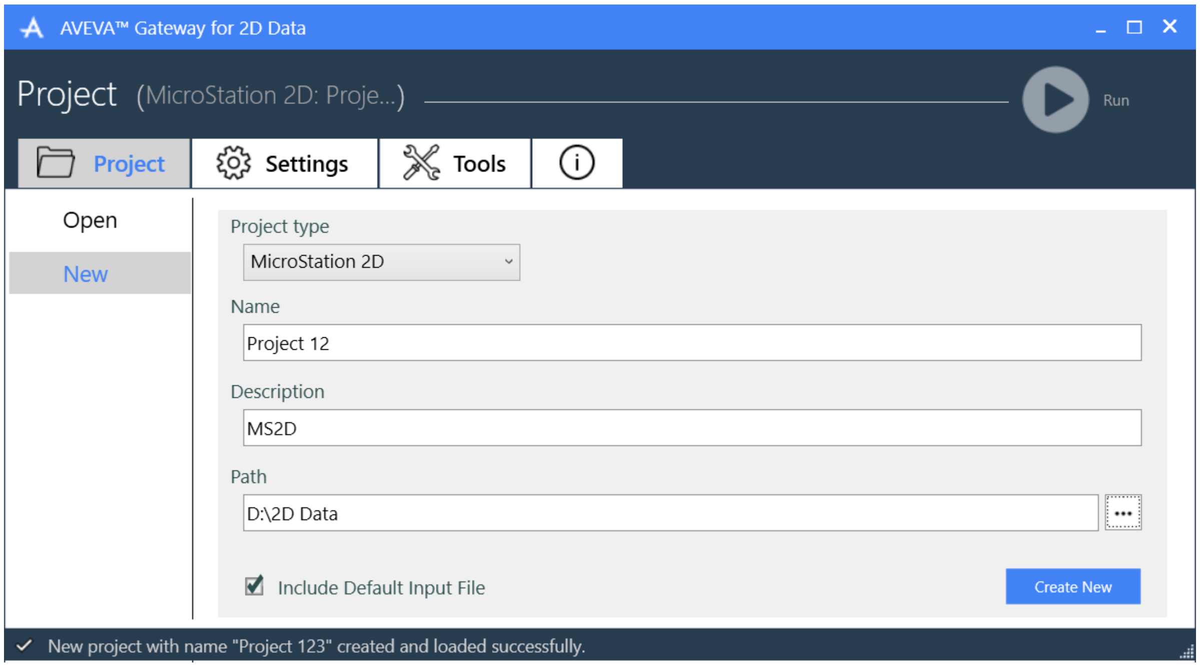
Note: The Gateway supports Universal Naming Convention (UNC) paths. By using UNC paths, connect to servers and other workstations without mapping to a drive. The UNC path syntax is as follows:`\\servername\sharename\directory`

It is possible to 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**.
2. Select the appropriate project type from the **Project type** dropdown box.


AVEVA™ Gateway for 2D Data
Project (MicroStation 2D: Proje...)
Run
Project
Settings
Tools
Open
New
Create New

Project type: MicroStation 2D

Name: Project 12

Description: MS2D

Path: D:\2D Data

☑ Include Default Input File

New project with name "Project 123" created and loaded successfully.

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

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

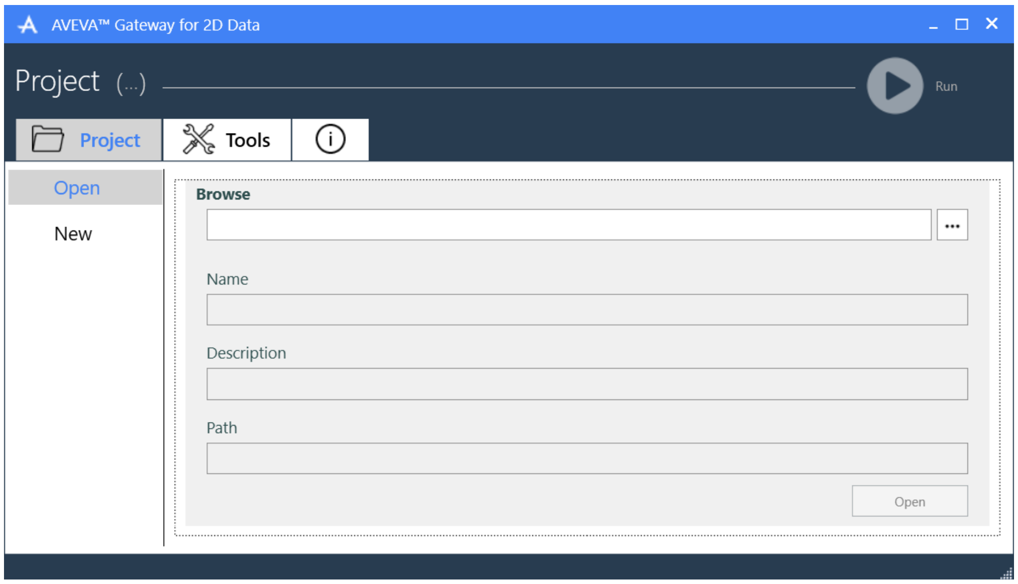
   If the folder does not exist, a new project folder is created containing the Project configuration file with the same Project Name. When a project is created, a project xml file is created with default setting. When the project is loaded successfully, a message is displayed in the status bar of the window. This message indicates that the project configuration files are loaded into the user interface.
6. Select the **Include Default Input File** box to create some example input files.

Open Projects

It is possible to 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, 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. 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 panels:

- **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 menu 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.

Note: A yellow triangle symbol near any of the panels indicates that some fields in that panel have unsaved changes or have invalid data, so you should open the panel, fix the invalid data, and save the settings within that panel.

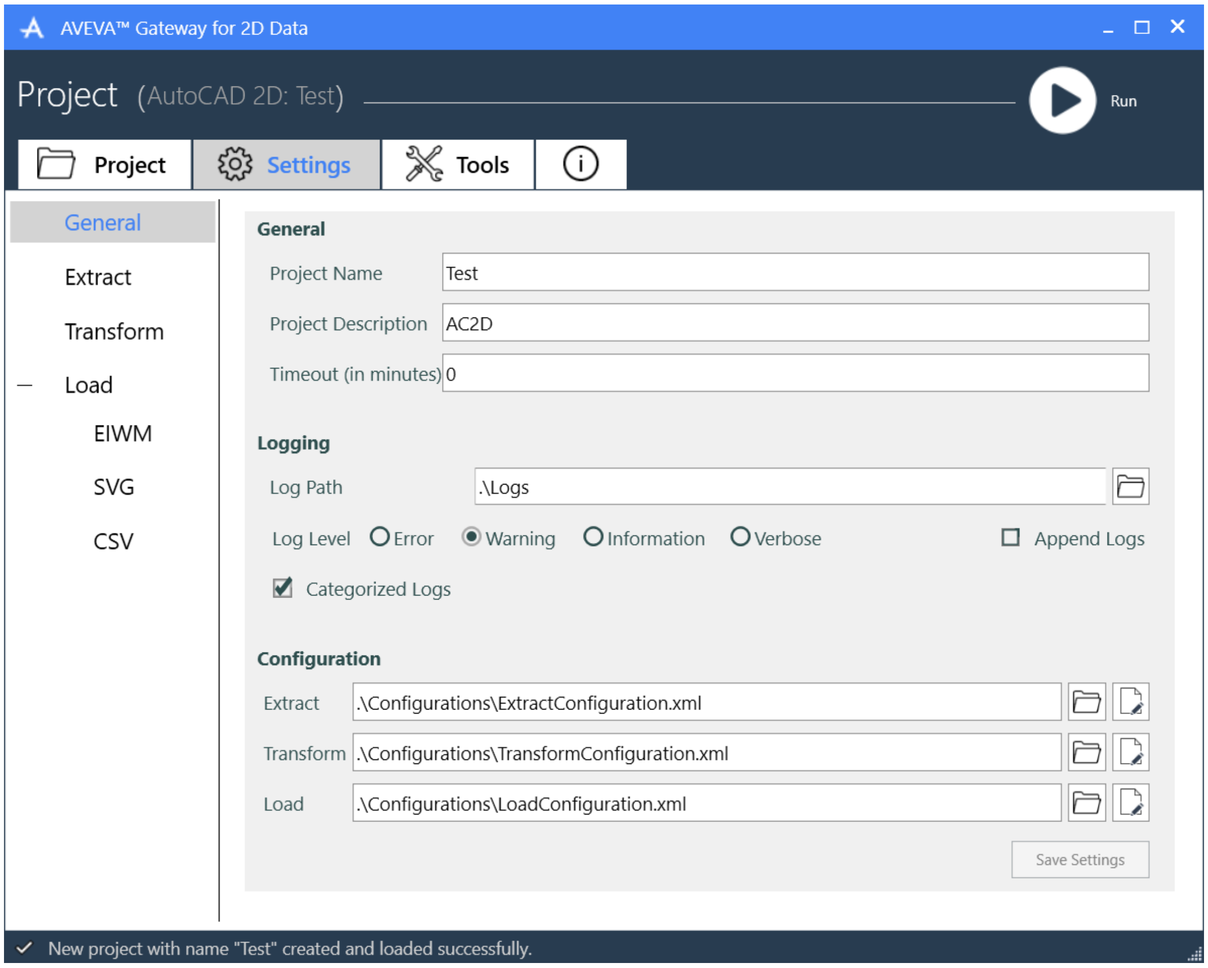
Using Relative Paths in the Configurations

Use of relative paths for file locations makes it easier to re-use and share Gateway projects. Specify all paths in the project's configurations in two ways by setting global paths or relative paths:

Example of Global paths in the Project file:

```
  <transformerConfigLocator
path="C:\TestProject\Configurations\TransformConfiguration.xml" />
  <extractorConfigLocator path="C:\ TestProject
\Configurations\ExtractConfiguration.xml" />
  <loaderConfigLocator path="C:\ TestProject \Configurations\LoadConfiguration.xml" />
  <log level="Warning" folderPath="C:\ TestProject\Logs" appendToLog="false" />
```

Example of Relative paths in the GUI:



General

Project Name: Test

Project Description: AC2D

Timeout (in minutes): 0

Logging

Log Path: .\Logs

Log Level: Error [✓ Warning] Information Verbose ☐ Append Logs

☑ Categorized Logs

Configuration

Extract: .\Configurations\ExtractConfiguration.xml

Transform: .\Configurations\TransformConfiguration.xml

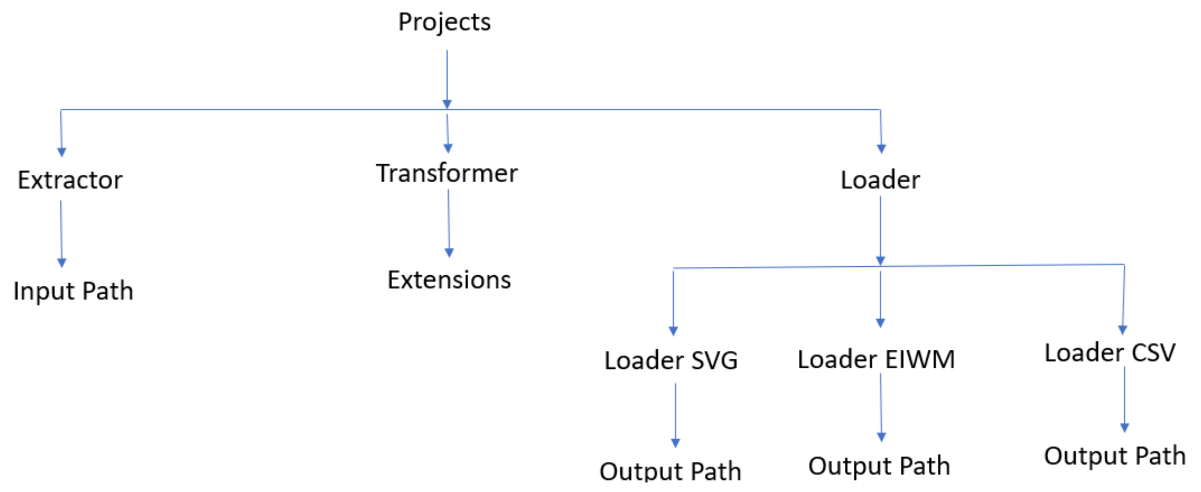
Load: .\Configurations\LoadConfiguration.xml

✓ New project with name "Test" created and loaded successfully.

Example of Relative paths in the Project file:

```
<transformerConfigLocator path=".\Configurations\TransformConfiguration.xml" />
<extractorConfigLocator path=".\Configurations\ExtractConfiguration.xml" />
<loaderConfigLocator path=".\Configurations\LoadConfiguration.xml" />
<log level="Warning" folderPath=".\Logs" appendToLog="false" />
```

Structure of the Project's Configuration and Mapping Files:



Assuming that the path of the project is `C:\Projects\test1\test1.xml`, set the other configuration files as follows:
Relative path is: `.\Configurations\ExtractConfiguration.xml`

Global path is: `C:\Projects\test1\Configurations\ExtractConfiguration.xml`

Relative Path is `.\Logs`

Global path is: `C:\Projects\test1\Logs`

Relative path is `..\Mappings\Patterns2d.xml`

Global path is: `C:\Projects\test1\Mappings\Patterns2d.xml`

General Area

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

To complete the project related information:

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

   Note: The **Project Name** is used as the name of the project configuration file and so 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. If set to 0 then no timeout is applied.

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

Logging Area

To specify different logging information, such as warning and error messages, use the fields in the **Logging** area. 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 file. 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.

Log Level	Severity
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 is 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.

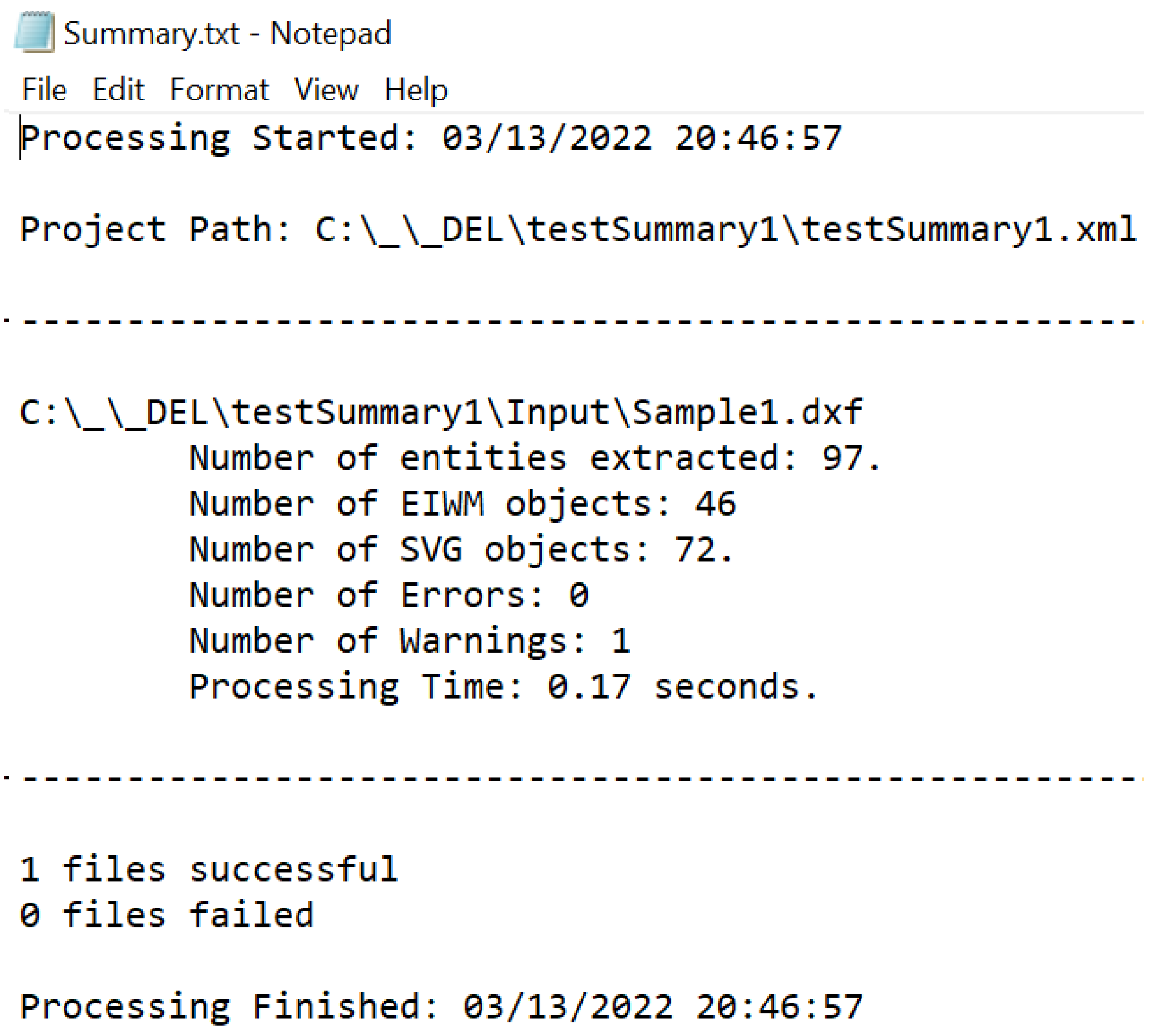
   ```
   <log level="Warning" folderPath="C:\Users\<user_name>\Test\Logs_Appended"
   appendToLog="false" />
   ```

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 SVG objects	SVG Loader	Number of 2D graphical objects in Object Model that were loaded into the output SVG 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.

Example:



```
Processing Started: 03/13/2022 20:46:57

Project Path: C:\_\_DEL\testSummary1\testSummary1.xml

-----------------------------------------------------

C:\_\_DEL\testSummary1\Input\Sample1.dxf
        Number of entities extracted: 97.
        Number of EIWM objects: 46
        Number of SVG objects: 72.
        Number of Errors: 0
        Number of Warnings: 1
        Processing Time: 0.17 seconds.

-----------------------------------------------------

1 files successful
0 files failed

Processing Finished: 03/13/2022 20:46:57
```

Configuration Area

To define the project configuration details, use the fields in the **Configurations** area.

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 input file(s). 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 file. 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 file. 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, 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 .

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

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

Separate Processing

The processing of large source files can sometimes consume so much of the server's resources that it results in system processing issues. To log these types of issues, the Gateway processing can be split between one process that loads the Gateway's configurations and another process that executes the project defined by those configurations. This then allows the launch process to log any issue due to the executing process failing. It also allows the launch process to terminate the executing process due to a timeout or cancellation.

This function is enabled by setting `<SeparateProcessing>true</SeparateProcessing>`. As this is mainly relevant for only large input files, the default value is false.

Repeat processing on failure

When an executing process fails due to a temporary lack of server resources, such as when another process is using most of the resources and then terminates, then it's possible that repeating the Gateway execution process may succeed when the system resources are available. When using Separate Processing, it's possible to also control the number of times to attempt to retry a failed process:

```
<RepeatOnFailure>3</RepeatOnFailure>
```

The above setting means that after a failed processing of the Gateway, the executing process will be repeated up to 3 times.

Note: These settings are only available by editing the main Gateway project configuration file.

Extract Settings

The Gateway can extract data from a variety of input files located in either a local File System or an Amazon Web Services (AWS) S3 Bucket, and for AutoCAD files, also an AVEVA Cloud Storage folder.

The details of the particular input storage to be read by the project are defined in the extract configuration in the `<input>` element, via the `source` parameter, for example, `<Input source="FileSystem">`.

Note: The values of the `source` attribute can be a case-insensitive representation of either "**FileSystem**" or "**S3**" or **AVEVA Cloud Storage**.

The sources from where the Gateway can extract data are as follows:

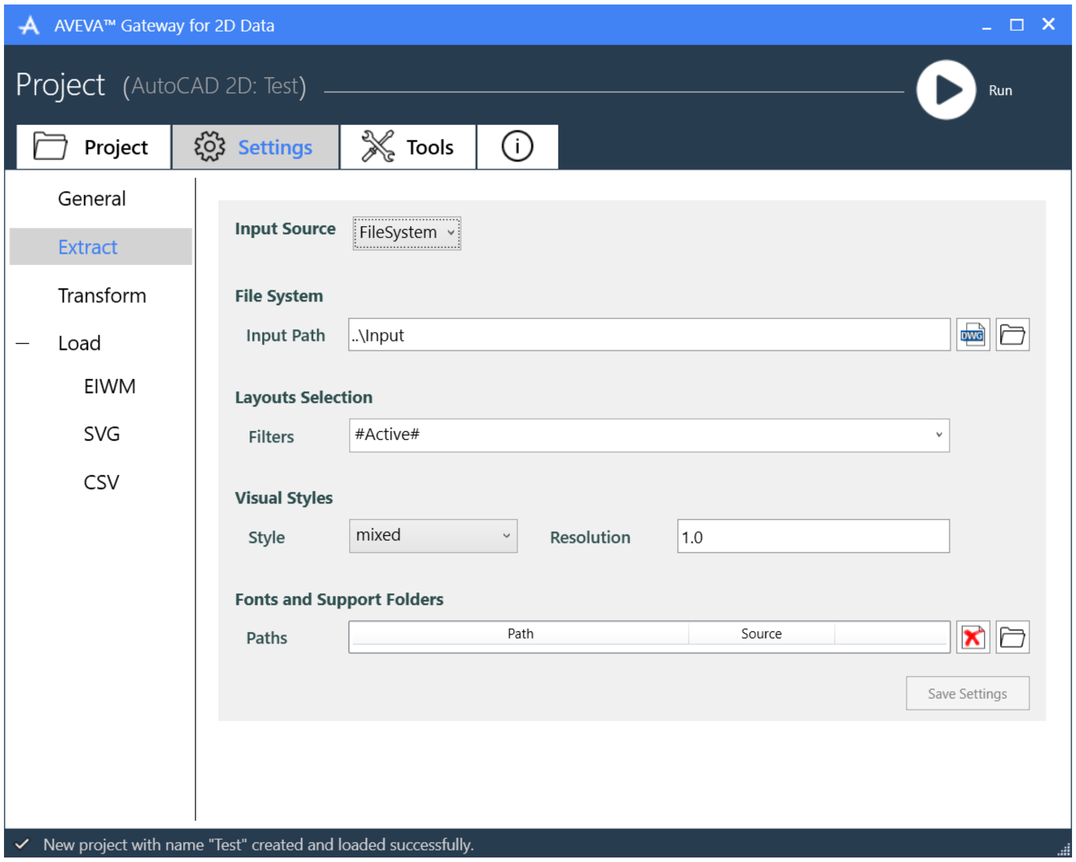
- AutoCAD 2D
- MicroStation 2D
- DEXPI

Extract AutoCAD 2D Files

The Gateway extracts the files from AutoCAD 2D data sources such as .DWG**/**.DXF files.

To configure extraction of the AutoCAD 2D files:

1. Click **Extract** from the left panel.
2. Select the type of **Input Source** from the drop-down box, that is, **File System**, **S3** or **AVEVA Cloud Storage**.



Input Source: FileSystem
Input Path: ..\Input
Filters: #Active#
Style: mixed
Resolution: 1.0

New project with name "Test" created and loaded successfully.

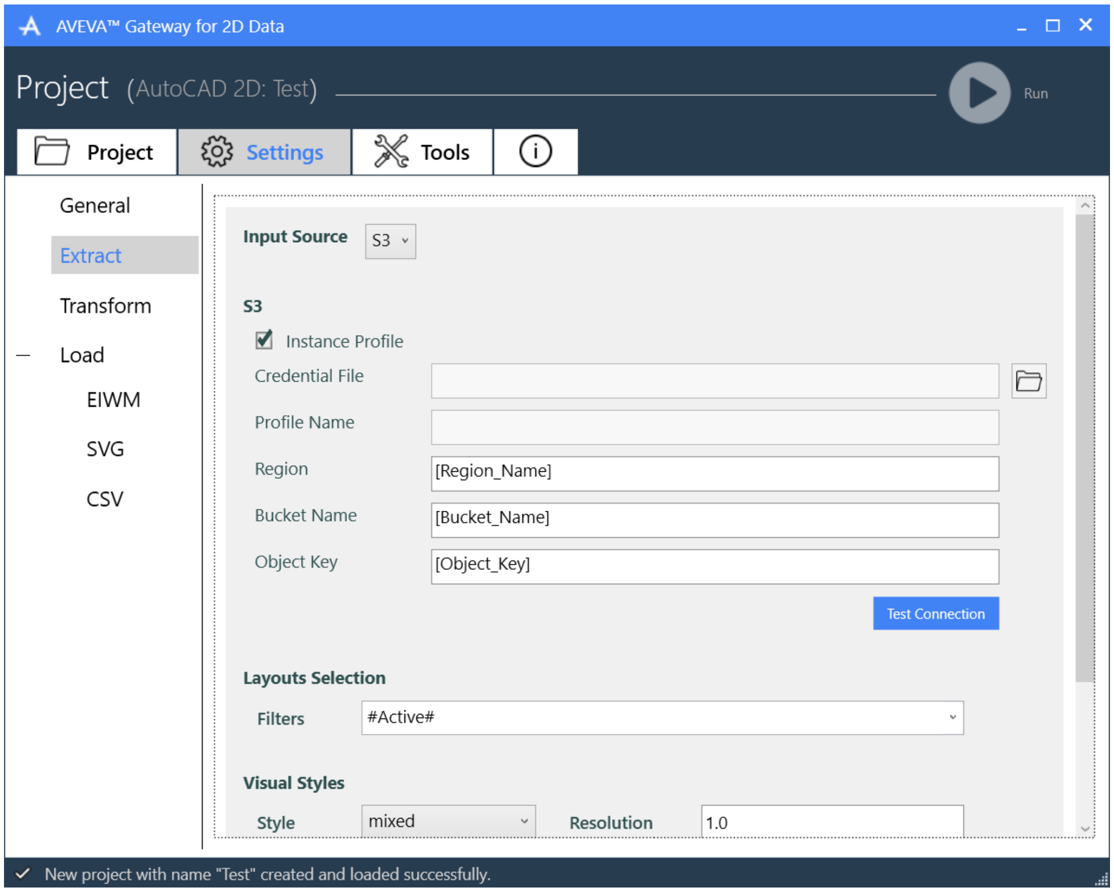
For FileSystem as Input Source:

1. Define the **Input Path** for the location of the input files (.DWG**/**.DXF files), either typing directly or use the **Browse File Path** to select a single file or **Browse Folder Path** to select a folder of files.
2. Define the **Layouts Selection** details, **Visual Styles**, and **Fonts and Support Folders** as mentioned below.
3. Select **Save Settings**.

For S3 as Input Source:

To configure the extractor for S3 as input source:

1. Define the elements in S3 Credential Details section. For more information, see Access an AWS S3 Bucket.
2. If you want to test connection to S3 Bucket, select **Test Connection**. The result of the test is displayed in the bottom left status bar.
3. Define the **Layouts Selection** details, **Visual Styles**, and **Fonts and Support Folders** as mentioned below.
4. Select **Save Settings**.



Input Source: S3

S3

- [✓] Instance Profile
- Credential File:
- Profile Name:
- Region: [Region_Name]
- Bucket Name: [Bucket_Name]
- Object Key: [Object_Key]

Layouts Selection

Filters: #Active#

Visual Styles

Style: mixed — Resolution: 1.0

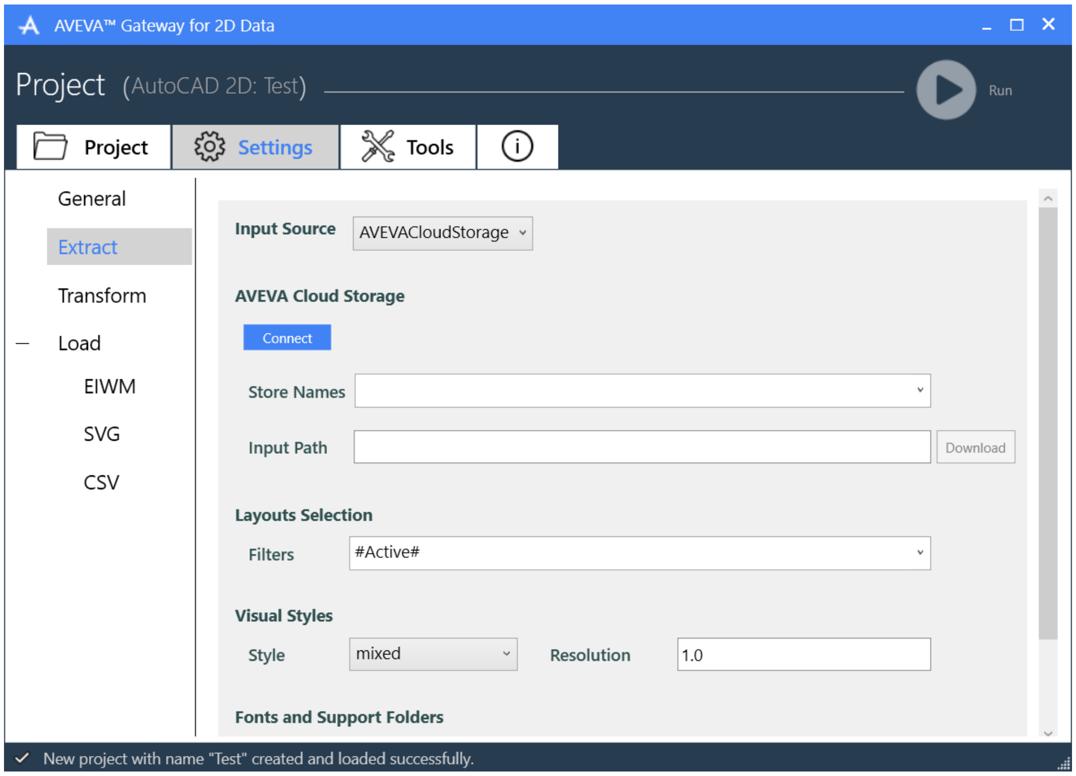
New project with name "Test" created and loaded successfully.

For AVEVA Cloud Storage as Input Source:

To configure the extractor for AVEVA Cloud Storage as input source:

1. Click **Connect** to establish connection with the AVEVA Cloud Storage. For more information about accessing AVEVA Cloud Storage, see Access AVEVA Cloud Storage.
2. Select the **Store** where you have uploaded the .DWG/.DXF files.

   The **Input Path** name can be given as a **Directory Name** or a **FileName**.



Input Source: AVEVACloudStorage

AVEVA Cloud Storage

Store Names:

Input Path:

Layouts Selection

Filters: #Active#

Visual Styles

Style: mixed

Resolution: 1.0

Fonts and Support Folders

✓ New project with name "Test" created and loaded successfully.

3. Define the **Layouts Selection** details, **Visual Styles**, and **Fonts and Support Folders**, as mentioned below.
4. Select **Save Settings**.

Layouts Selection:

Each .DWG file contains a Model space where the graphics of the model (drawing) are created and other layouts called paper spaces. Designers often use the paper spaces to add title blocks, tables, notes, dimensions, and even change the scale of the drawing for plotting purposes. Designers sometimes create these plotting entities in the Model space and do not use paper spaces at all, but usually one or more of the paper spaces get printed and would be expected to be viewed in *Workhub and Dashboard*.

To choose the layout view or views that needs to be rendered into one or more SVG files, define the **Layouts Selection** filter. This can contain one or more keywords or regular expressions, separated by a pipe "|" character. The possible keywords are:

- #Active# - extracts the last viewed layout when the drawing was saved (this is the default filter).
- #All# - extracts all layouts.
- #AllPaper# - extracts all paper layouts.
- #Model# - extracts the Model space.
- #1st paper# - extracts the first paper space.

The Layouts filter contains the name of a layout or a regular expression that can filter multiple names of paper spaces.

Example:

- #Model#|Tank.* - extracts the Model space and all paper spaces matched to the regular expression "Tank.*".

- #1st paper#|#Model# - extracts the Model space and first paper space after Model space.

Notes:

- It is possible to set tracking of changes on Extraction of data using annotations feature. For information about annotations, see Tracking Changes in Data.
- The Gateway does not extract any OVERLAID XREF files in models.
- The Gateway can identify some key attributes, for example, `ObjectID`, `ObjectName`, `Revision` and so on will be exported into EIWM file even without mapping in configuration. For more information about these key reserved attributes, see Appendix F: Reserved Attributes Used in the Gateway.

Visual Styles

AutoCAD can project three-dimensional models into two-dimensional viewports in several different styles, and supports multiple pattern styles that do not render efficiently in the SVG format. These can be controlled by extraction configuration to reduce processing and SVG rendering time:

- Specified hatches can be exported with solid fill type.
- Viewports can be exported as vectors or raster images (embedded images).

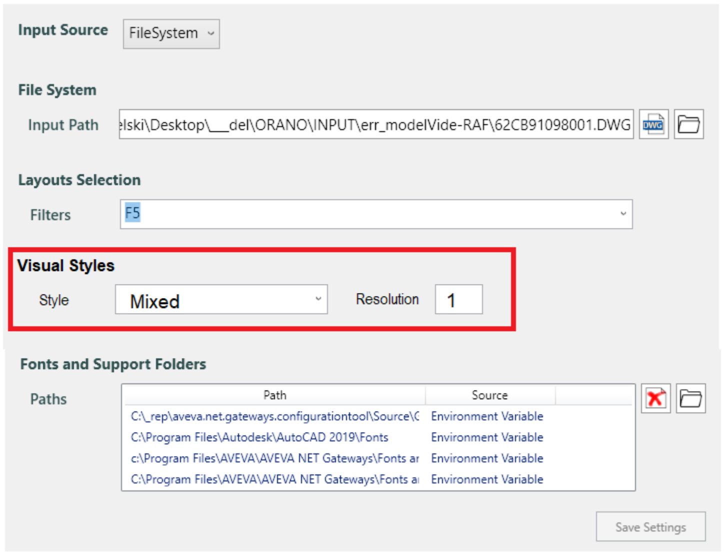
Notes:

- Data to raster image conversion should be used with caution as it can result in the loss of engineering data and tagging capabilities.
- Complex Hatch patterns can only be mapped to Solid patterns.

Example Extractor Configuration file:

```
<VisualStyles>
  <Viewports>
    <Viewport style="mixed" resolution="1"/>
  </Viewports>
  <Blocks>
    <Block name="companyLogo1" imageLocation="c:\logo1.png"/>
    <Block name="companyLogo2" imageLocation="c:\logo2.png"/>
  </Blocks>
  <Hatches>
    <Hatch inputFillPattern="DOTS" maxScale="0.1" outputFillPattern="SOLID"/>
    <Hatch inputFillPattern="AR-CONC" maxScale="0.1" outputFillPattern="SOLID"/>
  </Hatches>
</VisualStyles>
```

Select style and resolution for viewports in GUI:


Input Source: FileSystem

File System
Input Path: :lski\Desktop\__del\ORANO\INPUT\err_modelVide-RAF\62CB91098001.DWG

Layouts Selection
Filters: F5

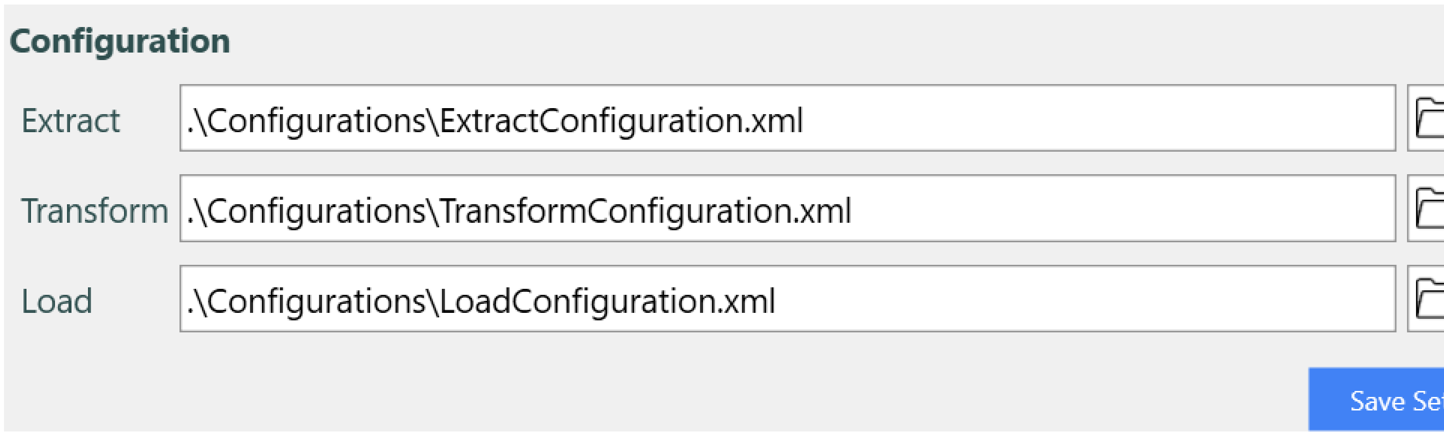
Visual Styles
Style: Mixed
Resolution: 1

Fonts and Support Folders
Paths

Path	Source
C:_rep\aveva.net.gateways.configurationtool\Source\C	Environment Variable
C:\Program Files\Autodesk\AutoCAD 2019\Fonts	Environment Variable
c:\Program Files\AVEVA\AVEVA NET Gateways\Fonts ar	Environment Variable
C:\Program Files\AVEVA\AVEVA NET Gateways\Fonts ai	Environment Variable

Access to the settings for the 'Blocks' and 'Hatches' nodes is via editing the configuration file and not available in the User interface.

Navigate to 'General' panel to edit the **Extract Configuration** file:


Configuration

Extract: .\Configurations\ExtractConfiguration.xml

Transform: .\Configurations\TransformConfiguration.xml

Load: .\Configurations\LoadConfiguration.xml

Viewports

The AutoCAD Gateway contains 10 types of Viewport styles:

- 2D Wireframe
- Wireframe
- Hidden
- Sketchy
- Realistic
- Shaded
- Shaded with edges
- Shades of Grey
- X-ray

- Conceptual (custom)

Viewport content can be extracted as wireframe or raster.

Setting "mixed"

- 2D Wireframe, Wireframe - extracted as wireframe.
- Hidden, Sketchy, Shaded with edges, Shades of Grey, X-ray, Realistic, Shaded - extracted as raster.
- Conceptual - depends on custom view settings.

Setting "wireframe"

- All viewports extracted as wireframe.

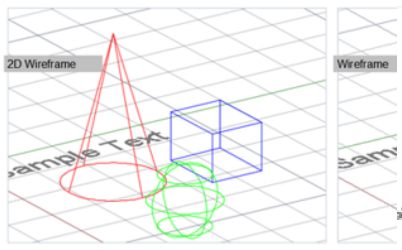
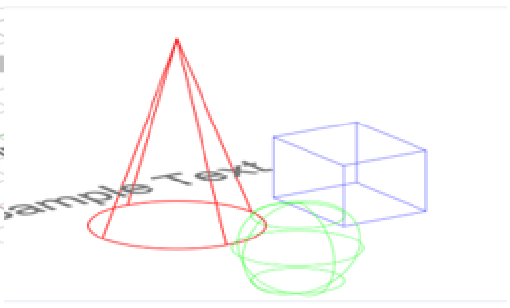
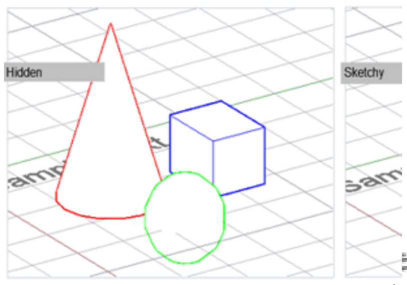
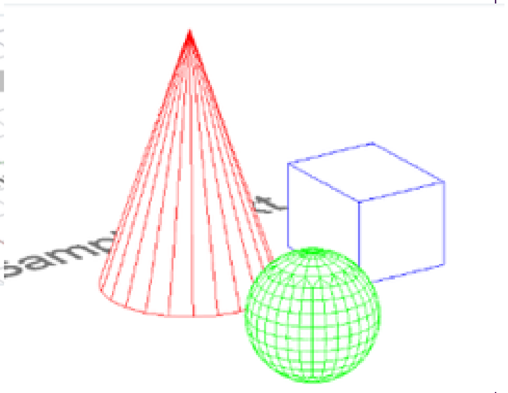
Setting "raster"

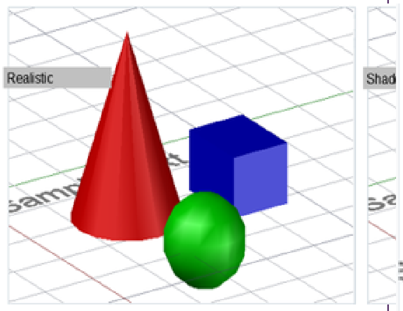
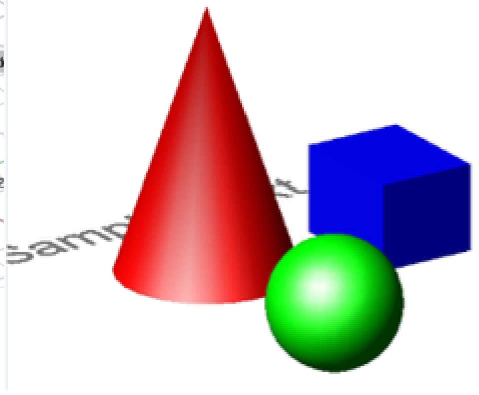
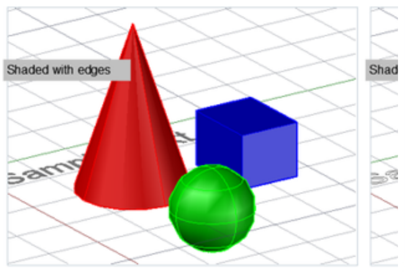
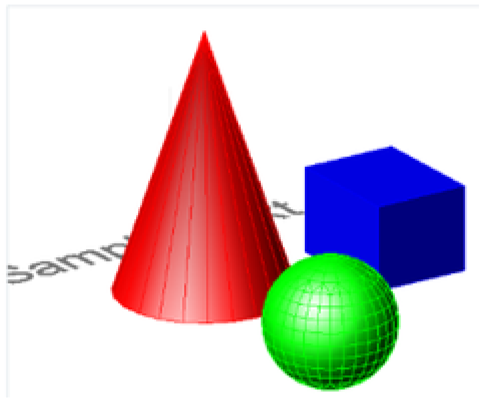
- All viewports extracted as raster images.

Notes:

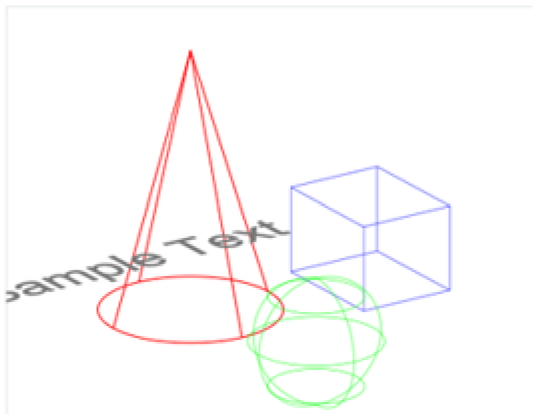
- Model space is also treated as a view port.
- A raster image is not scalable like a block's vector graphics so it loses its quality when stretched. Therefore, you should prepare the raster image with the same aspect ratio as the block's graphic.

The table below compares the styles in AutoCAD and SVG for the "mixed" setting.

AutoCAD	SVG
	 Wireframe
	 Raster

AutoCAD	SVG
	 Raster
	 Raster

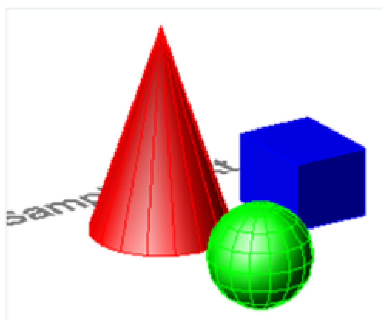
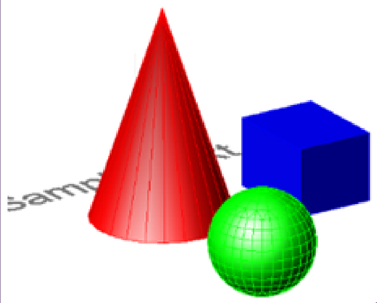
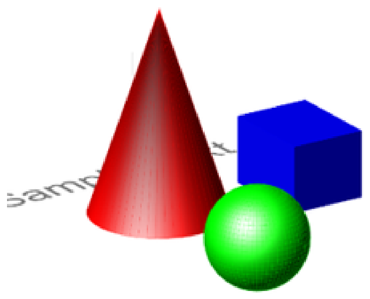
For the "wireframe" setting, all viewports are exported to SVG as wireframe graphics.

AUTOCAD	SVG
All types: 2D Wireframe Wireframe Hidden Sketchy Shaded with edges Shades of Grey X-ray Realistic Shaded Conceptual (custom)	

AUTOCAD	SVG

The following table compares raster quality in SVG for different "resolution" setting:

Resolution = 0.5, resolution = 1 (default), resolution = 5

Resolution = 0.5	Resolution = 1 (default)	Resolution = 5
		

Notes:

- Minimum resolution factor: 0.1.
- Maximum resolution factor: 10.
- A resolution of 5 creates crisp output and for most models higher resolution values only result in larger SVG files that take longer to render.
- The possible image formats can be: PNG, JPEG, and BMP.

Blocks

Specified block replaced with raster image.

Example of Block Configuration:

```
<Blocks>
<Block name="companyLogo1" imageLocation="c:\logo1.png"/>
<Block name="companyLogo2" imageLocation="c:\logo2.png"/>
</Blocks>
```

Hatches

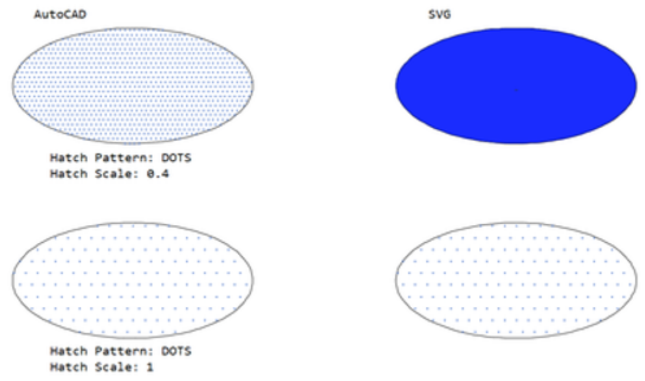
A pattern specified in "inputFillPattern" and with a scale that does not exceed the value specified in "maxScale" is mapped to the "outputFillPattern" (only SOLID fills are available).

Note: This mapping is part of the extraction process and is always done prior to any color presentation mapping. Therefore, the color of the output solid is the same as the input pattern unless modified by a color mapping,

Example:

A DWG drawing contains two hatches with a 'DOTS' pattern. The first hatch has a scale 0.4 and the second hatch 1.0. If the following configuration is specified, then the first will be exported as a solid fill and the second will be rendered in the original DOTS pattern as its scale is greater than the maxScale:

```
<Hatch inputFillPattern="DOTS" maxScale="0.5" outputFillPattern="SOLID"/>
```



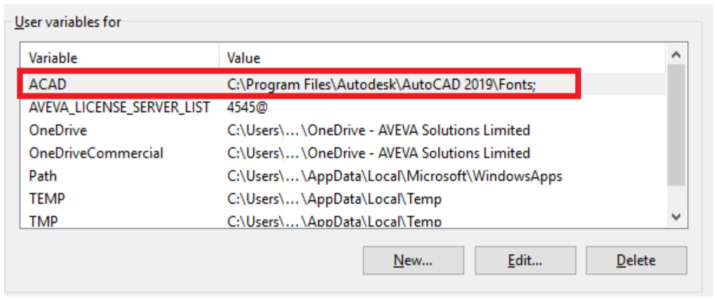
Fonts and Support Folders:

Some drawing elements use pre-defined shapes that are defined in separate files, for example, fonts. If those files are not in the same location as the drawing files, then you must specify the Support Folder, either in the project's configuration or in a system variable.

Specify the Support Folder location for the project in the Extract Configuration page.

As it is common to locate all font files in a common system folder, set this location in a **System Environment Variable** under the User variables for section.

The Gateway reads any location (or locations) saved into a system environment variable named ACAD along with the locations defined in the project configuration. Multiple folder paths should be separated by a semicolon, as shown in the following image:


User variables for

Variable	Value
ACAD	C:\Program Files\Autodesk\AutoCAD 2019\Fonts;
AVEVA_LICENSE_SERVER_LIST	4545@
OneDrive	C:\Users\...\OneDrive - AVEVA Solutions Limited
OneDriveCommercial	C:\Users\...\OneDrive - AVEVA Solutions Limited
Path	C:\Users\...\AppData\Local\Microsoft\WindowsApps
TEMP	C:\Users\...\AppData\Local\Temp
TMP	C:\Users\...\AppData\Local\Temp

Note: The ACAD environment variable defines the location of the SHX fonts directory.

SHX Definition File Processing

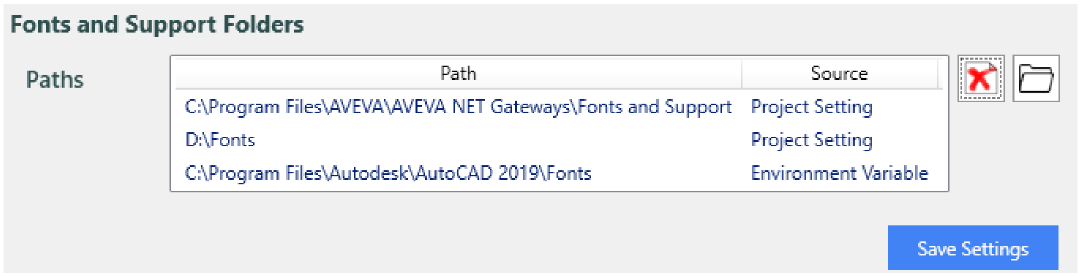
The SHX definition file used in the DWG/DXF drawing being processed is available during processing to create output graphics. SHX fonts can be placed in any location accessible from the Gateway program. Common SHX fonts are delivered together with AutoDesk AutoCAD or TrueView installations. An example default location is "`C:\Program Files\Autodesk\AutoCAD 2019\Fonts`".

- Defined directly in the system settings: The access path can be defined directly in the system settings as following: Environment Variables -> User Variables, "ACAD" variable.
- Customize the fonts: They can be also used custom fonts in a DWG/DXF drawing.
- Accessible: The file path is accessible during processing.
- Define Multiple Paths: The variable "ACAD" takes multiple paths to different locations of SHX files. If you set them in system settings, then separate them with semicolons, for example: "`C:\Fonts;E:\Custom Fonts`".

- If you do not want to use the environment variable "ACAD" to indicate the location of SHX fonts, do so in the Gateway's Extractor's settings.
  - Select "Browse for folder containing the fonts and support files" to add a location.
  - Select "Delete selected paths" to remove the selected location.

Note: A special case is the path where the processed DWG/DXF input file is located. In this location, the program always looks for SHX fonts.

Define multiple support folders in the project. The following image shows multiple support folders marked as `Project Setting` in the `Source` column. Any folders defined via the System User Variable are marked as `Environment Variable`.

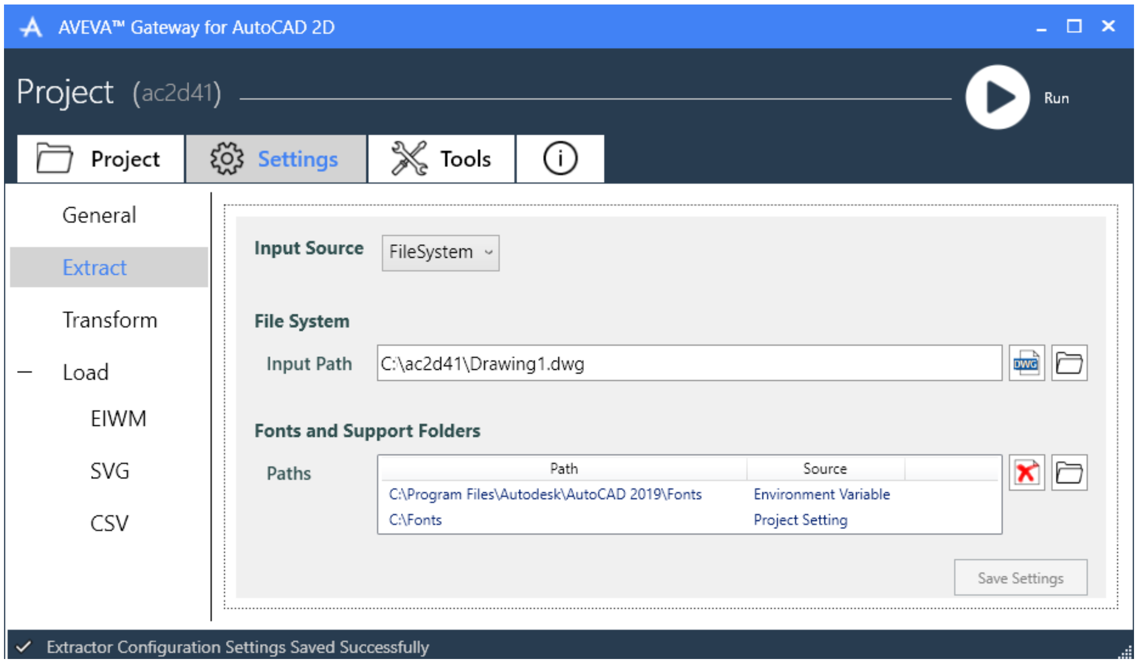


Path	Source
C:\Program Files\AVEVA\AVEVA NET Gateways\Fonts and Support	Project Setting
D:\Fonts	Project Setting
C:\Program Files\Autodesk\AutoCAD 2019\Fonts	Environment Variable

Location order when defining the SHX fonts is as follows:

- SHX temporary replacement folder - Not visible for user (see 'SHX to SHX' mapping in the below SHX fonts subsection). This is used only internally by the program for the mapping
- Locations defined in Environment Variables
- Locations defined in the Extractor configuration
- Location of input drawing

Multiple support folders can be defined in the project and are marked 'Project Setting' in the 'Source' column. Any folders defined via the System User Variable are marked as 'Environment Variable'.



Input Source: FileSystem

File System

Input Path: C:\ac2d41\Drawing1.dwg

Fonts and Support Folders

Paths

Path	Source
C:\Program Files\Autodesk\AutoCAD 2019\Fonts	Environment Variable
C:\Fonts	Project Setting

Extractor Configuration Settings Saved Successfully

Notes:

- **Limitation in the User Interface**: After the first processing, changing the paths set has no effect. If you want to use a different set of paths, then close the program, reopen, make changes, and start processing. It also applies if you want to change the list of file paths in the same project.
- Environment Variable cannot be modified from the Gateway's configuration page.

SHX fonts

The SHX font group is specific to AutoDesk AutoCAD drawings. It is a font of the Compiled Shape File type (SHX). It consists of font definitions as well as shape definitions for displaying the text.

During processing, all texts that use the font defined by SHX are vectorized and appear as polylines in the SVG output file. This is desirable when you want to accurately reproduce the appearance of the text. This solution usually causes SVG output files to be larger.

Note: If an SHX font is not made available then the Gateway will use the SHX "Simplex" font as a default to vectorize the text.

This vectorization feature can be configured by mapping of the SHX fonts. This mapping uses Transformer extension 'Text Mapping'. It contains section 'Font Mapping' which implements True Type mapping and CAD font mapping. If there is a mapping that corresponds to SHX it is passed to extractor. All CAD mappings are passed in parameter to extractor in one package. This is resolved by AC2D gateway invoker.

The following SHX font mapping combinations are available: SHX to SHX and SHX to TTF.

- **SHX to SHX**

  Consider mapping `simplex.shx` to `complex.shx`. The Gateway searches for the file `complex.shx` in all 'Fonts and support folders' and copies it to a temporary replacement folder (`%TEMP%\ACAD_SHX_REPL`) where its contents are used to create the `complex.shx` graphical elements at the locations of the input `simplex.shx` graphical elements.

  Example SHX to SHX mapping:

```
   <font from="simplex.shx" to="complex.shx" regEx="false" />
```

This expression maps SHX font "Simplex" to SHX font "Complex", so output SVG will contain text represented as polylines as a result of vectorization using SHX font "Complex".

Notes:

- Before each processing, this folder is cleaned up.
- SHX Font mapping attributes such as sizeFactor, italic or bold for are not supported.

- **SHX to TTF**

  Example SHX to True Type mapping:

  ```
     <font from="simplex.shx" to="arial" regEx="false" />
  ```

  This mapping maps SHX font "`simplex`" to True Type "`arial`". In output SVG, there will text with font family "Arial".

  It can be used regular expression in the "`from`" attribute.

  Note: If you want to map SHX font then regardless if you use RegEx form or not, the field "from" must contain "`.shx`" string to be differentiated from True Type font where setting extension is not necessary.

  Example SHX to True Type mapping with using regular expression:

  ```
     <font from="[a-z]*plex.shx" to="arial" regEx="true" />
  ```

  This expression maps all SHX fonts where filenames end with "`plex.shx`" (for example, `simplex.shx`, `complex.shx`) to True Type "Arial" font. In output SVG, text with font family "Arial" appears.

The following SHX font mapping combination is not available:

- **TTF to SHX** – The program does not allow the configuration to be loaded (when opening the project) or it will return an error when trying to save the Text Mapping extension configuration.

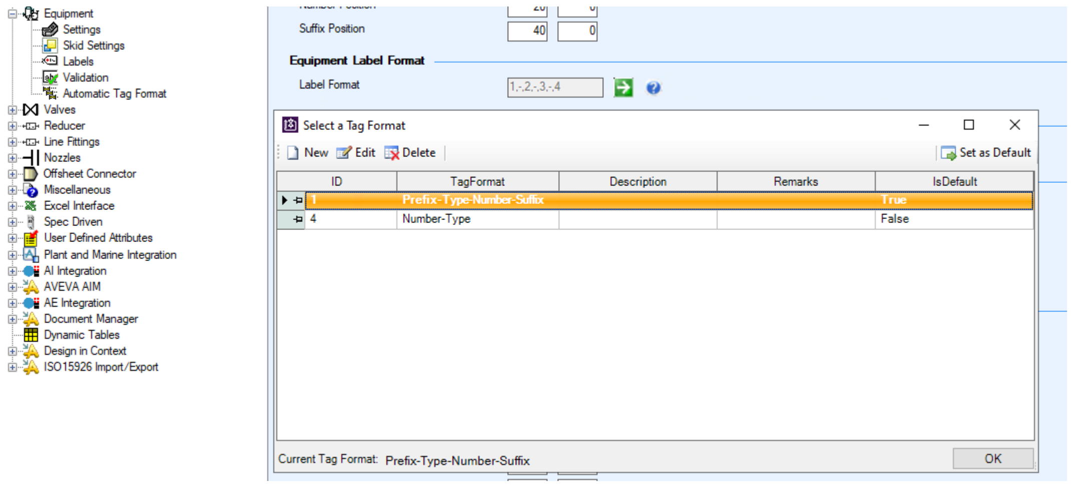
Logging

All fonts mappings are logged.

Use Equipment Display Tag

In an AVEVA P&ID Drawing, you can invoke multiple Tag rules for Equipment Element type and can set any one of them as the Display Tag over the Drawing instead of the Default Tag naming rule.

For example, in the following label settings for Equipment, Naming rule 1 is set to **‘True’** and so becomes the Default naming rule.



Suffix Position 40 0

Label Format 1,-,2,-,3,-,4

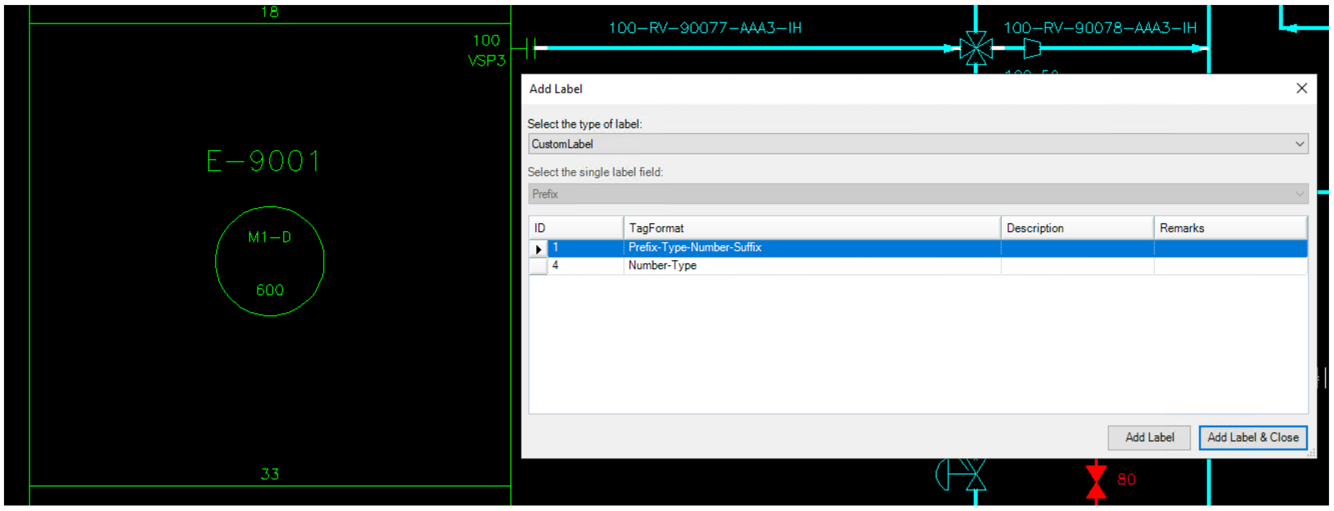
ID	TagFormat	Description	Remarks	IsDefault
1	Prefix-Type-Number-Suffix			True
4	Number-Type			False

Current Tag Format: Prefix-Type-Number-Suffix

Setting a different Display Tag for selected Equipment:

The AVEVA P&ID user can then change the display tag for any Equipment and choose the required naming rule by simply using the context menu options of the selected Equipment.

Example of an Equipment with Display tag set using the default naming rule:



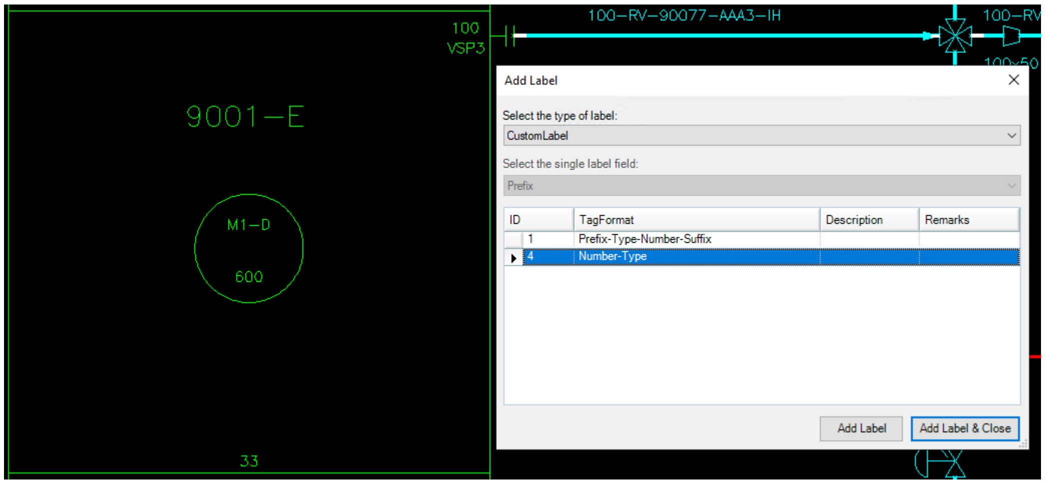
Add Label

Select the type of label: CustomLabel

Select the single label field: Prefix

ID	TagFormat	Description	Remarks
1	Prefix-Type-Number-Suffix		
4	Number-Type		

Example of the same Equipment with Display tag set using a different naming rule:



Add Label

Select the type of label: CustomLabel

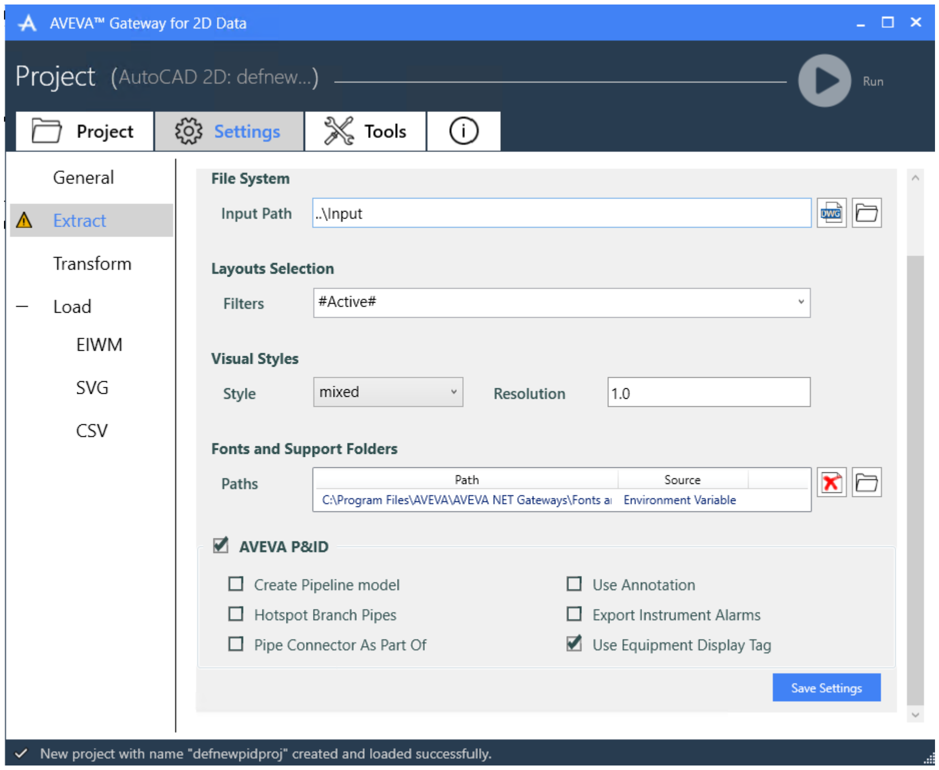
Select the single label field: Prefix

ID	TagFormat	Description	Remarks
1	Prefix-Type-Number-Suffix		
4	Number-Type		

Although this display tag function in AVEVA P&ID may change what is displayed in the AVEVA P&ID drawing, the saved DWG files still contains the data needed to derive the Default naming rule value, which the Gateway user might want to continue to use as the Object ID of the Equipment object in AIM.

In such cases, you can select the "**Use Equipment Display Tag**" option to either use the display tag as the Object ID of the Equipment or instead use the Default naming rule as the Object ID.

Note: The Equipment text in the SVG is not affected, and remains the same as displayed in AVEVA P&ID.



File System
Input Path: ..\Input

Layouts Selection
Filters: #Active#

Visual Styles
Style: mixed
Resolution: 1.0

Fonts and Support Folders
Paths

Path	Source
C:\Program Files\AVEVA\AVEVA NET Gateways\Fonts a	Environment Variable

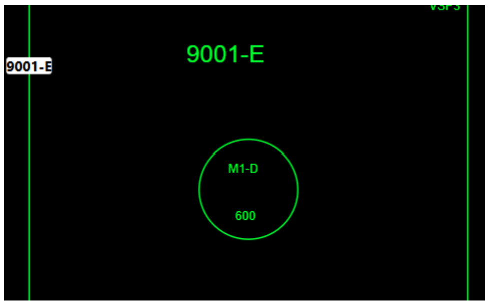
[✓] AVEVA P&ID

- [ ] Create Pipeline model
- [ ] Hotspot Branch Pipes
- [ ] Pipe Connector As Part Of
- [ ] Use Annotation
- [ ] Export Instrument Alarms
- [✓] Use Equipment Display Tag

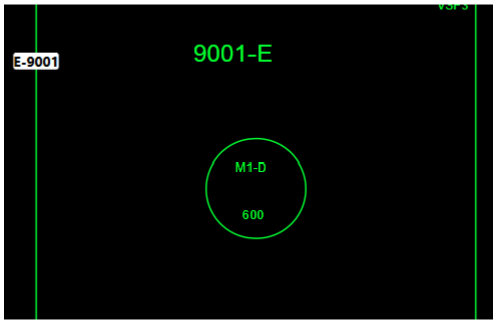
New project with name "defnewpidproj" created and loaded successfully.

When creating a new project the default value for this option is unchecked.

Hotspotting of Equipment in AIM when the Gateway option is checked:



Hotspotting of Equipment in AIM when the Gateway option is not checked:



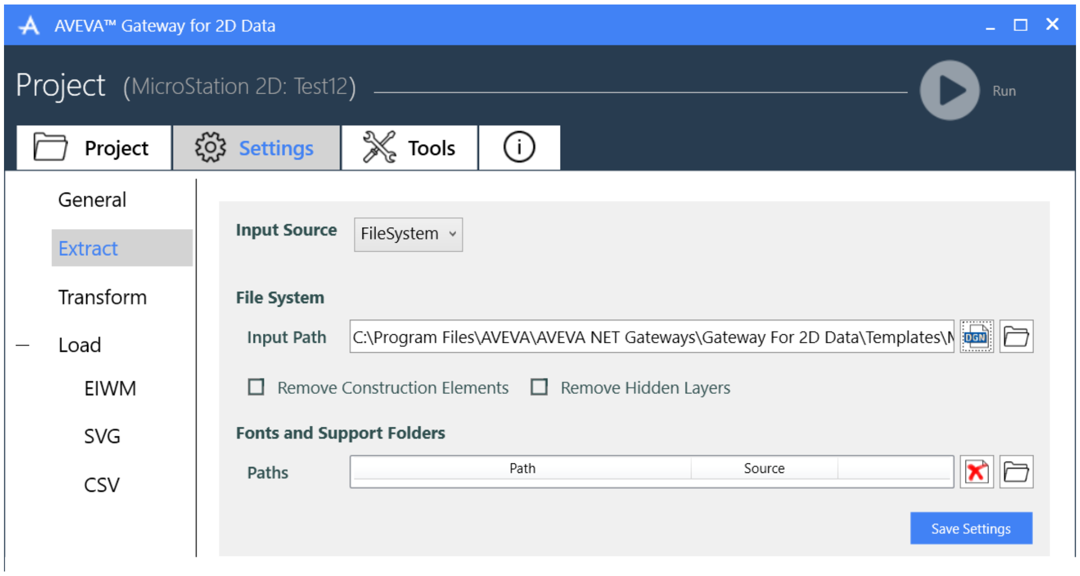
Extract MicroStation 2D Files

The Gateway extracts the files from MicroStation 2D data sources such as .DGN files.

Note: MicroStation drawings that refer to PDF and/or .DWG and/or Excel files are processed without including the referenced files.

To configure extraction of the MicroStation 2D files:

1. Click **Extract** from the left panel.
2. Select the type of **Input Source** from the drop-down box, that is, **File System** or **S3**.



Input Source: FileSystem

File System

Input Path: C:\Program Files\AVEVA\AVEVA NET Gateways\Gateway For 2D Data\Templates\M

☐ Remove Construction Elements ☐ Remove Hidden Layers

Fonts and Support Folders

Paths

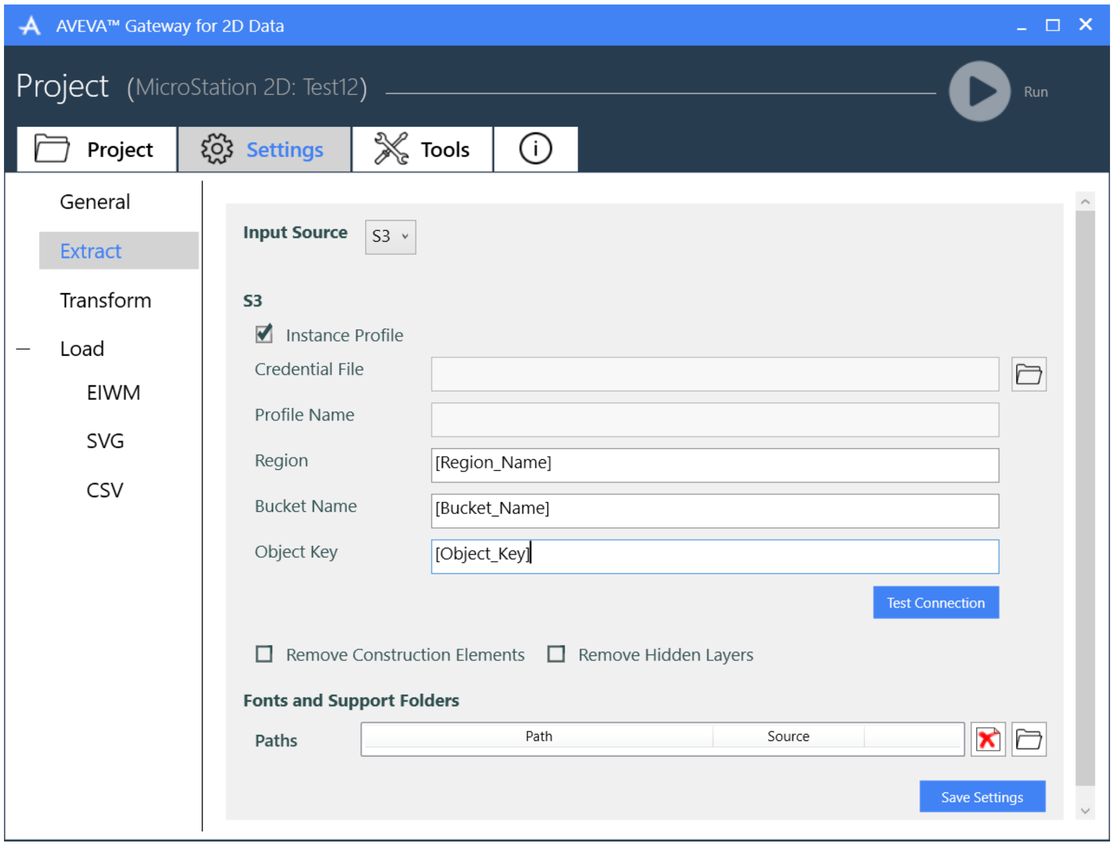
For FileSystem as Input Source:

1. Define the **Input Path** for the location of the input files (DGN files), either typing directly or use the **Browse File Path** to select a single file or **Browse Folder Path** to select a folder of files.
2. Select **Remove Construction Elements** box to exclude the construction elements from the output SVG file. When this option is cleared, it enables export of the construction elements to SVG output file. The default value is **cleared** to include the construction elements in the SVG.
3. If you select **Remove Hidden Layers** option, SVG files are produced without including items and their extents from layers which are set as hidden in the source drawing.
4. Define the **Layout Selection** details, **Visual Styles**, and **Fonts and Support Folders**, as mentioned below.
5. Select **Save Settings**.

For S3 as Input Source:

To configure the extractor for **S3** as input source:

1. Define the elements in **S3** Credential Details section. For more information, see Accessing an AWS S3 Bucket ().
2. If you want to test connection to S3, select Test Connection. The result of the test is displayed in the bottom left status bar.
3. Define the **Layout Selection** details, **Visual Styles**, and **Fonts and Support Folders**, as mentioned below.
4. Select **Save Settings**.



Input Source: S3

S3

☑ Instance Profile

Credential File	
Profile Name	
Region	[Region_Name]
Bucket Name	[Bucket_Name]
Object Key	[Object_Key]

☐ Remove Construction Elements ☐ Remove Hidden Layers

Fonts and Support Folders

Paths

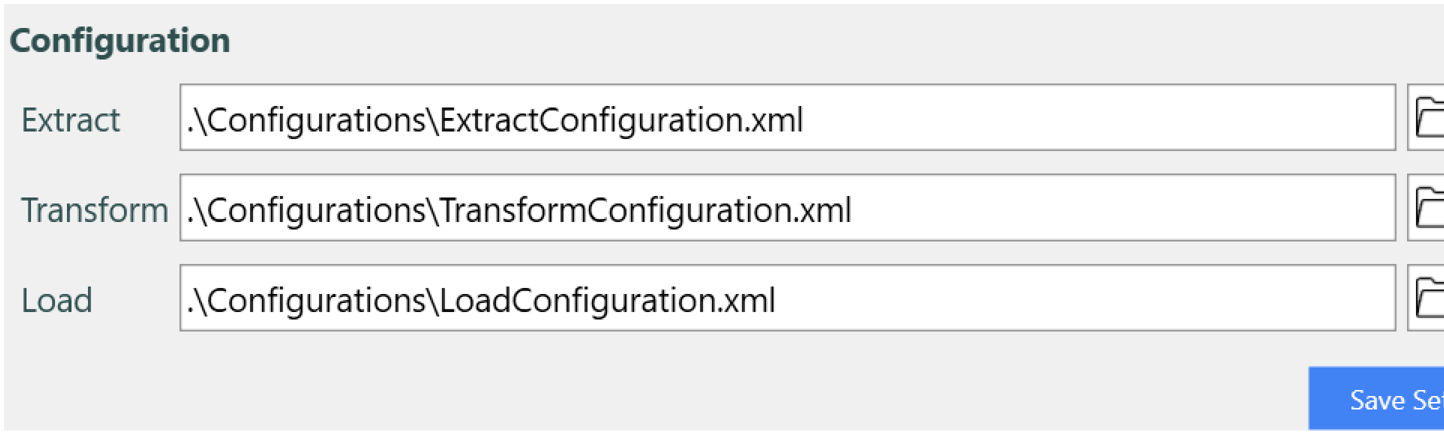
Visual Styles/Hatches

High density hatches can be rendered in the SVG with a solid fill type to reduce the processing and SVG rendering times.

MicroStation supports multiple pattern styles. Some of these that have a high density pattern do not render efficiently in the SVG format and increases the size of the SVG file. If the density is so high that it appears to be a fill pattern when viewed at normal resolutions, then it is recommended to map them to an equivalent-looking solid fill pattern in the SVG.

Note: This mapping is part of the extraction process and is applied prior to any color presentation mapping. Therefore, the color of the output solid is the same as the input pattern but can then be modified by a separate color presentation mapping in the **Transformation** configuration.

The **'Hatches'** node must be updated manually in the configuration file as it is not available in the User interface. Navigate to the **'General'** panel to edit the Extract Configuration file:


Configuration

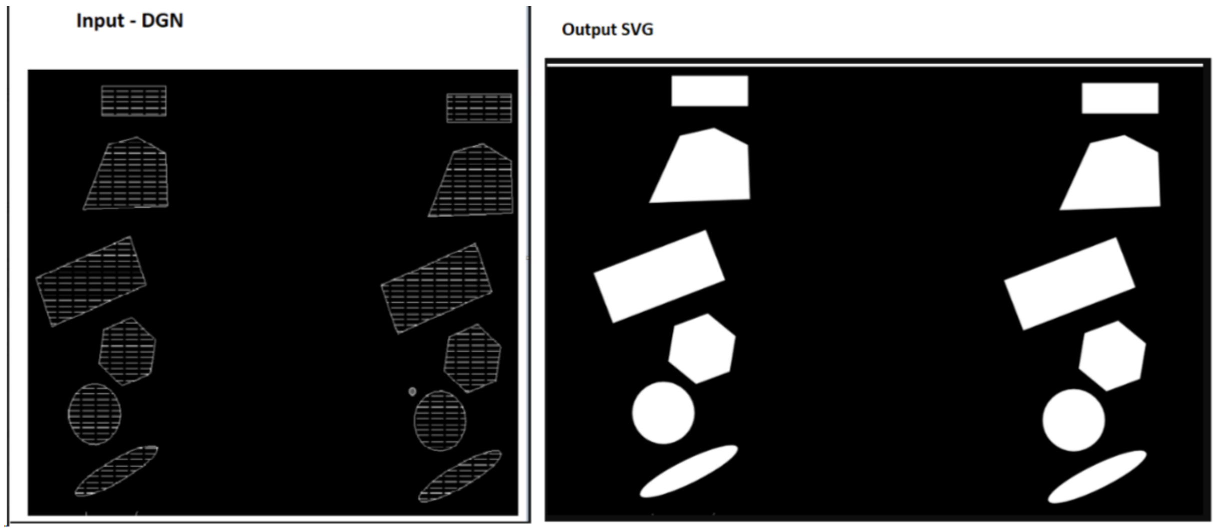
Extract	.\Configurations\ExtractConfiguration.xml
Transform	.\Configurations\TransformConfiguration.xml
Load	.\Configurations\LoadConfiguration.xml

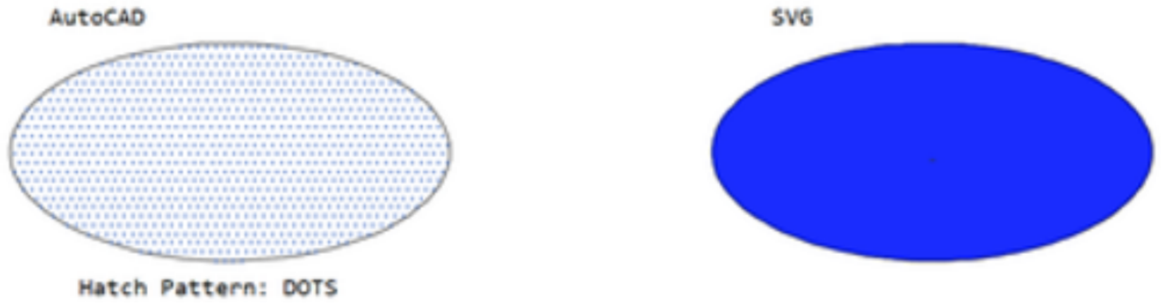
Example Extractor Configuration file:

```
<VisualStyles>
 <Hatches>
  <Hatch inputFillPattern="DOTS" outputFillPattern="SOLID"/>
 </Hatches>
</VisualStyles>
```

This results in the input DOTS pattern being converted to a solid pattern in the SVG.

Note: The MicroStation extractor is unable to read hatch densities from `.dgn` files, therefore, you can't use the `maxScale` parameter to apply the hatch mapping to only high density patterns.





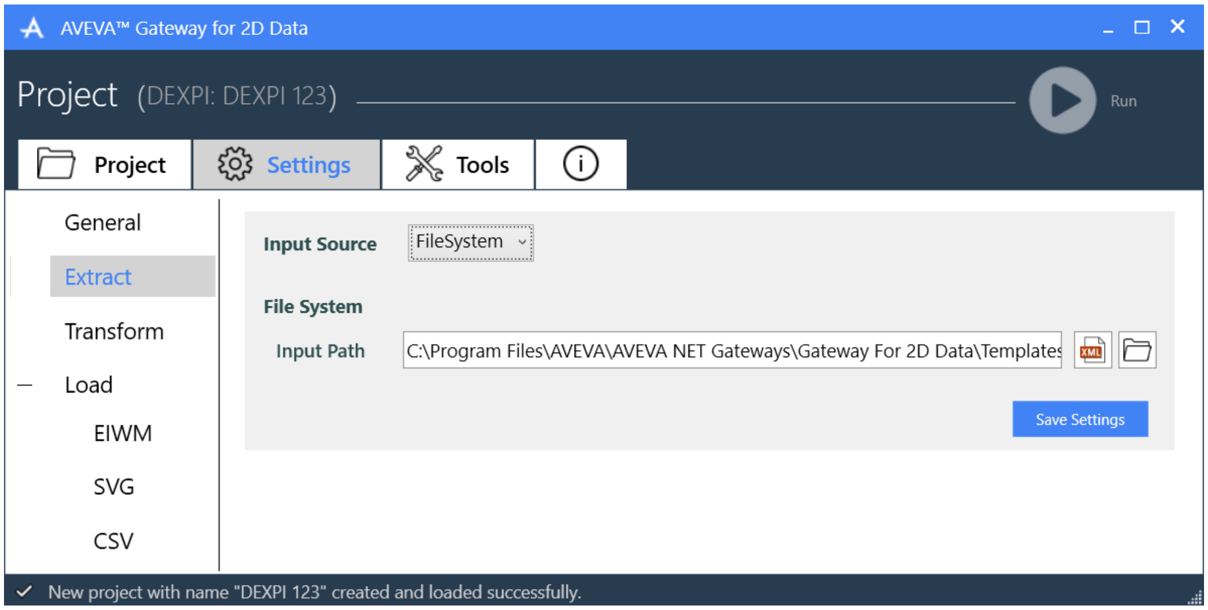
Note: For more information about limitations of a MicroStation 2D drawing, see Appendix I: Known Limitations .

Extract DEXPI Files

The Gateway extracts the DEXPI objects from DEXPI files.

To configure extraction of the DEXPI files:

1. Click **Extract** from the left panel.
2. Select the type of **Input Source** from the drop-down box, that is, **File System** or **S3**.



Input Source: FileSystem

File System

Input Path: C:\Program Files\AVEVA\AVEVA NET Gateways\Gateway For 2D Data\Templates

New project with name "DEXPI 123" created and loaded successfully.

For FileSystem as Input Source:

1. Define the **Input Path** for the location of the input files (DEXPI files), either typing directly or use the **Browse DEXPI File Path** to select a single file or **Browse Folder Path** to select a folder of files.
2. Select **Save Settings**.

For S3 as Input Source:

To configure the extractor for **S3** as input source:

1. Define the elements in **S3** Credential Details section. For more information, see Accessing an AWS S3 Bucket ().
2. If you want to test connection to S3, select **Test Connection**. The result of the test is displayed in the bottom left status bar.
3. Select **Save Settings**.

Identifying DEXPI Objects from a DEXPI File

A DEXPI file contains hierarchical data that represents a digital description of a Plant model, consisting of both graphical and engineering-type elements. These engineering elements can have parameters such as Tag names, maximum pressure, and so on and graphical elements form the P&ID diagram of a Plant model. The Gateway extracts both types of elements so that the engineering elements and their attributes can be converted to an EIWM file and the graphical elements can be converted to an SVG file. These files can then be imported into AIM using the Import Controller.

Engineering elements set to become EIWM objects must have a unique identifier (tag). The Gateway uses default DEXPI parameters for these object IDs, but you can change these parameters, if needed, via mapping configuration in the Gateway's Transformer.

- `File Object` is a mandatory object and contains manifest data about the input file (see Manifest References ()). These data are derived from the Plant model (parent node) in the DEXPI file. As the Plant model does not contain a unique identifier, the Gateway assigns the input file name, as stored in the `#MODEL_NAME#` attribute. For example, if the input file is `PID4545.xml` then the output file will be `PID4545_avngate.xml` or `PID4545_null.xml` and the ID of its document object would be `PID4545`.

The manifest data includes metadata about the input file and attributes of the plant information node.

```
<PlantModel xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="ProteusPIDSchema_4.0.1.xsd">
 <PlantInformation SchemaVersion="4.0.1" OriginatingSystem="AVEVAPID"
Date="2019-01-23"  Time="09:17:02" Is3D="no" Discipline="PID" Units="mm" >
 <UnitsOfMeasure Distance="mm" Pressure="atm"/>
</PlantInformation>
</PlantModel>
```

This manifest data has a document or file object in the EIWM of:

```
<Object>
      <ID>PID4545</ID>
      <Context>
        <ID>ContextId</ID>
      </Context>
      <ClassID>PID</ClassID>
      <Characteristic>
        <Name>Distance</Name>
        <Value>mm</Value>
      </Characteristic>
      <Characteristic>
        <Name>Pressure</Name>
        <Value>atm</Value>
      </Characteristic>
      <Characteristic>
        <Name>SchemaVersion</Name>
        <Value>4.0.1</Value>
      </Characteristic>
      <Characteristic>
        <Name>OriginatingSystem</Name>
        <Value>AVEVAPID</Value>
      </Characteristic>
      <Characteristic>
        <Name>Date</Name>
        <Value>01/23/2019 12:00:00 AM</Value>
      </Characteristic>
      <Characteristic>
        <Name>Time</Name>
        <Value>01/01/0001 9:17:02 AM</Value>
      </Characteristic>
      <Characteristic>
        <Name>Is3D</Name>
        <Value>no</Value>
      </Characteristic>
      <Characteristic>
        <Name>Units</Name>
        <Value>mm</Value>
      </Characteristic>
      <Characteristic>
        <Name>Discipline</Name>
        <Value>PID</Value>
      </Characteristic>
  </Object>
```

As noted above, the ID for the document or File Object in the EIWM is derived from the input file name (without file extension) and not the value of the Drawing object attribute "Name". Although these are often the same, the convention is to use the input file name.

- All other objects in the EIWM file are derived from Engineering elements that have a unique ID and are derived from the PlantItem or AnnotationItem objects defined in the DEXPI schema, as defined below by their classifications. These will have associations with the File Object (Plant model) and possibly also between each other.

  Note: The Plant model always contains a Drawing object which has a mix of engineering and graphical attributes. Although it is not derived from either a PlantItem or AnnotationItem, it can be converted to an EIWM object, if for example, the "name" attribute is unique.

DEXPI Schema 4.0.1

- **Plant Item**: ActuatingFunction, ActuatingSystem, ActuatingSystemComponent, Component, Equipment, InformationFlow, InstrumentationLoopFunction, InstrumentComponent, InstrumentConnection, InstrumentLoop, MetaData, Nozzle, PipingComponent, PipingNetworkSegment, PipingNetworkSystem, PlantArea, PlantStructureItem, ProcessInstrument, ProcessInstrumentationFunction, ProcessSignalGeneratingFunction, ProcessSignalGeneratingSystem and ProcessSignalGeneratingSystemComponent.
- **AnnotationItem**: Symbol, SignalConnectorSymbol, ScopeBubble, PropertyBreak, PipeSlopeSymbol, PipeFlowArrow, PipeConnectorSymbol, Label and InsulationSymbol.

DEXPI Schema 3.3.3

- **Plant Item**: Equipment, InstrumentComponent, InstrumentConnection, InstrumentLoop, JunctionBox, Nozzle, PipingComponent, PipingNetworkSegment, PipingNetworkSystem, PlantArea, ProcessInstrument, SignalLine, Terminal and TerminalStrip.
- **AnnotationItem**: InsulationSymbol, Label, PipeConnectorSymbol, PipeFlowArrow, PropertyBreak, ScopeBubble and SignalConnectorSymbol.

The following is an example of a DEXPI Equipment Graphical Element:

```
<Equipment ID="XMP_58" TagName="B-9004" ComponentClass="Column" StockNumber="VCL"
ComponentName="VCL" ComponentClassURI="http://Test.org/rdl/1234" >
<Nozzle ID="XMP_59" TagName="N1" ComponentClass="Nozzle" StockNumber="FLNN1"
ComponentName="FLNN1" ComponentClassURI=" http://Test.org/rdl/1234" >
</Nozzle>
</Equipment>
```

and the representation of the above Equipment Element in the EIWM transformed as an object is as shown below:

```
<Object>
      <ID>XMP_58</ID>
      <Context>
        <ID>ContextId</ID>
      </Context>
      <ClassID>Column</ClassID>
      <Association type="Contains">
        <Object>
          <ID>XMP_59</ID>
          <Context>
            <ID>Avngate</ID>
          </Context>
          <ClassID>Nozzle</ClassID>
        </Object>
      </Association>
      <Characteristic>
        <Name>ID</Name>
        <Value>XMP_58</Value>
      </Characteristic>
      <Characteristic>
        <Name>TagName</Name>
        <Value>B-9004</Value>
      </Characteristic>
      <Characteristic>
        <Name>StockNumber</Name>
        <Value>VCL</Value>
      </Characteristic>
      <Characteristic>
        <Name>ComponentClass</Name>
        <Value>Column</Value>
      </Characteristic>
```

```
<Characteristic>
        <Name>ComponentClassURI</Name>
        <Value>http://Test.org/rdl/1234</Value>
      </Characteristic>
      <Characteristic>
        <Name>ComponentName</Name>
        <Value>VCL</Value>
      </Characteristic>
      <Characteristic>
        <Name>ComponentType</Name>
        <Value>Normal</Value>
      </Characteristic>
      <Characteristic>
        <Name>ComponentTypeSpecified</Name>
        <Value>False</Value>
      </Characteristic>
      <Characteristic>
        <Name>Status</Name>
        <Value>Current</Value>
      </Characteristic>
    </Object>
```

Note: The DEXPI node names or classes are not part of the attribute information. There is no attribute that can be used to identify the node name or class in the transformation. The extractor creates one custom attribute called "`ElementName`" and the type of this custom attribute is "`system attribute`". This `system attribute` can be used in the transformation to identify the type of the object and derive certain behaviour for each object type. But the ElementName attribute is not included in the attributes of the EIWM Object.

Example:

```
<Equipment ID="XMP_58" TagName="B-9004" ComponentClass="Column" StockNumber="VCL"
ComponentName="VCL" ComponentClassURI="http://Test.org/rdl/1234" >
```

From the above example, an `Equipment` object will be extracted with a system attribute called "`ElementName`" and its value is "`Equipment`". Transformation configuration can use this element name to assign this value to the object's class, or the you may prefer to use another attribute value, such as `ComponentClass`. The `Elementname` can also be used to identify the type of the object and derive `Object ID` or remove/create associations.

Identifying Characteristics and Properties

Any node in the DEXPI file has parameters defined by "`name`" and "`value`" pairs are converted to characteristics in the EIWM. Any parameters which also have a unit will be converted to properties in the EIWM.

Characteristics in the EIWM:

```
      <Characteristic>
        <Name>TagName</Name>
        <Value>B-9004</Value>
      </Characteristic>
```

Properties in the EIWM:

```
      <Property>
        <Name>MinimumDesignPressure</Name>
        <Value>4</Value>
        <Units>barg</Units>
      </Property>
      <Property>
        <Name>MaximumDesignPressure</Name>
        <Value>32</Value>
        <Units>barg</Units>
      </Property>
```

Identifying Associations

The DEXPI schema defines "`Association`" classes that contain attributes "`type`" and a referenced "`ItemID`". These are converted to associations in the EIWM file. Any pair of parent/child nodes will also have an association created between them in the EIWM.

For example:

```
<PlantStructureItem ID="PBS0D9331D91F4D4144B93903B01D5C7F4A" ComponentClass="ProcessPlant"
ComponentName="DEXPI" >
    <Association Type="is a collection including"
ItemID="PBS945900F29F424D32A292F07BB90792E2" />
    <Association Type="is a collection including"
ItemID="PBS21965A6820724688BB443B3EB454E19E" />
</PlantStructureItem>
```

Association in EIWM:

```
    <ID> PBS0D9331D91F4D4144B93903B01D5C7F4A </ID>
      <Context>
        <ID>ContextId</ID>
      </Context>
      <ClassID> PlantStructureItem </ClassID>
 <Association type="is a collection including">
     <Object>
          <ID> PBS945900F29F424D32A292F07BB90792E2</ID>
          <Context>
            <ID>ContextId</ID>
          </Context>
          <ClassID>PlantStructureItem</ClassID>
        </Object>
```

```
   </Association>
</Object>
```

Identifying the Datasets

The DEXPI schema defines a "`Genericattributes`" class. Objects of this `Genericattributes` class contain a group of attributes, which are converted to dataset objects in the EIWM. An association to its parent (containing generic attributes) is also created. These dataset objects have a system attribute named "`DataSetName`" with the value of the "`Set`" parameter, present as an attribute at the `Genericattributes` node level. The dataset Object ID can be derived from any one of the attributes or "`DataSetName`" with the combination of parent Object ID. This can be used in the transformation to derive the dataset's Object ID.

```
<PipingNetworkSegment ID="SGE48BB28FADEF449D8F373CB27FBD7123"
ComponentClass="PipingNetworkSegment" ComponentClassURI="http://data.posccaesar.org/rdl/
RDS267704" DualFlow="false">
 <GenericAttributes Number="2" Set="DexpiAttributes">
  <GenericAttribute Name="FluidCodeAssignmentClass" Value="QSb" Format="string"
AttributeURI="http://sandbox.dexpi.org/rdl/FluidCodeAssignmentClass" />
        <GenericAttribute Name="PipingClassCodeAssignmentClass" Value="75HB13"
Format="string" AttributeURI="http://sandbox.dexpi.org/rdl/
PipingClassCodeAssignmentClass" />    </GenericAttributes>
</ PipingNetworkSegment>
```

The equivalent Dataset Object in EIWM might be:

```
<Object>
      <ID> DexpiAttributes </ID>
      <Context>
          <ID> contextId </ID>
      </Context>
      <ClassID>DataSet</ClassID>
      <Characteristic>
        <Name> FluidCodeAssignmentClass </Name>
        <Value> QSb </Value>
      </Characteristic>
      <Characteristic>
        <Name PipingClassCodeAssignmentClass </Name>
        <Value> 75HB13</Value>
      </Characteristic>
    </Object>
```

Any attribute value that includes a unit of measure will be converted to property in the EIWM, using the unit type defined with the attribute value, for example,

```
<GenericAttribute
Name="InsulationThickness"
AttributeURI="http://data.posccaesar.org/rdl/RDS4238040"
Value="80"
Format="double"
Units="Millimetre"
UnitsURI="http://data.posccaesar.org/rdl/RDS1357739"/>
```

If the `Units` value is not present, then the default value defined in the DEXPI input file's header will be used, if relevant. If the `Units` value is not present or is not a correct value, then the relevant default values for the unit types, as part of the `unitOfMeasure` class defined in the schema, will be used. If a suitable value is not present then a warning will be logged and the attribute will be converted to a characteristic in the EIWM, instead of a property.

Note: Deriving characteristic values or units from `ValueURI` or `UnitsURI` is not supported.

The Gateway extracts the data from the DEXPI files in the form of Engineering and Graphics data, as listed below:

Engineering Objects	Graphics 2D Objects
ID (Required)	ID (with reference to Engineering Object)
TagName (Optional)	Presentation - RGB, color, Layer, Line Type, Line Weight information - (Required)
Persistent ID (Optional)	Extent - Bounding Box Information for the object (Required)
	Position/Orientation - Position of the object with respect to provided coordinate system (Required)
	Line, Circle, Trimmed Curve
	Font, Width, Height, String, Justification, Text Angle, Slant Angle
	NumPoints, FlowIn, FlowOut
	Coordinate - Lines / Polylines with minimum two numeric points
	Connection Points: To Indicate flow and connection information for the nodes between nozzle and equipment or any connection object (Required)
	Any graphics object which are defined to represent the graphics into XML

DEXPI File Structure

DEXPI file normally contains the following elements (nodes):

- Plant Model
- Process Instruments
- Drawing Object
- Piping Network System
- Shape Catalogue
- Drawing Graphics

All the above nodes of the DEXPI XML file get exported, by following DEXPI Schema.

The subsequent sections describe about those DEXPI File elements.

Plant Model

This node is a parent node in the DEXPI XML file and it holds the complete plant data with respect to the drawing:

- `Plant Information`: Plant information node contains information related to schema and drawing units. If this file is exported from any source system then, this node contains information about the originating system, module name, created or exported data time, discipline used while exporting /creating this file and so on.

- `PlantArea`: In the P&ID content of an XMpLant file, this is just an object that may be used to represent the name and attributes of named part(area) of a plant. A Plant Area is not permitted to contain model elements but may be referenced to associated segments, Equipment and so on with a named `PlantArea`. A `PlantArea` element inherits elements and attributes from the base type `PlantItem`.
- **Child Elements:**

Element Name	**Cardinality**
Component	0.*

- **Attributes:**

Attribute Name	**Required**	**Description**
Name	N	A string representing the name of the area. See Character encoding.

Example:

```
<PlantArea Name="Area-42">
       <Extent .. />
       <Association ../>
       <Association ../>
       <Association ../>
</PlantArea>
```

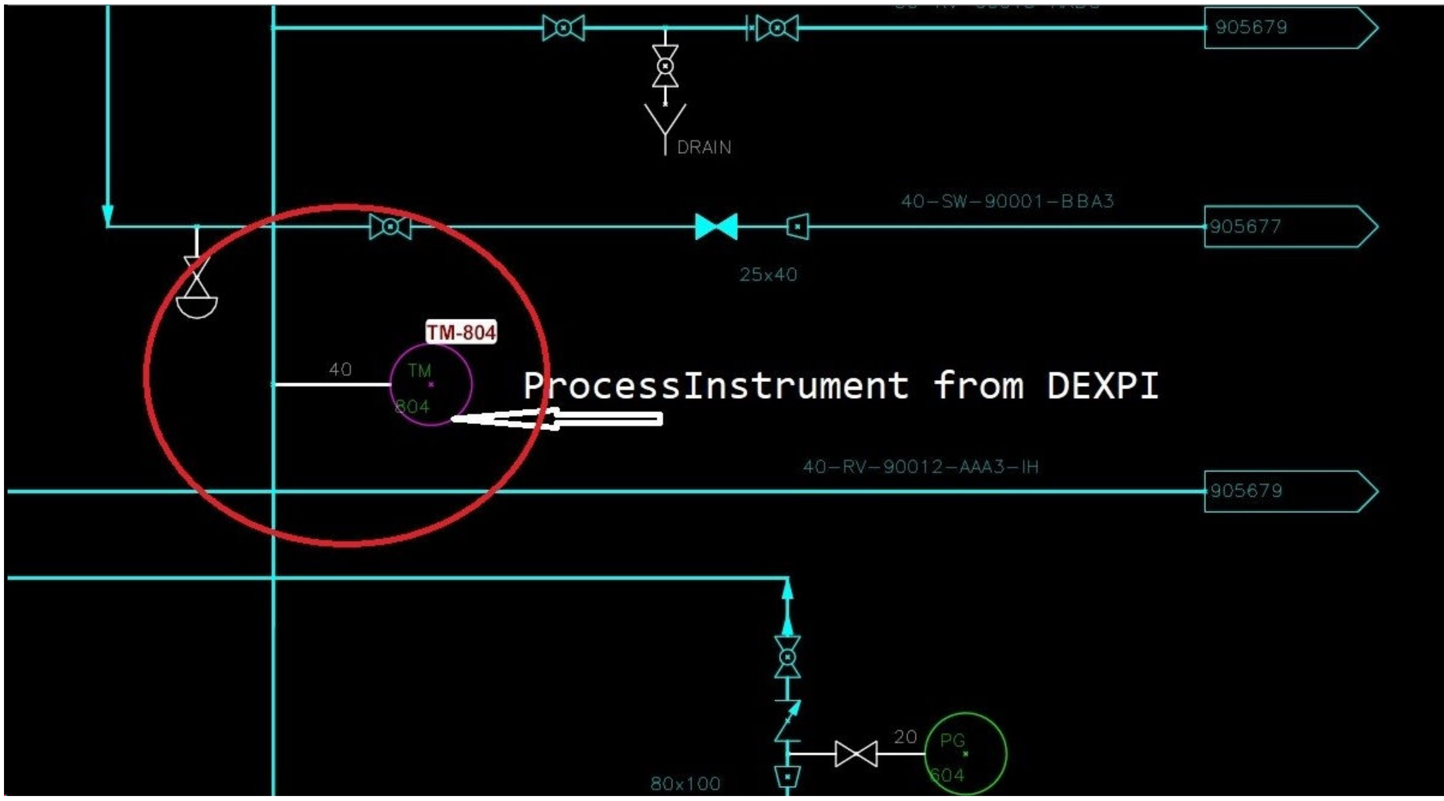
Process Instruments

- Process Instruments represent the physical instruments that are common to piping systems. These instruments inherit elements and attributes from the base type '**Plant Item**' (see schema 3.3.3 or 4.0.1). They contain information relating to components, diameter, operator type, wall thickness and process instrument elements. The Process Instrument from the DEXPI file usually contains two kinds of data:
- Engineering data - Includes name, ID, component class and possibly generic attributes.
- Graphics data - Defines the position, extents, presentation information and primitives such as line, circle and so on that represent the physical process instrumentation in the P&ID diagram.
- You can modify the engineering and graphics data by using mapping files. Using the transformation mappings, you can modify the engineering object data, attribute name and value, presentation, font types family and related graphics data as per requirement. For more information, refer to the mapping file from the configuration folder.
- The following is an example of a Process Instrument in a DEXPI file with respect to engineering and graphics data.

```
   <ProcessInstrument ID="ID_33" TagName="TM-804" ComponentClass="LevelTransmitter">
   <Extent>
   <Min X="710" Y="258" Z="0" />
   <Max X="726" Y="274" Z="0" />
   </Extent>
   <Position>
   <Location X="718" Y="266" Z="0" />
   <Axis X="0" Y="0" Z="1" />
```

```
<Reference X="1" Y="0" Z="0" />
</Position>
<Circle Radius="8" … />
<Text … />
<Text … />
<NominalDiameter … />
</ProcessInstrument>
```

This node will process in the SVG output as Process Instrument with tag TM-804 and with a nominal diameter 40, as shown below:



Drawing Object

Drawing Object node defines the drawing border and drawing border related text entities. This node may contain another drawing border node as well.

By default, a drawing object and its properties along with the text entities are exported into SVG format. The Engineering data is included as part of the Document object in the EIWM file. The following table lists the node elements under Drawing:

Node Element	**Description**
Drawing Border	(Graphics data) This element stores the drawing border/outline but this is at the discretion of the export tool.
Drawing Object Information	(Graphics data) This node contains metadata about the drawing and graphical annotation for the drawing object that the file contains, such as border related text entities and geometry. It might contain multiple drawing border nodes.

Node Element	Description
	The Red (R), Green (G) and Blue (B) values of this element define the background color for the drawing. This is in contrast to the normal use of the Presentation element that defines foreground style information. Their values can be changed in the presentation mapping configuration.
Extent	(Graphics data) Extents define the drawing area that contains all the graphics objects. Drawing extents cannot be used as an engineering object. It defines the minimum and maximum viewbox to the DEXPI geometry contents.
GenericAttributes	(Engineering data) Defines any application specific properties relating to the Drawing.
Name	(Engineering data) Defines the identity for the drawing, this must be unique in the project.
Type	(Engineering data) Defines the fixed value of PID.
Title	(Engineering data) Defines the drawing title.

Using the Color Mapping Method:

You can modify the colors of the output SVG drawing using presentation mapping, described in Presentation Mapping.

Piping Network System

A Piping Network System contains the information for a physical piping network system in the plant. Each piping network system contains many piping network segments.

Each piping network segment specifies a collection of components with common engineering properties that define a single process flow.

The following is the representation of connection for the piping components with connectors:

```
<PipingNetworkSystem>
<PipingNetworkSegment>
<Connection FromID="PID_17" ToID="PID_15" ToNode="2" FromNode="1" />
</PipingNetworkSegment>
<ConnectionPoints  NumPoints="3"  FlowIn="1"  FlowOut="2" >
<Node  ID="XMP_2717">
<Position>
<Location X="0.000000" Y="0.000000" Z="0.000000"/>
<Axis X="0.000000" Y="0.000000" Z="1.000000"/>
<Reference X="1.000000" Y="0.000000" Z="0.000000"/>
</Position>
</Node>
<Node  ID="XMP_2718" Type="process">
<Position>
<Location X="0.000000" Y="0.000000" Z="0.000000"/>
<Axis X="0.000000" Y="0.000000" Z="1.000000"/>
<Reference X="1.000000" Y="0.000000" Z="0.000000"/>
</Position>
</Node><Node  ID="XMP_2719" Type="process">
```

```
<Position>
<Location X="0.000000" Y="0.000000" Z="0.000000"/>
<Axis X="0.000000" Y="0.000000" Z="1.000000"/>
<Reference X="1.000000" Y="0.000000" Z="0.000000"/>
</Position>
</Node>
</ConnectionPoints>
</PipingNetworkSystem>
```

Note: There is one Connection object which connects two Pipingnetworksegment objects `PID_17` and `PID_15`, represented as an association in the EIWM between the two objects.

`FromID` and `ToID` are the piping connector / component / piping segment information and nodes are the potential flow connections between the parent object and the connection objects.

If necessary, attributes and `"connected to"` associations can be modified using standard transformation mapping configuration.

The following are the Piping classes in the DEXPI schema:

- PipingNetworkSystem
- PipingNetworkSegment
- PipingComponent
- PipeConnectorSymbol
- PipeFlowArrow
- PipeSlopeSymbol
- PropertyBreak
- ScopeBubble
- Symbol
- SignalConnectorSymbol
- CrossPageConnection
- Text

For more information, see the schema 3.3.3 and 4.0.1.

The following table describes some properties of Piping Network Systems and Piping Components.

Properties	**Description**
Piping Network System	
`PipingNetworkSystem ID="XMP_3"`	Unique ID for each object
`TagName="A-20"`	Actual Tag for each object
`ComponentClass="PipingSystem"`	Class for each object
`ComponentName`	This field is not mandatory or missing for network system
`Specification="A1A"`	Extra attribute to fill the object details

PipingComponent	
`ID="XMP_2035"`	Unique ID for each object
`ComponentName="SPCBLNOP"`	Unique name between Shape Catalogue and individual object from xml file
`ComponentClass="SpectacleBlindOpen"`	Component class information for the Piping Component
`Specification="A3B"`	Specification for the Piping Component

Extract "ConnectionPoints"

This represents connection between two objects. Connection node attributes "FromID" and "ToID" are the piping connector / component / piping segment information. In the following example, FromID(PID_17) and ToID(PID_15) are two Pipingnetworksegment objects, they are being added as association called "is connected to" to the object where the connection node is present. If necessary, attributes and "connected to" associations can be modified using the standard transformation mapping configuration.

```
<Connection FromID="PID_17" ToID="PID_15" ToNode="2" FromNode="1" />
```

Extract ConnectionPoints Node:

ConnectionPoints are a collection of connection nodes and each node can have a flow direction as defined by the parameters of the ConnectionPoints.

Node objects may be created as EIWM objects with an association to the ConnectionPoint's parent (for example, a Piping Component).

Each Node object contains two system attributes, that is, "ElementName", "NodeIndex" and one custom attribute called "ConnectionFlow". This will be included in the EIWM Node object attributes.

The NodeIndex will be assigned in the same order as the Node in the input file's ConnectionPoints and the "ConnectionFlow" value is derived from the FlowIn or FlowOut parameters of the ConnectionPoints. The Object ID of the node object can be derived with the combination of the parent ID and "NodeIndex".

For example:

```
<ConnectionPoints NumPoints="4" FlowIn="1" FlowOut="2" >
<Node>
  <NominalDiameter Value="25.000000" Units="mm" />
</Node>
<Node>
  <NominalDiameter Value="50.000000" Units="mm" />
</Node>
</ConnectionPoints>
```

where each value of the FlowIn or FlowOut parameter is the index of the relevant node object.

Cross Page Connector

When a piping object is connected to a component on another drawing, this will be indicated by the use of a Cross Page Connector. For example, the input drawing 905675.xml may contain the following cross page connector with respect to another drawing 905679.xml:

```
<PipeConnectorSymbol ID="XMP_155" ComponentClass="FlowOutPipeConnectorSymbol"
ComponentName="FLAG-OUT" >
<CrossPageConnection DrawingName="905679" LinkLabel="RV-90006" />
</PipeConnectorSymbol>
```

where the CrossPageConnection parameters define the name of the other drawing "905679" and a label for that connection "RV-90006", which may be the name of the component in the other drawing that it's connected to. This information can be represented in the output EIWM by a pipeconnectorsymbol object with an "is crosspage to" association to the other drawing 905679.

For example:

```
<Object>
      <ID>XMP_155</ID>
      <Context>
        <ID>Avngate</ID>
      </Context>
      <ClassID>FlowOutPipeConnectorSymbol</ClassID>
      <Association type="is crosspage to">
        <Object>
          <ID>905679</ID>
          <Context>
            <ID>Avngate</ID>
          </Context>
          <ClassID>P&amp;ID</ClassID>
        </Object>
      </Association>
      <Characteristic>
        <Name>ID</Name>
        <Value>XMP_155</Value>
      </Characteristic>
</Object>
```

The creation of this "is crosspage to" association is done in the transformation configuration, for example:

```
<Associations keepUnmappedAssociations="true">
   <Association>
      <Type value="is crosspage to" />
      <Conditions>
           <Attribute name="CrossPageConnectionDrawingName" />
      </Conditions>
      <TargetID value="[CrossPageConnectionDrawingName]">
      </TargetID>
      <TargetClassID value="P&amp;ID" />
    </Association>
</Associations>
```

In a PipeConnectorSymbol object, the cross page connector is connected to a CrossPageConnection object and connected drawing object. The is crosspage to parameter is used in the Cross Page Connector, which creates an association between PipeConnectorSymbol and CrossPageConnection.

Shape Catalogue

Shape Catalogue defines the following information in the drawing:

- Graphic information used in the drawing, such as the template definition for the graphics. It consists of a list of equipment, nozzle, piping component, and process instrumentation function.
- Extent information that can be used anywhere in the drawing along with the position factor and presentation, generic attributes and so on.
- The grouping of shapes to import, export and view the custom shapes. You can model complex custom shapes using your modelling software and import the shapes to your models.

Note: Standard symbols are used to represent the components in the diagrams. These symbols are not to scale and are not dimensionally accurate. The referred object consists of the actual positioning and dimensions. They are used to represent a certain type of component. These symbols are also labeled with words, letters, and numbers to further identify and specify the components that they represent. The diagrams do not represent the physical locations and proximity of each component in the Shape Catalogue, as the refereed component defines the physical location. The purpose is not to serve as an illustration of the system process.

```
ShapeCatalogue Name="Symbols/ Symbol Library /ShapeCatalogue"
```

All graphical data that is standardized by an ISO norm should be identical in all documents. The goal of the `ShapeCatalogue` is to outsource this graphical data to a common catalogue, to ensure unambiguous use of symbols and to support their re-usability.

For a correct reference to a symbol in the `ShapeCatalogue`, both elements are required to have the same `ComponentName` and to be of the same type (for example, `Equipment`, `PipingComponent`).

```
<ShapeCatalogue Name="Symbol Library" >
     <Nozzle ID="XMP_62" ComponentName="FLNN1" ComponentClass="Nozzle" >
          <PolyLine NumPoints="2">
              <Coordinate X="4.000000" Y="0.000000" Z="0.000000"/>
              <Coordinate X="0.000000" Y="0.000000" Z="0.000000"/>
         </PolyLine>
     </Nozzle>
</ShapeCatalogue>
<Nozzle ID="XMP_10" TagName="N2" ComponentClass="Nozzle" StockNumber="FLNN1"
ComponentName="FLNN1" ComponentClassURI="http://data.posccaesar.org/rdl/RDS415214" >
     <Extent>
         <Min X="478.646447" Y="625.500000" Z="0.000000"/>
         <Max X="485.000000" Y="630.500000" Z="0.000000"/>
     </Extent>
      <Presentation Layer="AS_EQU" LineType="Solid" LineWeight="0.25" colour="Lime" R="0"
G="1" B="0" />
      <Position>
          <Location X="485.000000" Y="628.000000" Z="0.000000"/>
          <Axis X="0.000000" Y="0.000000" Z="1.000000"/>
          <Reference X="-1.000000" Y="0.000000" Z="0.000000"/>
       </Position>
</Nozzle>
```

The graphic representations of symbols in the `ShapeCatalogue` element are stored exclusively in relative coordinates, meaning that the symbol's coordinates relate to the position (`x = 0`, `y = 0`, `z = 0`).

In addition, negative values in the `Extent` usually imply that the symbol's position, especially in symmetric symbols, is the point zero, and therefore, its coordinates are to be interpreted as relative.

A symbol from the `ShapeCatalogue` element, can be rotated or scaled by using `Reference` and `Scale`. The `Reference` is defined by the cosine and sine of the rotation angle.

Drawing Graphics

This node contains all the graphics that are present or defined to represent the actual Plant Model. Graphics indicates the Presentation, extent, position, list of text entities to represent the engineering information, list of generic attributes to define the intelligent information to represent the graphics as engineering element and list of other supporting graphics to represent it completely.

- `Presentation`: Specifies the layer name, color information along with R, G and B values, line type and line weight information.
- `Extent`: Defines the drawing extent, the minimum and maximum values of XYZ coordinates.
- `Position`: Defines the exact location of the graphics object in the drawing. This node contains following three child nodes:
    - `Location`: Defines the exact position of the graphics in the drawing.
    - `Axis`: Defines the information of what axis the graphics element is respect to the drawing coordinates.
    - `Reference`: Defines the information of rotation angle with respect to the drawing coordinates.

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 the following transform extensions:

- `Patterns 2D`: Allows you to select, via `Conditions`, which type of graphical objects to amend so that tags can be associated or disassociated with graphical elements using the `Group` and `Ungroup` functions and add new `Attributes` to the grouped and ungrouped objects. For more information, see Patterns 2D ().
- `Base Mapping`: Modifies the engineering data of a model's objects present in the extracted data. For more information, see Base Mapping ().
- `Presentation Mapping`: Adjusts the output graphics properties, such as materials and color. For more information, see Presentation Mapping ().
- `Text Mapping`: Used in 2D drawings to customize textual objects visible in the drawing, either by varying the font that is used or mapping specific characters/text strings to alternative values. For more information, see Text Mapping ().
- `Rescale 2D Transformation`: Enables you to adjust/scale the SVG graphics, OLE object resolution and line thickness. This may be needed when the drawing's resolution is not matched with the resolution of the monitor or printer where the SVG will be viewed. For more information, see Rescale 2D ().
- `CSV Reporting`: 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 ().

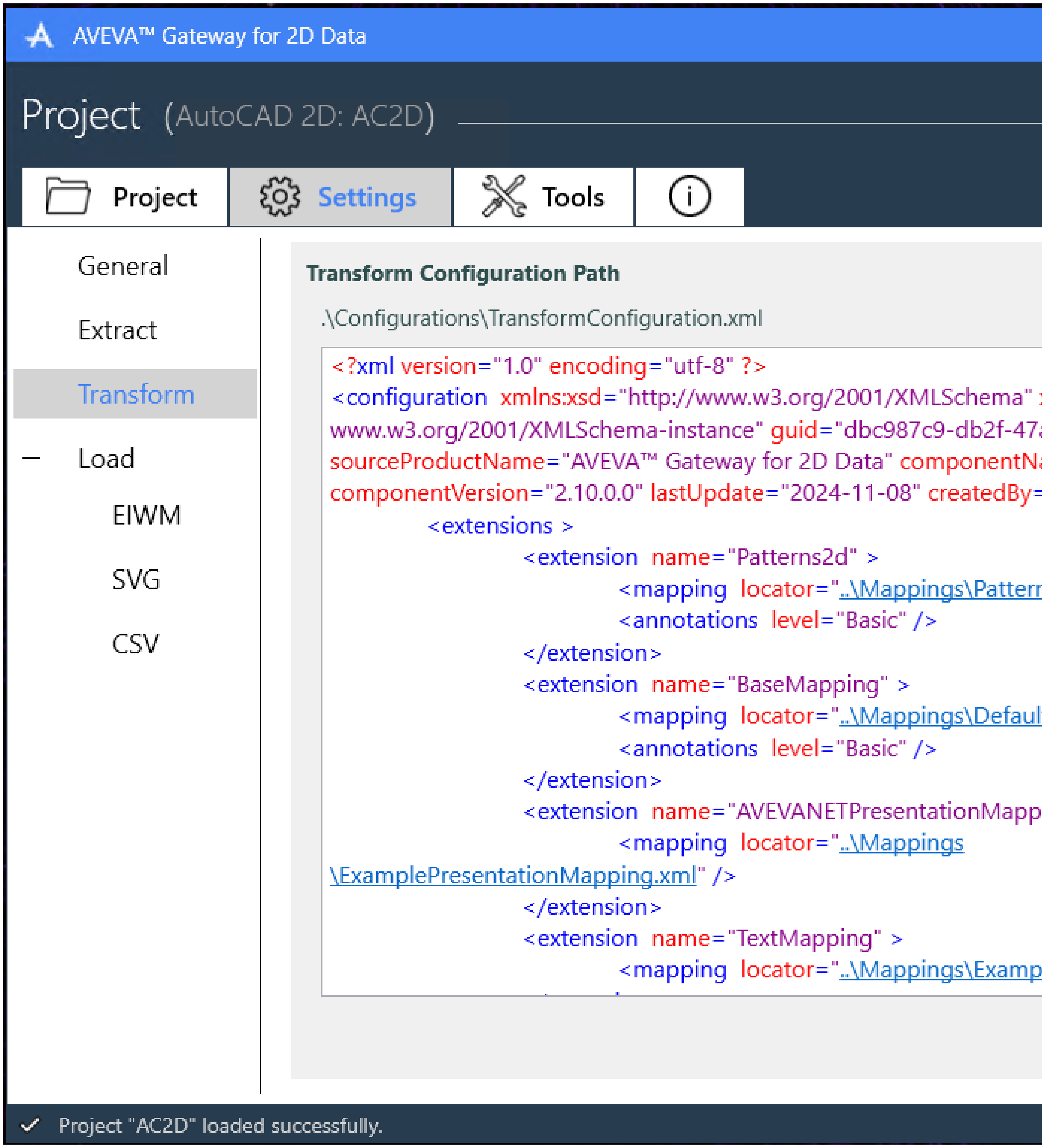
Notes:

- 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 it**s mappi**ng configurations.

The **Transform** panel 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** panel in the **Project Settings** page to open the **Transform** details page.



Transform Configuration Path

.\Configurations\TransformConfiguration.xml

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration xmlns:xsd="http://www.w3.org/2001/XMLSchema"
www.w3.org/2001/XMLSchema-instance" guid="dbc987c9-db2f-47
sourceProductName="AVEVA™ Gateway for 2D Data" componentN
componentVersion="2.10.0.0" lastUpdate="2024-11-08" createdBy=
        <extensions >
                <extension name="Patterns2d" >
                        <mapping locator="..\Mappings\Pattern
                        <annotations level="Basic" />
                </extension>
                <extension name="BaseMapping" >
                        <mapping locator="..\Mappings\Defaul
                        <annotations level="Basic" />
                </extension>
                <extension name="AVEVANETPresentationMapp
                        <mapping locator="..\Mappings
\ExamplePresentationMapping.xml" />
                </extension>
                <extension name="TextMapping" >
                        <mapping locator="..\Mappings\Examp
```

✓ Project "AC2D" loaded successfully.

2. Click Edit to open the settings in a **Notepad** to 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.

4. After all the settings have been saved successfully, click **Run** to process the input file(s).

Notes:

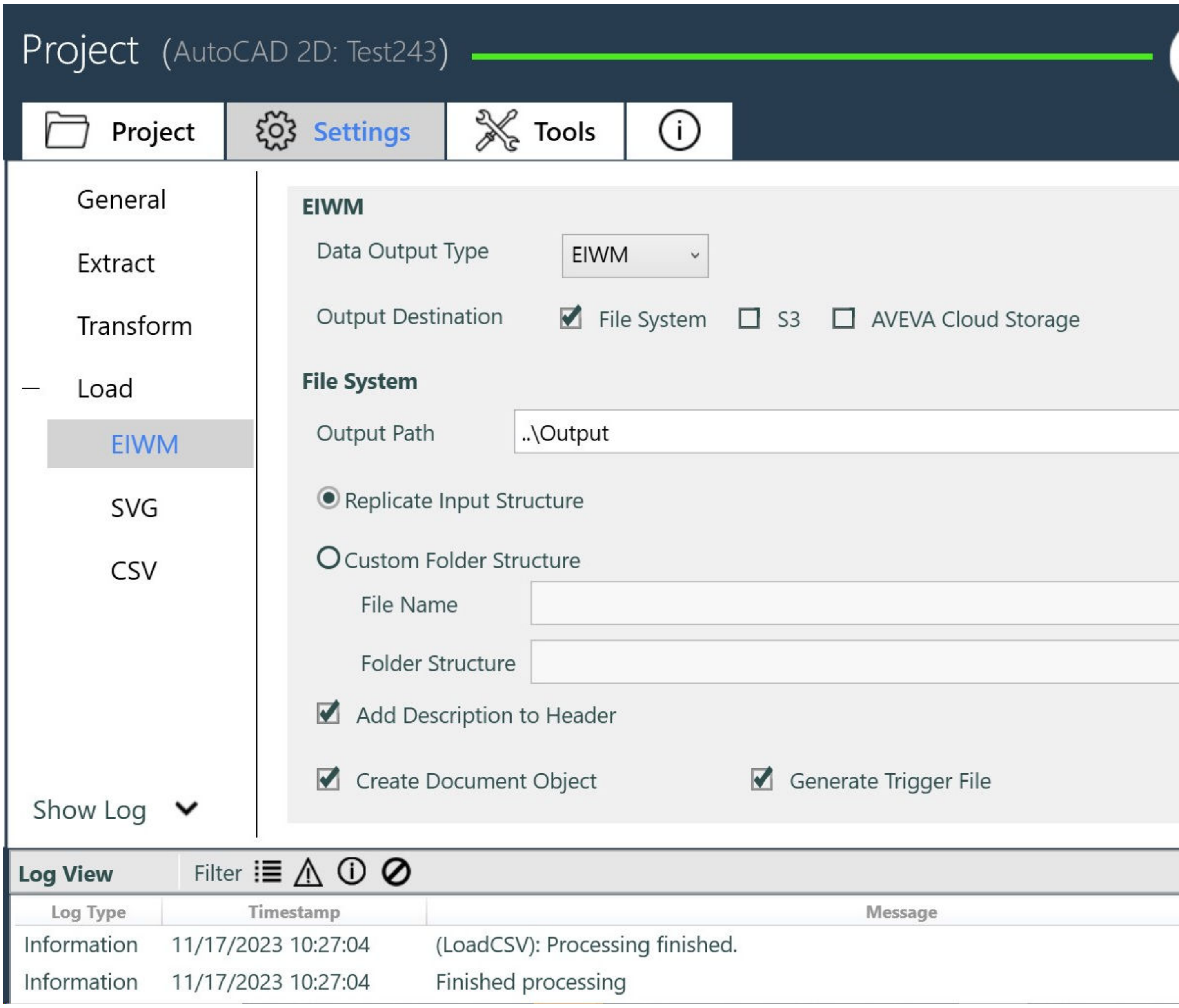
- Any update in the configuration document must be saved before running the process.
- Set the tracking of changes provided by each Transform extension using annotations feature, as shown in the above XML configuration. For more information about annotations, see Appendix C: Tracking Changes in Data.
- 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. The output folder must be in the File System, not an S3 Bucket nor an AVEVA Cloud Storage folder.
- For more information about handling system attributes, see Appendix H: 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 SVG 2D model representation. The Load component also exports the transformed data into various other CSV files.

Load 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.



EIWM

Data Output Type: EIWM

Output Destination: [✓ File System] S3 AVEVA Cloud Storage

File System

Output Path: ..\Output

[✓ Replicate Input Structure]

Custom Folder Structure

File Name:

Folder Structure:

[✓ Add Description to Header]

[✓ Create Document Object] [✓ Generate Trigger File]

Log Type	Timestamp	Message
Information	11/17/2023 10:27:04	(LoadCSV): Processing finished.
Information	11/17/2023 10:27:04	Finished processing

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. 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 **S3** or an **AVEVA Cloud Storage** folder, or any combination of these. Configure this in the Load EIWM Configuration file using the OutputDestination element, for example, `<OutputDestination Destination="FileSystem">`.

  Notes:

  - The `Destination` values can be a case-insensitive representation of either "FileSystem" or "S3", or "Both".
  - **File System Details**: Under **Output** Path option, type the output file path or click the **Browse for Folder** icon to search for the output folder path.
  - **S3**: For more information while using S3 as the output destination, see [Access an AWS S3 Bucket]() ().
  - **AVEVA Cloud Storage**: For more information while using ACS as the output destination, see [Access AVEVA Cloud Storage]().

- **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 each input file in that structure will be in the corresponding output sub-folders. 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.

  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** box can be the same as defined in the **Folder Structure**. If **File Suffix** setting is blank (default) then the file's suffix is determined by the project context:

  - A project context of `avngate` will set the suffix to "`_avngate.xml`".
  - A project context set to any value other than `avngate` will set the suffix to "`_null.xml`".

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`, assuming that `File Suffix` setting was blank and **Project Context** was set to `avngate`.

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`, assuming that **File Suffix** setting was also blank.

- 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>
```

This 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

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 AVEVA NET 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	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. **Note**: The nested context must be resolved as separate sub folders in the hierarchy.
EIWM reserved attributes	EIWM reserved attributes can be used as elements in the folder structure:

Expected Entries in a Folder Structure	Description
	`[eiwm:id]`, `[eiwm:classid]`, `[eiwm:name]`, `[eiwm:revision]` or `[eiwm:context]`
Manifest attributes	For more information about manifest attributes, see Manifest References ().
Reserved placeholders	Reserved placeholders can be used as elements in the folder structure: • `[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}]`. **Note**: The formats can be from any of the formats as specified under the MSDN DateTime Formats.
Static values	The static values are simple text values and are left unchanged and can be defined without any square brackets. Example: `<FileName value="output_[ eiwm:id].xml" />`. 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 is as follows.

  ```
  <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: It is possible to modify the same setting from both configuration and the EIWM Loader GUI page.

- **Project Context**: **By default, it is set to `Avngate`. It controls the EIWM object's output context, also it affects EIWM files suffix when File Suffix is left blank. It can be set to empty resulting in EIWM objects generated with empty context, unless it's defined in base mapping.**
- **File Suffix**: By default, it is set to empty, it controls the generated EIWM files suffixes if not left blank.

  This allows for pre-processing by the AVEVA AIM Import Controller to overwrite `Avngate` with any Vnet file definition for the project context (refer to the AVEVA AIM User Guide). Characters such as '/', ':', '*', '?', '"', '<', '>' and '|' are not allowed to be used in `File Suffix` setting.
- **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. 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**, 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:

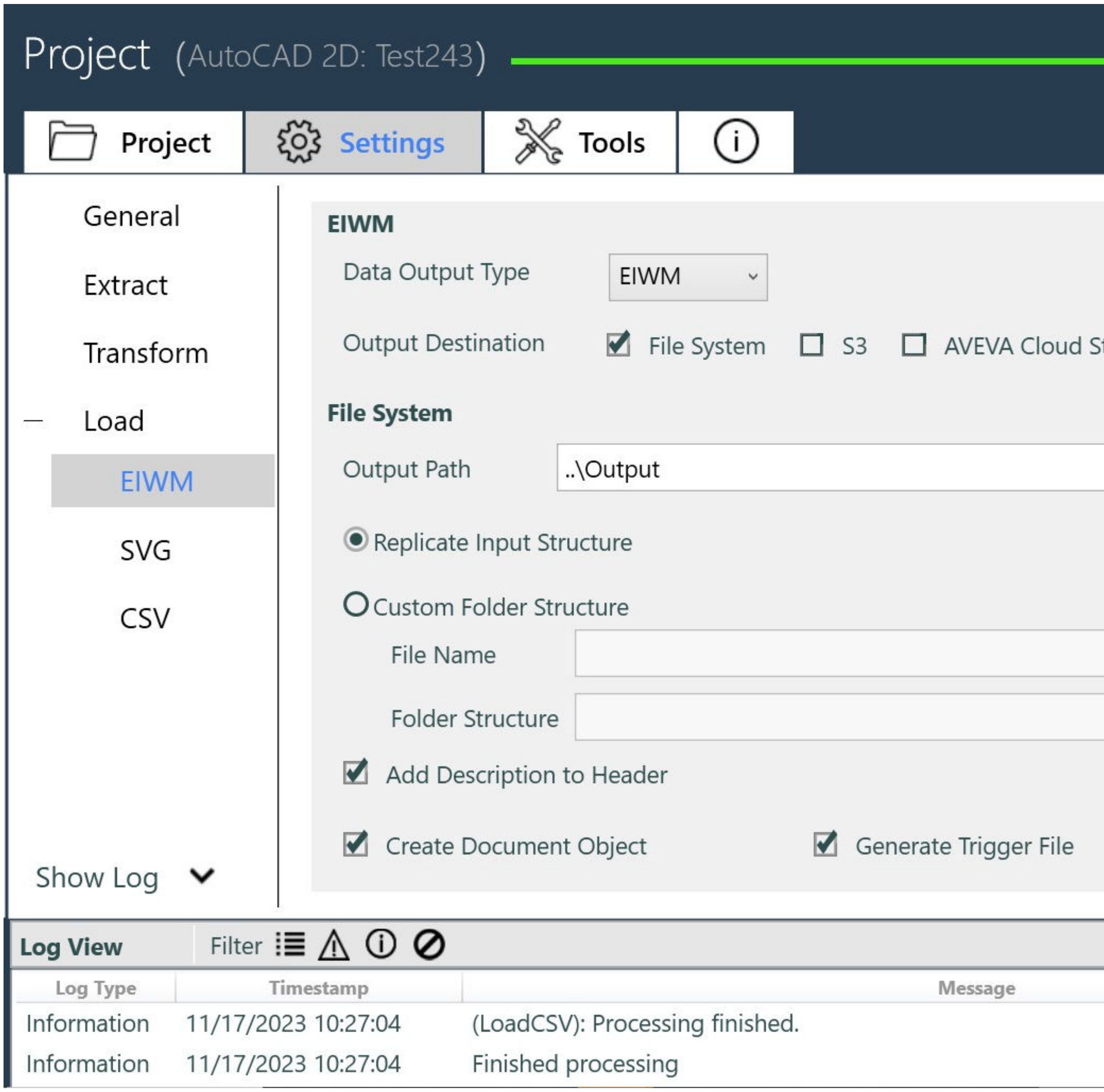
```
   <outputEIWM>
                .........
                <createAlias apply="true" attribute ="GlobalID" />
                ..........
      </outputEIWM>
```

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>
```



EIWM

Data Output Type: EIWM

Output Destination: [✓ File System] S3 AVEVA Cloud S

File System

Output Path: ..\Output

[✓ Replicate Input Structure]

Custom Folder Structure

File Name

Folder Structure

[✓ Add Description to Header]

[✓ Create Document Object] [✓ Generate Trigger File]

Log Type	Timestamp	Message
Information	11/17/2023 10:27:04	(LoadCSV): Processing finished.
Information	11/17/2023 10:27:04	Finished processing

- **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™ Gateway for
   2D Data" componentName="Load" componentVersion="2.10.0.0" >
     <components>
       <component name="LoadEIWM" locator=".\LoadEIWMConfiguration.xml" />
       <component name="LoadSVG" locator=".\LoadSVGConfiguration.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™ Gateway for 2D Data"
componentName="LoadEIWM" componentVersion="2.10.0.0" >
    <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>
    <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 need to use 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" contextSuffix="" />
```

Modify the output file formatting as follows:

- **Context Suffix:** Set the EIWM file name's suffix to be appended to the end of the file name. Set this to empty, in which case the EIWM file name suffix will be set to `_avngate` or `_null`, depending on the `projectContext` value.
- **Splitting Output File**: 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**: 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

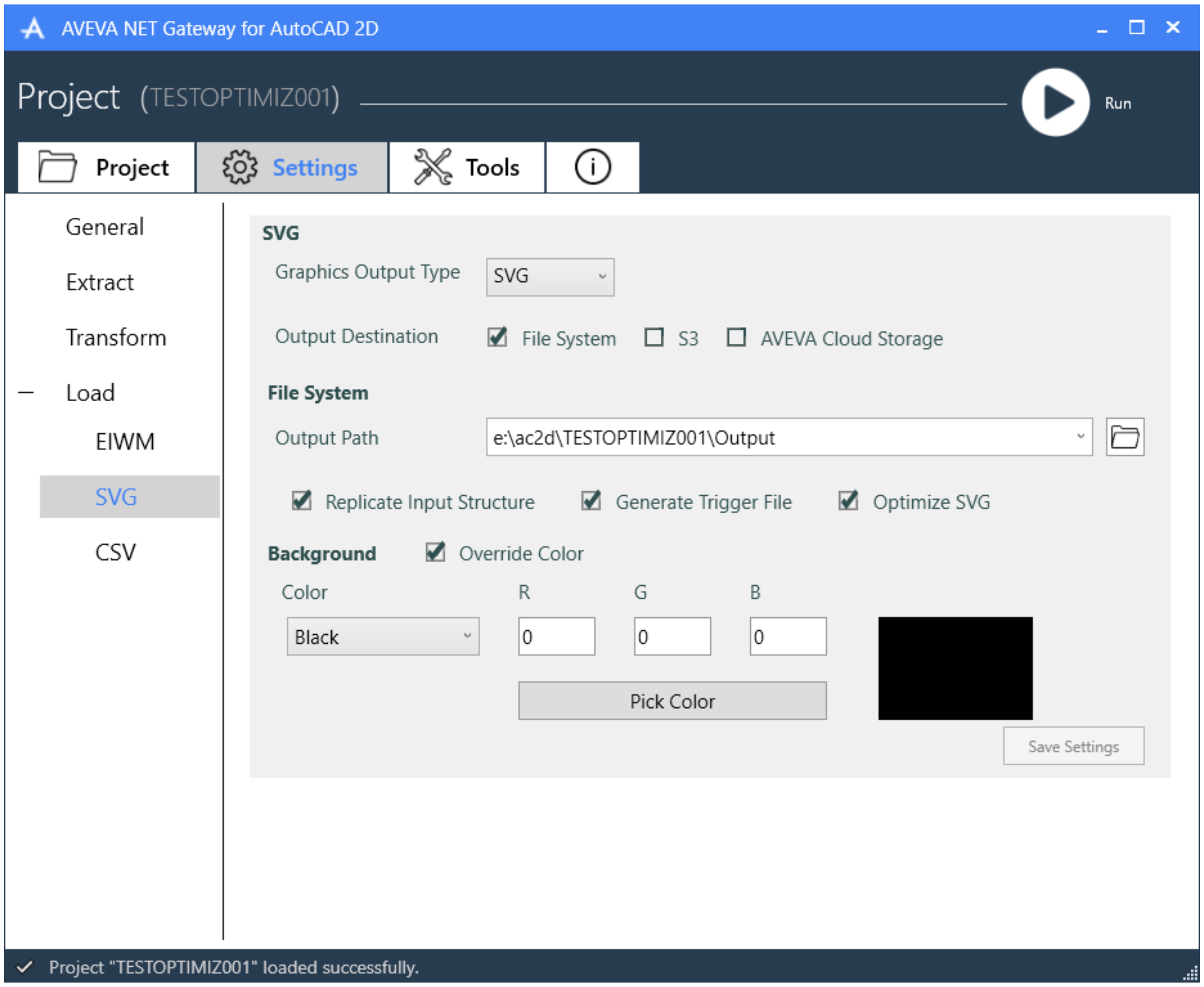
the multiple template mapped to the EIWM objects. This setting can be modified using `<file ... suffix="_part_"` attribute in the loader configuration setting. The output for above-defined setting is as follows:

- `Wall_part_001_null.xml`
- `Wall_part_002_null.xml`

- **Suffix number format**: 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`

Load SVG

Load exports graphical data together with metadata to SVG format. Select the graphical data output type SVG format under the **Load, SVG** settings.



SVG

Graphics Output Type: SVG

Output Destination: [✓ File System] S3 AVEVA Cloud Storage

File System

Output Path: e:\ac2d\TESTOPTIMIZ001\Output

[✓ Replicate Input Structure] [✓ Generate Trigger File] [✓ Optimize SVG]

Background [✓ Override Color]

Color: Black R: 0 G: 0 B: 0

Project "TESTOPTIMIZ001" loaded successfully.

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

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

- **Graphics Output Type**: Select the output file type for storing the graphics, such as SVG (XML-based two-dimensional format). Use **None** option if no 2D 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 **AWS** or **Both**.
  - **File System Details**:
    - **Output Path**: Type the Output file path or click the **Browse for Folder** panel to enter the output folder path.
  - **S3 Bucket Details:** 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 ().
- **Replicate Input Structure**: Select this option if the input folder contains subfolders and you want the relevant output files to be located in the same subfolders of the output location.
- **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. The `generateTriggerStart` option is used to generate a trigger start file which directs the AIM Import Controller to start importing the output files in to AIM. The trigger file will be created after all processing has completed, if no errors have occurred. Configure this using the `generateTriggerStart` element, for example, `<generateTriggerStart apply="true" />`.

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.

- **Optimize SVG**: This option restructures the SVG without changing the SVG appearance, which usually reduces its size and rendering time, but does increase the Gateway processing time.
- **Background:** Select the background color of drawings. The background color of the **SVG** files generated by the project can be selected by any of these methods:
  - **Override Color**: Some of the input drawings used for creating the SVG may or may not contain information about the background colour. If a background colour is specified in the input drawing, set the SVG's background to the same colour, or use a different background colour by setting the **Override Colour** option.
  - **Color**: Select the required color from the Color list, for example, Black.
  - **R**, **G**, **B**: Enter the **RGB** values of the color in the fields provided.
  - **Pick Color**: Click **Pick Color** and select the required color from the standard color selection dialog. A preview of the selected color is displayed next to the **Pick Color** button.

  Notes:

  - If any graphic or text has the same color as the background color, then the color of this item is inverted. For example, if the background color is Black and any graphics or texts have source color of Black, then these items will be converted to White in the output SVG.

- Raster images are normally exported in their source colors. Bitonal (two colors) raster images are exceptions as they have two colors:
- Primary: If the image's Primary color is the same as the SVG Background Color, then it is changed to be visible against the new background color (this is similar to the behavior of AutoCAD application).
- Background: If the input image has a transparent background, then the image's background in the output SVG is also transparent, otherwise, the image's background color is set to the same background color as configured for the SVG.

- **Save Settings**: Click **Save Settings** to save the settings.

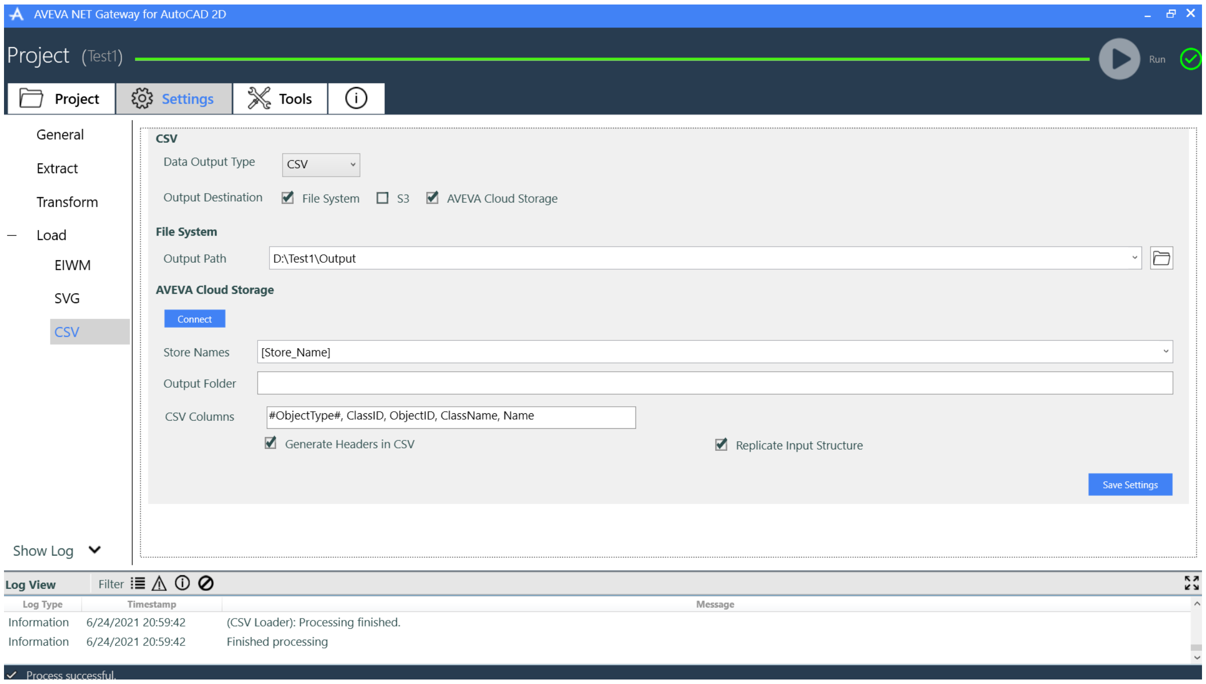
Notes:

- Set the tracking of changes provided by SVG Load on data using annotations feature. For information about annotations, see Appendix C: Tracking Changes in Data.
- Exporting 'wipeout frames' from DWG to SVG is dependent on the global '`WIPEOUTFRAME`' variable in the drawing:
  - Values '0' and '2' mean that frames are not written to SVG.
  - Value '1' means that frames are written to SVG.
- When the input DWG/DXF document contains a wipeout object, then tag objects metadata grouped by Patterns 2D are assigned separately to each member of the SVG group, rather than to the single grouped object. This is to ensure correct rendering of the SVG group members where those behind the wipeout object are not visible or only partially visible.

  This causes each member of the group to be highlighted when the mouse hovers over each, but when any one of them is selected then all members are selected with the one single tag.

Load CSV

This option enables you to configure the **CSV** settings.



CSV
Data Output Type: CSV
Output Destination: ☑ File System ☐ S3 ☑ AVEVA Cloud Storage
File System
Output Path: D:\Test1\Output
AVEVA Cloud Storage
Store Names: [Store_Name]
Output Folder:
CSV Columns: #ObjectType#, ClassID, ObjectID, ClassName, Name
☑ Generate Headers in CSV
☑ Replicate Input Structure

```
Log Type     Timestamp            Message
Information  6/24/2021 20:59:42   (CSV Loader): Processing finished.
Information  6/24/2021 20:59:42   Finished processing
```
✓ Process successful.

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**, **S3** or **AVEVA Cloud Storage**.
  - **File System Details**:
    - **Output Path**: Type the Output file path or click **Browse for Folder** button to enter the output file path.
  - **S3 Bucket Details:** For more information, see Accessing an AWS S3 Bucket ().
- **Create Tag List CSV Report**: If you want to get a CSV file report containing a list of all the tagged objects and classifications in the output folder, select **Create Tag List CSV Report**.
- **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 files 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 need to execute the project. While executing the project configuration, the Gateway searches for the input location and generates output files.

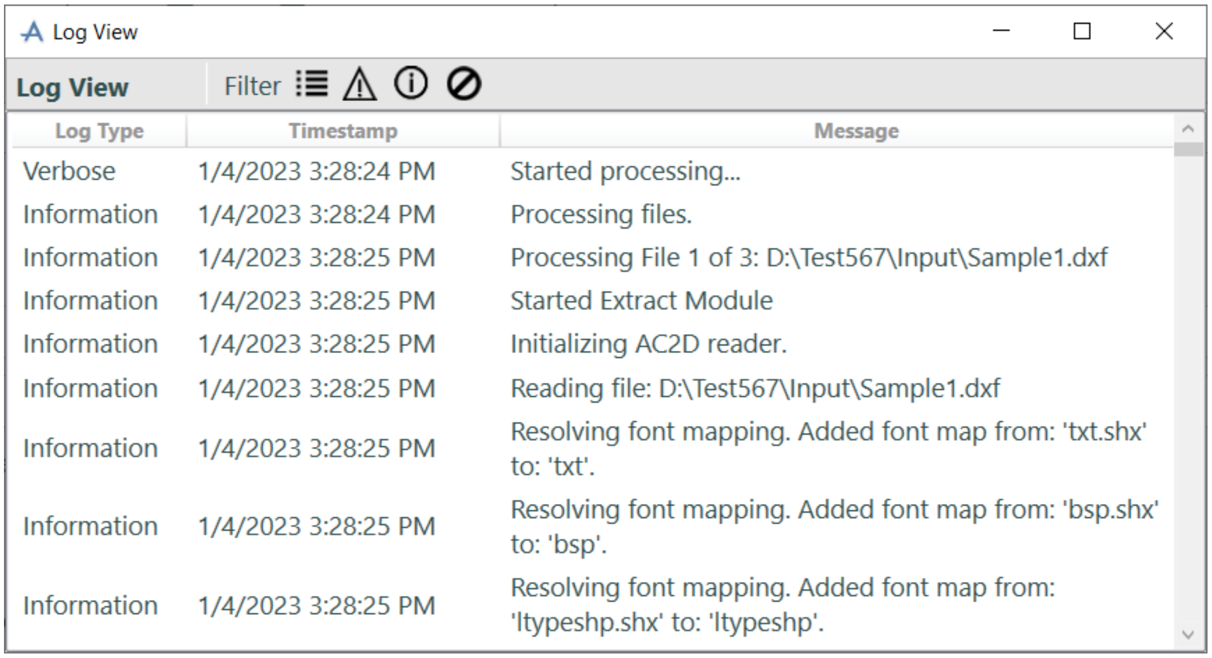
To execute the selected project:

1. Click **Save Settings** to save the configuration settings of **Extract**, **Transform** and **Load**.

   After the successful saving of the configuration files, it enables the **Run** button.
2. Click **Run** to execute the project.

Note: The input files 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
Verbose	1/4/2023 3:28:24 PM	Started processing...
Information	1/4/2023 3:28:24 PM	Processing files.
Information	1/4/2023 3:28:25 PM	Processing File 1 of 3: D:\Test567\Input\Sample1.dxf
Information	1/4/2023 3:28:25 PM	Started Extract Module
Information	1/4/2023 3:28:25 PM	Initializing AC2D reader.
Information	1/4/2023 3:28:25 PM	Reading file: D:\Test567\Input\Sample1.dxf
Information	1/4/2023 3:28:25 PM	Resolving font mapping. Added font map from: 'txt.shx' to: 'txt'.
Information	1/4/2023 3:28:25 PM	Resolving font mapping. Added font map from: 'bsp.shx' to: 'bsp'.
Information	1/4/2023 3:28:25 PM	Resolving font mapping. Added font map from: 'ltypeshp.shx' to: 'ltypeshp'.

Notes:

- 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.
- 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

If you open the old project versions containing out-of-date configuration, 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, unless the upgradedProjectLocation option is used to specify a different target directory for the upgraded configuration, see Run the Gateway from the Command Line. 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.

Limitations:

- The Gateway does not support segmentation, so it cannot migrate the Gateway Configuration Tool configuration that specifies a ClippingBox.
- Both `prefixqualifierclassification` and `suffixqualifierclassification` parameters are not converted. Warning is logged for each pattern that contains these parameters.

Access an AWS S3 Bucket

An S3 Bucket is a container for objects stored in Amazon 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 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. 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 n**ame (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//SVG/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*
```

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

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

- 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.

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

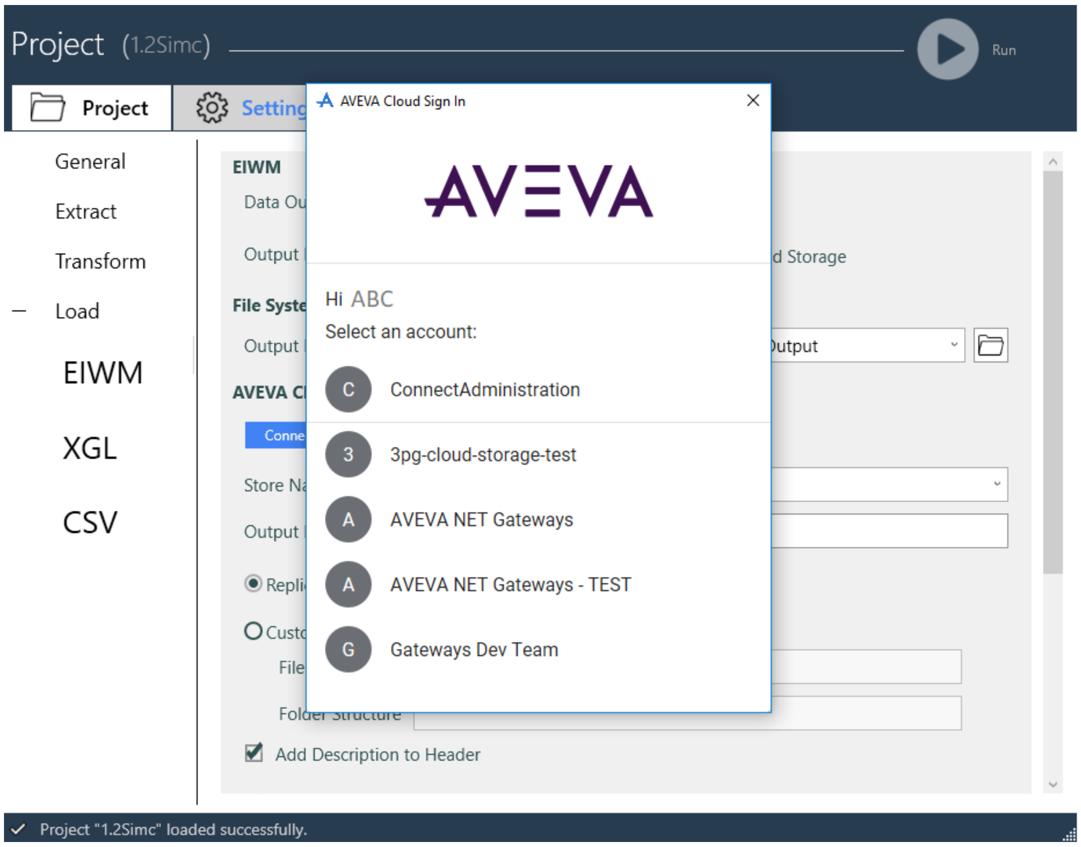
Note: The above-listed environment variables are mandatory for connecting to ACS.

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.2DData.ACS_credentials.



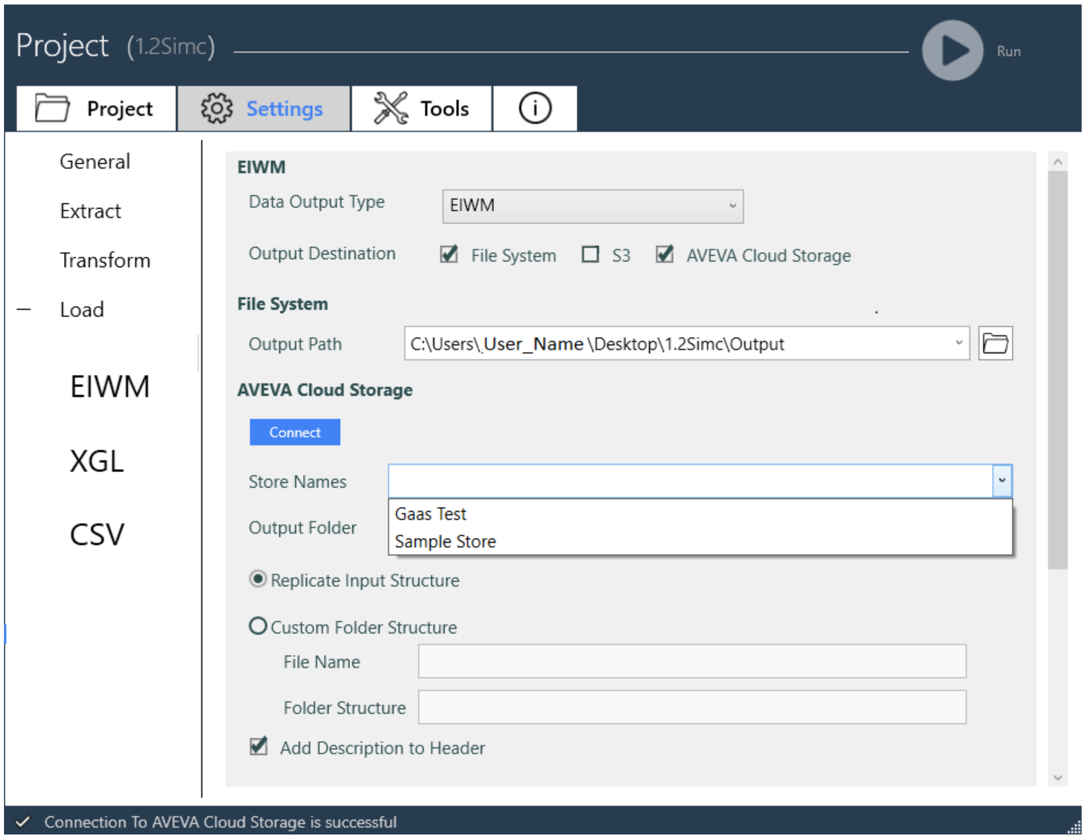
Project "1.2Simc" loaded successfully.

1. **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.



EIWM

Data Output Type: EIWM

Output Destination: [✓ File System] ☐ S3 [✓ AVEVA Cloud Storage]

File System

Output Path: C:\Users\User_Name\Desktop\1.2Simc\Output

AVEVA Cloud Storage

Store Names: Gaas Test, Sample Store

Output Folder

[✓ Replicate Input Structure] Custom Folder Structure

File Name

Folder Structure

[✓ Add Description to Header]

Connection To AVEVA Cloud Storage is successful

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

The Loaders configuration files such as EIWM, SVG, 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="EIWM" />
<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:

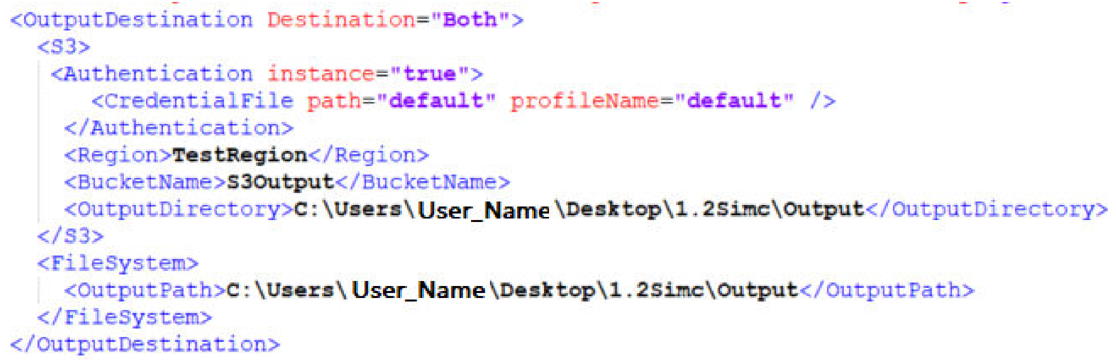
- If you select only the FileSystem, the loader configuration will be as follows:

```
<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: 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

You can run the Gateway from the command line.

Note: 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 and without waiting for the Gateway to complete the execution.

All the logs before project execution (such as command line parameters validation logs or project configuration file load logs) are logged into the default log location:

```
C:\Users\<username>\AppData\Roaming\AVEVA\AVEVA Gateway for 2D
Data\Logs\2DDataGateway_Log_<GUID>.txt
```

The logs during project execution are logged separately to the log folder as defined by you either from command prompt or in the configuration file.

Note: A log file is generated for each .DWG or .DXF file.

At the command line, type the following:

```
"C:\Program Files\AVEVA\AVEVA NET Gateways\Gateway For 2D
Data\AVEVA.NET.Gateways.2DData.exe" <-cp> <-in> <-op> <-lp> <-extcp> <-trnsfcp> <-ldrcp> <-
context> <-upgradedProjectLocation> <-vf> <-vsviewports style:resolution> <-vsblock
name:imageLocation> <-VSHATCH inputFillPattern:maxScale:outputFillPattern>
```

where the parameters are listed as follows:

Command Line Parameter	Description
Common Parameters	
`-cp <project configuration file path>`	(mandatory) Project configuration file path, for example, "`C:\Doc Gateway Training\Example Config.xml`".
`-in <input file/folder path>`	(optional) Input file/folder path for the input files. **Note**: This option cannot be used to select the input present in S3 bucket or in ACS.
`-op <output folder path>`	(optional) Output folder path for all the the output EIWM/SVG/CSV files.
`-lp <log folder path>`	(optional) Log folder path.
`-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.

Command Line Parameter	Description
`-ma "<manifest attribute name1>:<manifest attribute value1>"`	(optional) Allows the passing of a manifest attribute value from the command line. Specify any number of manifest attributes with this syntax through command line.
`-ma <manifest attribute name2>:<manifest attribute value2>`	(optional) Allows the passing of a manifest attribute value from the command line.
`-context <project context>`	(optional) Sets the context for EIWM objects.
`-upgradedProjectLocation <folder path for upgraded Gateway Configuration Tool project>`	(optional) Sets the new folder location for the upgraded Gateway Configuration Tool project.
Parameters Related to AC2D	
`-vf <viewFilter>`	(optional) Layout selection filter. See Layout Selection filter syntax in Section 4.3.2.
`-vsviewports style:resolution <Viewport style>`	'style' values: mixed, wireframe, raster 'resolution' values (optional, default = 1): 0.1 - 10 short param name: -VSV
`-VSBLOCK name:imageLocation <Block style>`	'name' value: block name 'imageLocation' value: image location that replaces block graphics short param name: -VSB
Parameters Related to AC2D and MS2D	
`-VSHATCH inputFillPattern:maxScale:outputFillPattern <Hatches Style>`	`inputFillPattern value`: hatch name `maxScale value`: hatch scale property `outputFillPattern value`: only 'SOLID' available `short param name`: -VSH

Notes:

- Any optional parameter used at the command line takes priority over the equivalent configuration in the project configuration file. This is done without overwriting the value in project configuration file.
- 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:

With only project configuration file: `"C:\Program Files\AVEVA\AVEVA NET Gateways\Gateway For 2D Data \`AVEVA.NET.Gateways.2DData.exe" `-cp "D:\Test\New_1\New_1.xml"`

With project configuration file and input folder: `"C:\Program Files\AVEVA\AVEVA NET Gateways\Gateway For 2D Data\`AVEVA.NET.Gateways.2DData.exe" `-cp "D:\Test\New_1\New_1.xml" -in "D:\Test\New_2\Input" Takes Input source files from command line parameter "D:\Test\New_2\Input"`

With project configuration file and output folder: `"C:\Program Files\AVEVA\AVEVA NET Gateways\Gateway For 2D Data\`AVEVA.NET.Gateways.2DData.exe" `-cp "D:\Test\New_1\New_1.xml" -op "D:\Test\New_2\Output" Generates output in the Output Path provided from command line "D:\Test\New_2\Output"`

With project configuration file and logs folder: `"C:\Program Files\AVEVA\AVEVA NET Gateways\Gateway For 2D Data\`AVEVA.NET.Gateways.2DData.exe" `-cp "D:\Test\New_1\New_1.xml" -op "D:\Test\Logs" Generates logs in the Logs Path provided from command line "D:\Test\Logs"`

With project configuration file and Extract configuration file: `"C:\Program Files\AVEVA\AVEVA NET Gateways\Gateway For 2D Data\\`AVEVA.NET.Gateways.2DData.exe" `-cp "D:\Test\New_1\New_1.xml" -extcp "D:\Test\New_2\Configurations\ExtractorConfiguration.xml" Takes Extractor Configuration file from command line parameter "D:\Test\New_1\New_1.xml" -extcp "D:\Test\New_2\Configurations\ExtractorConfiguration.xml"`

With project configuration file and Transform configuration file: `"C:\Program Files\AVEVA\AVEVA NET Gateways\Gateway For 2D Data\`AVEVA.NET.Gateways.2DData.exe" `-cp "D:\Test\New_1\New_1.xml" -trnsfcp "D:\Test\New_2\Configurations\TransformerConfiguration.xml" Takes Transformer Configuration file from command line parameter "D:\Test\New_1\New_1.xml" -extcp "D:\Test\New_2\Configurations\TransformerConfiguration.xml"`

With project configuration file and Load configuration file: `"C:\Program Files\AVEVA\AVEVA NET Gateways\Gateway For 2D Data\`AVEVA.NET.Gateways.2DData.exe" `-cp "D:\Test\New_1\New_1.xml" -ldrcp "D:\Test\New_2\Configurations\LoaderConfiguration.xml" Takes Loader Configuration file from command line parameter "D:\Test\New_1\New_1.xml" -extcp "D:\Test\New_2\Configurations\LoaderConfiguration.xml"`

With project configuration file and project context: `"C:\Program Files\AVEVA\AVEVA NET Gateways\Gateway For 2D Data\`AVEVA.NET.Gateways.2DData.exe" -cp `"D:\Test\New_1\New_1.xml" -context "XXX|YYY|ZZZ" updates context in EIWM output file from command line parameter "XXX|YYY|ZZZ"`

With project configuration file and manifest attributes: `"C:\Program Files\AVEVA\AVEVA NET Gateways\Gateway For 2D Data\`AVEVA.NET.Gateways.2DData.exe" -cp `"D:\Test\New_1\New_1.xml"` -ma "#MODEL_NAME#:My Output"

With Gateway Configuration Tool project configuration file and output folder for upgraded project: `"C:\Program Files\AVEVA\AVEVA NET Gateways\Gateway For 2D Data\AVEVA.NET.Gateways.2DData.exe" -cp "c:\GCTproject\GCTproject.xml" -upgradedprojectlocation "C:\UpgradedProject"`

Viewports Style:

Example 1: `-vsviewports mixed:0.5`

Example 2: `-vsv wireframe`

Example 3: `-vsv raster:5`

Block Style:

Example 1: `-vsblock "companyLogo1":"c:\logo1.png" -vsblock "companyLogo2":"c:\logo2.png"`

Example 1: `-vsb "companyLogo.*":"c:\logo.png"`

Hatches style:

Example 1: `-vshatch DOTS:0.01:SOLID -vshatch AR-CONC:999:SOLID`

Example 2: `-vsh AR-CONC:0.1:SOLID`

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, command line prompt throws an exception stopping the execution process.

  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 mentioned in the config file is taken as the default output path.
  - 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 in that provided path and the output files are generated there.
  - 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.2DData.exe`.

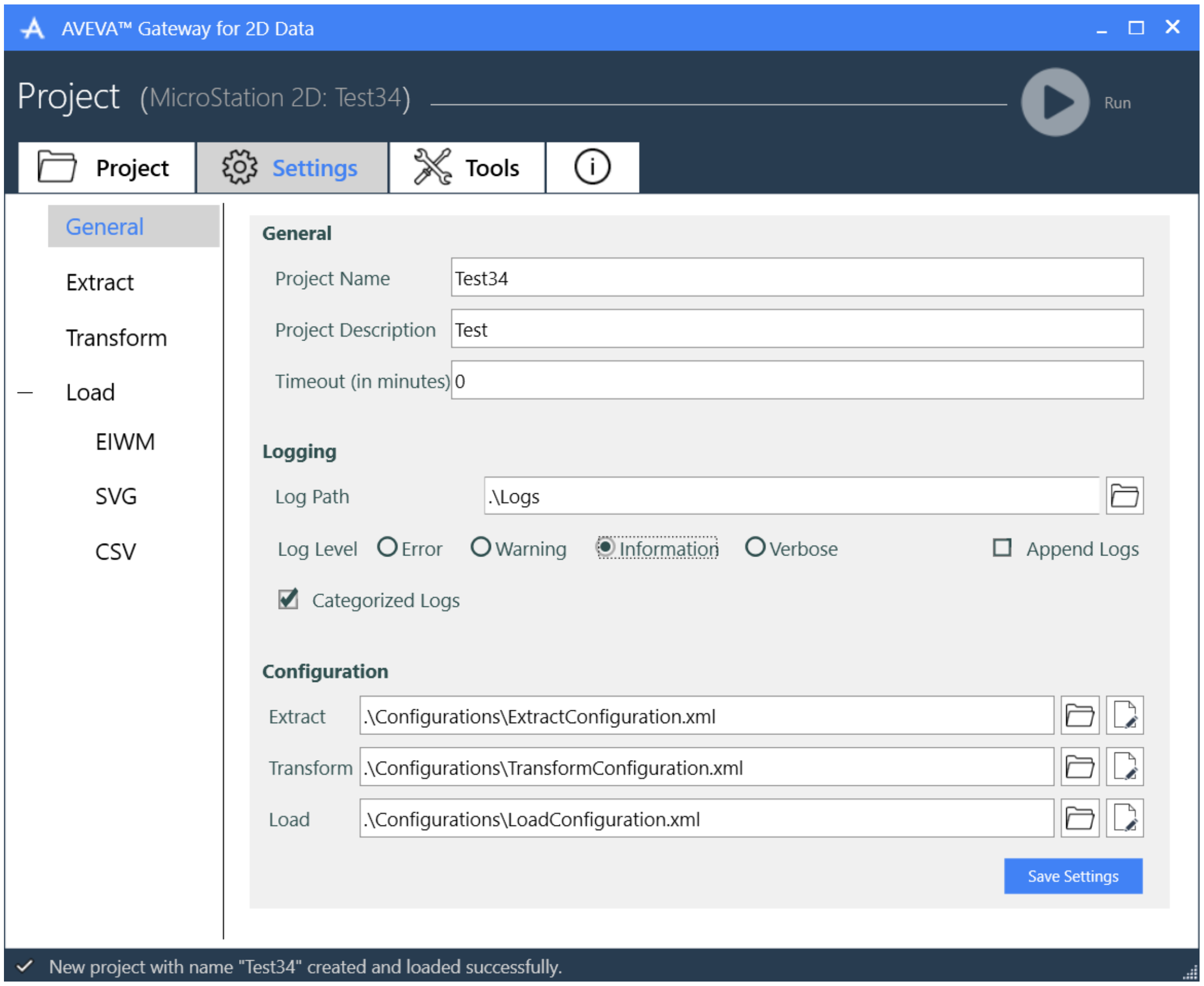
Exit Code	Definition
0	Success
1	Gateway processing stopped because of general error
3	Gateway processing canceled
7	Processing failed in extractor
9	Processing failed in transformer
11	Processing failed in SVG loader
13	Processing failed in EIWM loader

17	Processing failed due to no input files
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.

Chapter 4 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.



Project Name: Test34
Project Description: Test
Timeout (in minutes): 0
Log Path: .\Logs
Log Level: Error Warning [✓ Information] Verbose
Append Logs: ☐
Categorized Logs: ✓
Extract: .\Configurations\ExtractConfiguration.xml
Transform: .\Configurations\TransformConfiguration.xml
Load: .\Configurations\LoadConfiguration.xml

✓ New project with name "Test34" created and loaded successfully.

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 except that messages with level 'Verbose' are never added to the JSON report file.
- 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 2D Data"`	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.
`"SourceName"`	`"Sample.dwg"`	Name of source data.
`"SuccessfulObjectsNumber"`	`1255`	Number of successfully processed objects.
`"ProcessingResult"`	`"Success"`	Overall result of the processing: Success/Failure/Timeout/Canceled
`"ProcessingResultNotes"`	`"Warnings"`	Additional note about the result of the processing.
`"ReturnCode"`	`0`	Numeric code returned from the processing
`"Configurations"`	...	List of configuration files.
`"OutputData"`	...	List of output files.
`"Options"`	...	Additional processing options.
`"Reports"`	...	List of messages (see the following note).

Note: The `ProcessingResult` field is set according to the following rules:

- `"Timeout"`, if processing is broken due to configured timeout and is set to be not reported as error.
- `"Canceled"`, if processing is broken due to the user’s cancellation.
- `"Failure"`, if a major error has caused the processing to be terminated early.

The `ProcessingResultNotes` field is set according to the following rules:

- `"No errors, no warnings"`, if processing is completed without any errors nor warnings.
- `"Warnings"`, if processing is completed with warnings.
- `"Minor errors"`, if processing is completed with errors.
- `"Minor errors, warnings"`, if processing is completed with minor errors and warnings.
- `"Processing Cancelled"`, if processing is broken due to the user’s cancellation.

- `"Timeout occurred"`, if processing is broken due to configured timeout.
- `"Failed in <module name>"`, if processing failed in specified module.

The `ReturnCode` field is set according to the exit code returned from the Gateway.

The JSON report then contains one or more status messages:

Field	**Value Example**	**Description**
`"ID"`	`2`	Number of the message.
`"DateTime"`	`"06-02-2020 19 06 43"`	Timestamp of the message.
`"SubComponent"`	`"AutoCAD 2D"`	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 caused by a cancellation or a timeout.
Debug	Low level technical information describing processing details. Only for severity Debug.
Environment	Issue influenced by system environment such as not enough memory and read only file.
Extract	Problem with extracting some source data like wrong item or unrecognized API behavior.
Information	Typical informational 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.

Message Category	Message Description
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	Output data was automatically adjusted to conform with the output format requirement.
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 features 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 5 Appendix A: Mapping Configuration

Mapping configuration files are generally defined in XML format. 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:

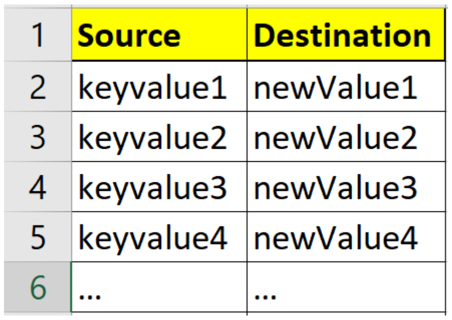
```
<MSExcelLookupDataSource
     id="Excel Map"
     file="C:\Mapping\Data Value Lookup\Mapping.xls"
     table="Sheet1"
     sourceColumn="[Source]"
     targetColumn="[Destination]" />
```

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, 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="OLEDB.NET=true;PLSQLRSet=true;Data
Source=Host;User     ID=user;password=pass;"
  ID=user;password=pass;"
  query="SELECT [Key], [Value] FROM [Table1]"
  sourceColumn="Key"
  targetColumn="Value" />
```

or

```

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

Notes:

- 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: RegEx 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, configure an SSL encrypted connection between the two.

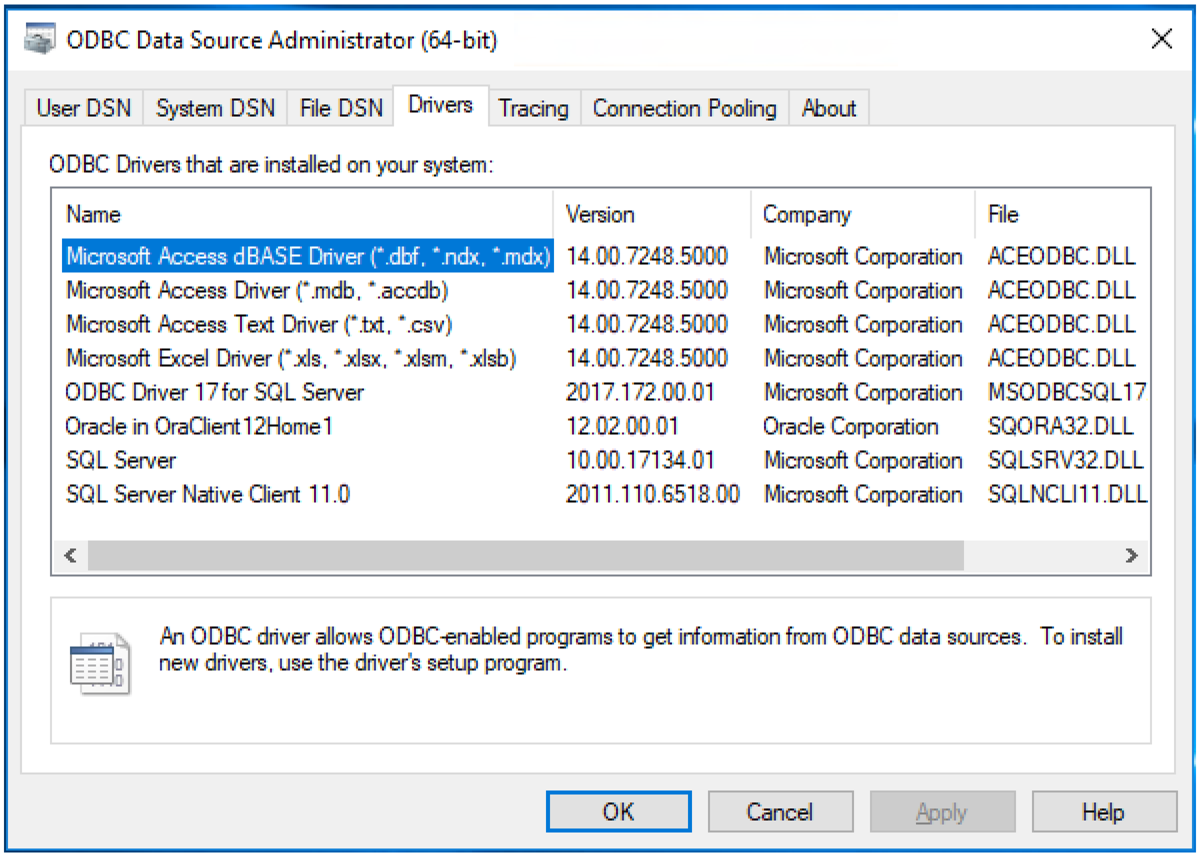
- Because the encrypted password is associated with the local machine where the Gateway is running, other machines cannot use the configuration file directly. You will have to encrypt the password before using the configuration created in another machine. Obtain an encrypted password using the encryption utility.

Migrating 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:



ODBC Drivers that are installed on your system:

Name	Version	Company	File
Microsoft Access dBASE Driver (*.dbf, *.ndx, *.mdx)	14.00.7248.5000	Microsoft Corporation	ACEODBC.DLL
Microsoft Access Driver (*.mdb, *.accdb)	14.00.7248.5000	Microsoft Corporation	ACEODBC.DLL
Microsoft Access Text Driver (*.txt, *.csv)	14.00.7248.5000	Microsoft Corporation	ACEODBC.DLL
Microsoft Excel Driver (*.xls, *.xlsx, *.xlsm, *.xlsb)	14.00.7248.5000	Microsoft Corporation	ACEODBC.DLL
ODBC Driver 17 for SQL Server	2017.172.00.01	Microsoft Corporation	MSODBCSQL17
Oracle in OraClient12Home1	12.02.00.01	Oracle Corporation	SQORA32.DLL
SQL Server	10.00.17134.01	Microsoft Corporation	SQLSRV32.DLL
SQL Server Native Client 11.0	2011.110.6518.00	Microsoft Corporation	SQLNCLI11.DLL

An ODBC driver allows ODBC-enabled programs to get information from ODBC data sources. To install new drivers, use the driver's setup program.

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:

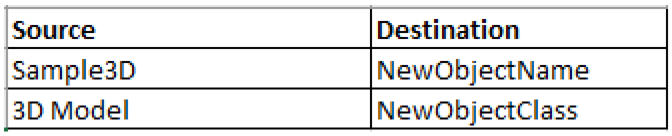
- for SQL Server
- https://www.oracle.com/in/database/technologies/releasenote-odbc-ic.html for Oracle

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.

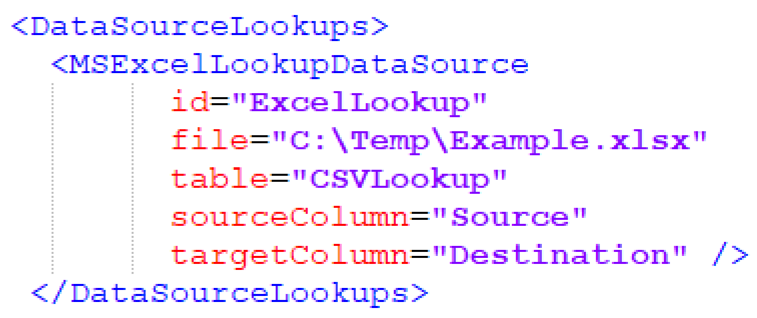
Example of an Excel file content:



Source	Destination
Sample3D	NewObjectName
3D Model	NewObjectClass

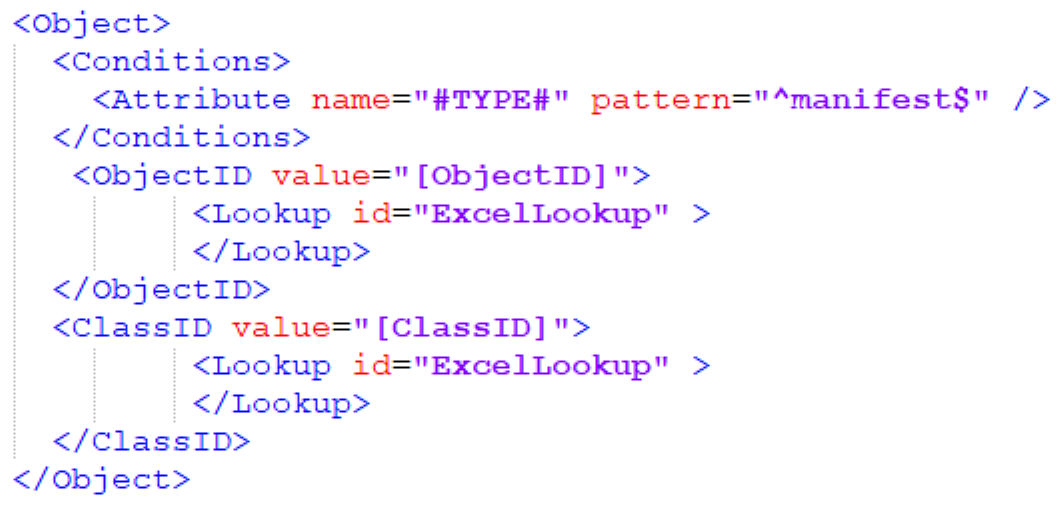
The following example modifies the `ObjectID` and `ClassID` of the Document Object (Manifest) by matching the input data to the `Source` column and using the `Destination` column from the lookup file.

Lookup Declaration:



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

`Object ID` and `Class ID` Transformation:



```
<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>
```

Handling Custom/Proxy Objects

AutoCAD 2D drawings may not be extracted correctly if they contain custom/proxy objects. If the Gateway cannot process these objects, you should open the drawing in AutoCAD and save these models to the native AutoCAD DWG format. To do this, use an **Object Enabler** (program supplied by the creator of the objects), to read (open) these custom objects in AutoCAD and use specific commands of the **Object Enabler** to convert the proxy objects to native objects before writing to a new DWG file.

Default Value Mapping in Attribute

While creating new attributes, 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>
</ObjectMappings>
```

Note: If an object is not associated with any template Lookup ID, a default TemplateID is generated with 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.

It is possible to 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 a 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}`".

Using Space as a Compose Delimiter

When defining transformations in Compose, if a space needs to be the delimiter value then use "\s" to define space as a delimiter rather than a space character (" ").

Using \s instead of a regular space prevents misinterpretation and ensures proper parsing of the attribute values.

Example Configuration

```
<Object>
      <Conditions>
        <Attribute name="ClassID" pattern=".*" />
      </Conditions>
      <Attributes keepUnmappedAttributes="True">
        <Attribute name="Type">
          <Name value="NewType" />
          <Value value="[Type]">
            <Transforms>
              <InsertBefore pattern=".*" value="#" />
              <Replace pattern="-" value=" " />
              <InsertAfter pattern="Connector" value="#" />
              <Keep pattern=".*" />
              <Compose delimiters="\s" constructor="{2} ## {3}" />
            </Transforms>
          </Value>
        </Attribute>
      </Attributes>
    </Object>
```

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

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. Use AutoNumber 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 file 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`.

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 file 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 "AC2D".
#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".
#INPUT_LOCATION#	Full path of the input .DWG/.DXF file.
#INPUT_FOLDER#	Parent directory of the input .DWG/.DXF file.
#MODEL_NAME#	Derived from the input .DWG/.DXF filename excluding its extension.
#KEEP_FOLDER_TREE#	TRUE, if input folder hierarchy is preserved in output generation.
#TARGET_LOCATION#	Output folder path.
#UNITS#	Source drawing units.
#LAYOUT_NAME#	Derived from the input layout name that is processed.

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 type="^is referenced in$" refID="associated">
    <Conditions>
      <Attribute name = "#TYPE#" pattern = "^manifest$" />
    </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>
```

Note: The use of "parent" for identifying attributes of associated objects can only be used when updating associations and cannot be used as refID.

User-Defined Manifest Attributes

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 type="^is referenced in$" refID="associated">
      <Conditions>
         <Attribute name = "#TYPE#" pattern = "^manifest$" />
      </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 2D 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. 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]"/>
<ClassID value ="[Classification]" />
   <Attributes>
   <Attribute name="*" >
         <Remove />
      </Attribute>
    </Attributes>
```

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

Patterns 2D

Patterns2d configuration allows you to select, via `Conditions`, which type of graphical objects to amend so that tags can be associated or disassociated with graphical elements using the `Group` and `Ungroup` functions and add new `Attributes` to the grouped and ungrouped objects. The Patterns2d extension is similar to base mapping of engineering objects but allows transformations related to graphical objects.

It is possible to filter objects by attributes and associations kept in engineering data associated to graphical elements. Select types of graphical objects that will be grouped in the defined area around selected object.

The following is the complete set of configuration settings supported by Patterns2d extension:

```
<Patterns2d xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/
XMLSchema-instance" sourceProductName="AVEVA™ Gateway for 2D Data"
componentName="Patterns2d" componentVersion="2.10.0.0"
  <Object>
    <Conditions>
       <ObjectType ObjectType="
                symbol,
                symbol_single,
                symbol_single_textonly,
                symbol_nested,
                symbol_nested_textonly,
                table,
                dimension,
                mleader,
                mtext,
                oval,
                circle,
                ellipse,
                arc_closed,
                arc_open,
                all_lines_closed,
                polyline_closed,
                spline_closed,
                region,
                text,
                line,
                all_lines_open,
                polyline_open,
                spline_open,
                point,
                3dsolid,
                leader,
                fill"
                />

        <Attribute  Name="handle"
                            Value=".*" />

        <Association Type="is a part of"
                         TargetAttributeName=".*"
                         TargetAttributeValue=".*" />

                <Size   min="0"
                        max="30" />

        <NumberOfObjects     min="1" max="80000" />
        </Conditions>
        <Group Separator="-" JoinTextFrom="topleft" IncludeParts="false" GrabTypes="
         all,
       symbol,symbol_single,
       symbol_single_textonly,
       symbol_nested,
       symbol_nested_textonly,
       table,
       dimension,
       mleader,
       mtext,
       oval,
       circle,
       ellipse,
       arc_closed,
       arc_open,
       all_lines_closed,
       polyline_closed,
       spline_closed,
       region,
       text,
       line,
       all_lines_open,
       polyline_open,
       spline_open,
       point,
       3dsolid,
       leader,
       fill"

       OrderByGroup="true"
       KeepTextOnly="true" >
       <IncludeArea allDirections="1.1"
        top="0.6"
        bottom="5.0"
        right="1.5"
        left="0.9"
        />

      </Group>
      <Attribute Name="grouped" Value="" />

      <Attribute Special="ExtentCoordinates" />

      <Ungroup KeepSingleSymbols="true" />

   </Object>
```

```
</Patterns2d>
```

Graphical types from AutoCAD and MicroStation drawings are extracted as the following Patterns 2D types (for the purpose of grouping):

Patterns 2D type	**AutoCAD type (attribute)**	**MicroStation type (attribute)**
text	Text (text)	Text (text)
mtext	MText (mtext)	Node (mtext)
symbol	Block Reference (blockreference) Proxy (proxyentity)	Cell (cell) Shared Cell (cell) Smart Solid (cell) Tagged object + Tags (cell) Complex Shape (complexshape)
point	Dot (point)	Multi-Line (point) Line - zero length (point)
line	Line (line) MLine (line)	Line (line)
circle	Circle (circle)	Circle (circle)
ellipse	Ellipse (ellipse)	Ellipse (ellipse)
polyline	Polyline (polyline)	Complex Chain (polyline)
arc	Arc (arc)	Arc (arc)
spline	Spline (spline)	Bspline Curve (spline)
region	Region (region)	Associative Region (region) Shape (region)
table	Table (table)	Table (table)
leader	Leader (leader)	
mleader	Multileader (mleader)	Note (mleader)
dimension	Dimension - all types (dimension)	Dimension (dimension)
fill	Hatch (hatch)	Graphics with Fill set to Opaque or Outlined (hatch)
3dsolid	3DSolid (3dsolid)	

`Conditions: ObjectType, Attribute, Association, Size, NumberOfObjects`.

Attribute and Association conditions can be used multiple times and `ObjectType`, `Size`, `NumberOfObjects` conditions can be used only once.

Due to regular expressions used for `Attribute` and `Association` settings, specify multiple values to extend the range of selected objects.

When multiple criteria are used in a `Condition` only objects that satisfy all the criteria are selected for the subsequent actions.

The `Patterns2d Conditions` are as follows:

- `ObjectTypes`: This is mandatory and allows selection by object type, where at least one type must be provided. Some keywords represent a few types:
- `all`: `all object types`
- `symbol`: `symbol_single`, `symbol_single_textonly`, `symbol_nested`, `symbol_nested_textonly`
- `symbol_single`: `symbol_single_textonly`
- `symbol_nested`: `symbol_nested_textonly`
- `oval`: `circle`, `ellipse`, `arc_closed`
- `allLines_open`: `polyline_open`, `spline_open`
- `allLines_closed`: `polyline_closed`, `spline_closed`, `region`

  Note: `symbol` is the generic term for a block reference, cell, symbol or shape, depending on the source system.

- `Attribute`: (optional) Same condition as in Base Mapping.
- `Association`: (optional) Same condition as in Base Mapping. Additionally, this parameter allows filtering by `TargetAttributeName` and/or `TargetAttributeValue`.
- `Size`: (optional) Refers to size of the object in comparison to the drawing's size. Default =100 (%) values: from 0.0 to 100.0.

  Specify the minimum and maximum (`min` and `max`) size of the objects that should be selected for grouping or ungrouping. Values provided denote the percent range within which a given object must fit in relation to the entire square area of the drawing.

  For example: <Size min="3" max="10"/> means that objects whose rectangular area takes at least 3% and not more than 10% of whole drawing area will take part in the selected action (Group or Ungroup).

- `NumberOfObjects`: 'This parameter allows you to select graphical objects based on the number of elements, for example, if the ObjectType is `mtext` and NumberOfObjects has `min="3"` and `max ="3"` then only tuplets (tags consisting of 3 lines) will be selected.

  (optional) default =1000000000, values: from 0 to 1000000000.

Grouping:

- **Group**: This action aggregates all of the objects selected by the Conditions into a single grouped object with ObjectID formed from the concatenation of its text elements from the labels belonging to the objects included in this group. When a Group is created then regardless of what types of objects belong to that group, it contains only one tag and does not contain any children objects. This way of storing data is adjusted to rendering systems like Workhub and Dashboard will provide proper tagging and highlighting of tagged objects.

Note: The minimum number of objects in a group is one single text object. A group can contain any number of primitives. After an object is assigned to a group, it won't be subject to any subsequent Group or Ungroup actions.

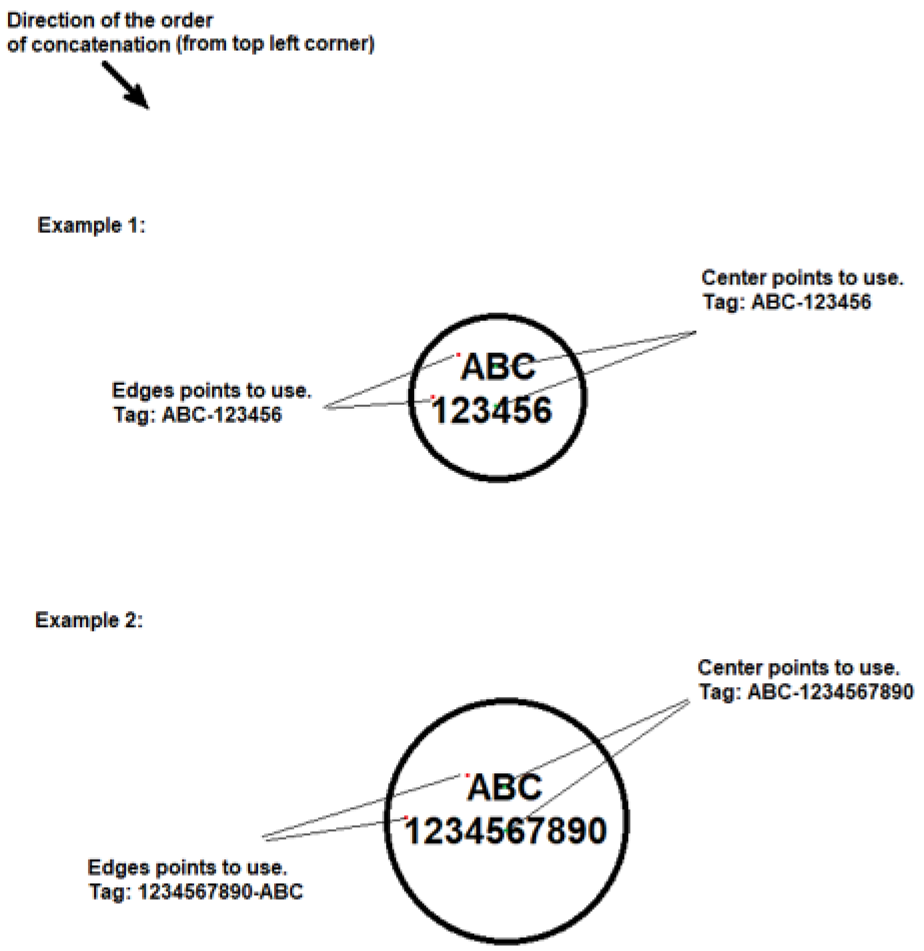
Settings:

- `Separator`: Any string, default '-', used for concatenating text items into common ID.
- `JoinTextFrom`: Default: topleft, values: top, bottom, right, left, topleft, topright, bottomleft, bottomright, used for setting from which direction text will be concatenated.

Note: If the text is rotated from the horizontal it is first unrotated before the `JoinTextFrom` directions are applied. If grouped text objects are rotated from the horizontal and the maximum difference between all texts angles is less than 5%, then they are first unrotated before the JoinTextFrom directions are applied.

- `TextControlPoint`: The order of concatenation is determined by the position defining the location of each text element.

  There are two values: Center (Default) and Edges (which is the start position in X, Y of each text element).



- `IncludeParts`: If set to true, then any object that is partly within the selected object's extents will be included in the group, otherwise, if `false` (default) only objects that are fully enclosed in the selected objects extents will be grouped.
- `GrabTypes`: Filters which type of objects will be grouped together. The default is text and mtext and all other object types are allowed (see example above).

  Objects of types selected for GrabTypes are collected into one tagged group.

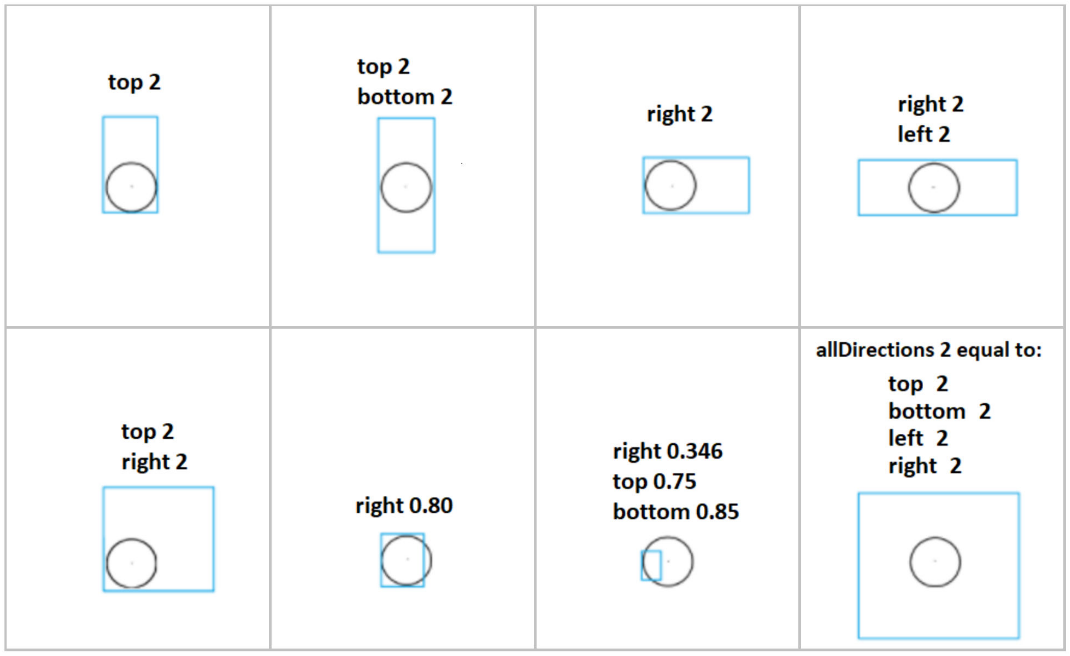
  Note: If there is no text object in the group, then ID is not set but objects are still grouped. Use Base mapping to tag these objects, therefore, it's best to apply Pattern 2D mapping before Base mapping.
- `KeepTextOnly`: If true, then regardless of what types of objects are selected by `Conditions`, `GrabTypes` and `IncludeArea`, only text objects. This way of tagging can be useful in the systems where during selection only highlighting of the text is expected.
- `IncludeArea`: Allows you to vary the search area of the object's extents by changing its overall size either symmetrically (if allDirections parameter is used) or in each orientation direction (if top, bottom, left or right is used, including combinations of these).

  Examples:

  - `allDirections=2.5`
  - `left=0.5 right=2.3 top=2.0 bottom=0.1`

  For example, `left=0.5` and `bottom=2.0` means searching for objects in 50% of horizontal length of the object (right-half of the original extent), additionally, this extent area will be extended into bottom direction; and vertical length will be extended to 200% of the original length.
- When `left+right<1.0` or `top+bottom<1.0` or `allDirections<0.5,` then a configuration error is generated.
- Therefore, the minimal settings will provide at least a single common point (`allDirections=0.5`) or a single line (`right+left=1 or top+bottom=1`), if `IncludeParts="true"` and will result in including all objects crossing that common point or that single line.

Properties: left, right, top, bottom, allDirections



Note: By default, if the `IncludeArea` setting is not used, then all values are set to 0 for Object types `symbol, table, dimension, text, mtext, mleader`; so only child objects that belong to the main object are grouped regardless of their location. You may include objects which do not belong to the main object by increasing values in the `IncludeArea` setting. This will add some objects placed within the main object's extent or also add some objects which are placed out of the main object's extent.

For the rest of the Object types that do not have any child objects, default value of `IncludeArea` parameter for each of these attributes is set to 1.0 (100%). This means grouping all child objects from the extent of the main object. This range can be reduced by using values less than 1.0 or can be extended by setting more than 1.0.

- `Attribute`: Adds attribute to grouped object. `Name` is mandatory, `Value` is optional. Add any number of attributes, including attributes normally set in Base mapping such as `ClassID`.
- `Attribute Special="ExtentCoordinates"`: If specified then the grouped object's extents are added as four attributes containing the coordinates of the minimum and maximum positions: `#Xmin#, #Xmax#, #Ymin#, #Ymax#`.

Ungrouping

- `Ungroup`: Works in the opposite way to grouping by splitting structured data into smaller chunks so they can be used independent-single objects or single symbols, depending on the setting. It uses the same conditions for grouping but uses a different (shorter) list of `objectTypes` for filtering only those objects having structure to ungroup:
  - `symbol`
  - `symbol_Single`
  - `symbol_SIngle_TextOnly`
  - `symbol_Nested`

- `symbol_Single_TextOnly`
- `table`
- `dimension`
- `mtext`
- `mleader`

Settings:

- `KeepSingleSymbols`: If this option is set to `true`, then symbols are not ungrouped completely to separate objects level but ungrouping stops when each symbol no longer contains any nested symbol. In other words, with this option all nested symbols are moved to the root level.

  Note: This setting does not have an effect on other structured data that do not contain symbols.

  Additional Notes:

  - Any number of ‘`Object`’ nodes can be defined.
  - In each `'Object'` node, you only use a ‘`Group`’ or ‘`Ungroup`’ node but not both.
  - Do not group the same object twice.
  - If you ungrouped `symbol_nested` into multiple `symbol_single` objects, you still use another `'Object'` node to ungroup `symbol_single` into separate primitives.
  - Using multiple `Patterns2d` extension configurations provide the same effect like using multiple `'Object'` nodes in one `Patterns2d` extension configuration.
  - Objects extracted by the Smart P&ID Extractor save tag-related text in attributes named `"#TEXT#"` and multiple graphics may be extracted as a single object. If there are multiple text labels in this multiple graphics, then the `#TEXT#` attribute value of the group is a concatenation of these text labels, ordered by position from top-left to bottom-right. These objects cannot then be ungrouped in Patterns 2D but may still be grouped with other objects. In this case the Patterns 2D grouping behaviour would apply where if there are multiple `"#TEXT#"` attributes then the ObjectID of the group is a concatenation of these values, ordered by direction configured in the Patterns 2D configuration. If the objects have not been grouped by Patterns 2D, then base mapping may be needed to assign their `"#TEXT#"` values to ObjectID.

Base Mapping

Base Mapping modifies the engineering data of a model’s objects present in the extracted data. 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 loading into AIM.

An Object Mapping example is as follows:

```
<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_SVG#`: Sets output name for SVG.

`#TARGET_NAME_CSV#`: Sets output name for CSV.

Example of Base Mapping:

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

Note: The Output name for **EIWM** is overridden by setting the **File Name** under **Custom Folder Structure** if set.

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`.

When using AutoCAD 2D, MicroStation 2D, or Smart P&ID Extractor, text strings from drawings are stored in attributes named `#TEXT#`, which can then be used to tag objects.

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="[#TEXT#]">
  </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="[#TEXT#]">
    <Lookup id="ExcelLookup" >
      <FailAction action = "DiscardObject" />
    </Lookup>
  </ObjectID>
    ...
</Object>
```

In this case, the value of the "[#TEXT#]" 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.

Use Transforms to modify the value of an attribute, as shown below:

```
<Object>
    ...
<Conditions>
    <Attribute name="object_class" pattern="Door" />
  </Conditions>
  <ObjectID value="[#TEXT#]_[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". 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="[#TEXT#]_[associated:#TEXT#]" />
    ...
</Object>
```

In this case above, the value of the [#TEXT#] attribute along with the value of [#TEXT#] 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="[#TEXT#]"/>
</Object>
```

Note: The log files indicate a warning message whenever multiple objects in the EIWM have a common `Object ID` value. 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.
```

Value Control

This optional parameter controls how the combined value of multiple attributes are affected when any of the attribute values are blank or null (that is, missing). It takes one of the below three states:

- **Always** – in this case the value is a literal combination of the referenced attributes and separators (if the Separator parameter is defined). This is the default behaviour.
- **RemoveEmpty** – in this case if a referenced attribute is empty, blank (either single or multiple blank spaces) or NULL, then both the attribute and its relevant separator (if the Separator parameter is defined) are not included in the value.
- **IgnoreIfAnyEmpty** – in this case if any of the referenced attributes have a value that is Empty, blank (single or multiple blank spaces) or NULL then the value is not updated.

Example 1:

```
val1: "A"
val2: " " (or empty, multiple blank spaces or NULL)
val3: "C"
<ObjectID value="[assoc1:val1]-[assoc2:val2]-[assoc3:val3]"
valueControl="RemoveEmpty"/>
```

Result: `ObjectID` value is updated to `"A--C"`.

Note: The '-' characters are not recognized as separators hence are not also removed, see the Separator parameter below.

Example 2:

```
val1: ""

val2: " "

val3: null
<ObjectID value="[assoc1:val1]-[assoc2:val2]-[assoc3:val3]"
valueControl="RemoveEmpty"/>
```

Result: `ObjectID` value is not updated.

Example 3:

```
val1: "A"
val2: null
val3: "C"
<ObjectID value="[assoc1:val1]-[assoc2:val2]-[assoc3:val3]"
valueControl="RemoveEmpty"/>
```

Result: `ObjectID` value is updated to `"A--C"`.

Example 4:
```
val1: "A"
val2: " " (or empty, multiple blank spaces or NULL)
val3: "C"
<ObjectID value="[assoc1:val1]-[assoc2:val2]-[assoc3:val3]"
valueControl="IgnoreIfAnyEmpty"/>
```

Result: `ObjectID` value is not updated.

Example 5:
```
val1: ""
val2: " "
val3: null
<ObjectID value="[assoc1:val1]-[assoc2:val2]-[assoc3:val3]"
valueControl="IgnoreIfAnyEmpty"/>
```

Result: `ObjectID` value is not changed.

Example 6:
```
val1: "A"
val2: null
val3: "C"
<ObjectID value="[assoc1:val1]-[assoc2:val2]-[assoc3:val3]"
valueControl="IgnoreIfAnyEmpty"/>
```

Result: `ObjectID` value is not updated.

Example 7:
```
val1: "A"
val2: " "
val3: "C"
<ObjectID value="[assoc1:val1]-[assoc2:val2]-[assoc3:val3]"
valueControl="Always"/>
```

Result: `ObjectID` value is updated to `"A- -C"`.

Example 8:
```
val1: ""
val2: " "
val3: null
<ObjectID value="[assoc1:val1]-[assoc2:val2]-[assoc3:val3]" valueControl="Always"/>
```

Result: `ObjectID` value is updated to `"- -"`.

Example 9:
```
val1: "A"
val2: null
val3: "C"
<ObjectID value="[assoc1:val1]-[assoc2:val2]-[assoc3:val3]" valueControl="Always"/>
```

Result: `ObjectID` value is updated to `"A--C"`.

Separator:

The **Separator** parameter allows an explicit separator function to be applied, where the value of this parameter is inserted between the referenced attribute values.

Note: The Separator parameter cannot be used if text characters are also included in the value definition.

Example 1:

```
val1: "A"
val2: "B"
val3: "C"
<ObjectID value="[assoc1:val1][assoc2:val2][assoc3:val3]"
Separator="-" />
```

Result: `ObjectID` value is updated to `"A-B-C"`.

Example 2:

```
val1: "A"
val2: " " -> It can be Empty, single or multiple blank space, NULL
val3: "C"
<ObjectID value="[assoc1:val1][assoc2:val2][assoc3:val3]"
valueControl="Always" Separator="-" />
```

Result: `ObjectID` value is updated to `"A- -C"`.

Example 3:

```
val1: "A"
val2: " " (or empty, multiple blank spaces or NULL)
val3: "C"
<ObjectID value="[assoc1:val1][assoc2:val2][assoc3:val3]"
valueControl="RemoveEmpty" Separator="-" />
```

Result: `ObjectID` value is updated to `"A-C"`.

Example 4:

```
val1: "A"
val2: " " (or empty, multiple blank spaces or NULL)
val3: "C"
<ObjectID value="[assoc1:val1][assoc2:val2][assoc3:val3]"
valueControl="IgnoreIfAnyEmpty" Separator="-" />
```

Result: `ObjectID` value is not updated.

Example 5:

```
val1: " " (or empty, multiple blank spaces or NULL)
val2: "B"
val3: "C"
<ObjectID value="[assoc1:val1][assoc2:val2][assoc3:val3]"
valueControl="Always" Separator="-" />
```

Result: `ObjectID` value is updated to " `-B-C"`.

Example 6:

```
val1: "A"
val2: "B"
val3: "C"
<ObjectID value="random[assoc1:val1]_[assoc2:val2]_[assoc3:val3]"
valueControl="IgnoreIfAnyEmpty" Separator="-" />
```

Result: A validation Error is raised that the **Separator** parameter cannot be used with any other text inside the value expression other than referenced attributes.

Note: This parameter can be used whenever any attribute value is being derived from multiple attributes.

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.
```

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>
```

Use attributes from associated objects to modify the value of `ClassID` as shown below:

```
<Object>
   ...
   <IncludeAssociatedObjectAttributes type = "^FillsVoids$" refID = "[ClassName]" >
      <Conditions>
       <Attribute name = "ClassName" pattern = "^OPENINGELEMENT" />
       <Attribute name = "Name" pattern = "^element1$" />
      </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`.

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>
```

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.

To modify context value, use Lookups and transforms.

```
<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>
```

Use objects from associated objects to modify the value of ContextID, as shown below:

```
<Object>
   ...
 <IncludeAssociatedObjectAttributes type = "^FillsVoids$" refID = "[ClassName]" >
  <Conditions>
   <Attribute name = "ClassName" pattern = "^OPENINGELEMENT" />
   <Attribute name = "Name" pattern = "^element1$" />
  </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.

To modify the ObjectName value, use Lookups and transforms.

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

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

Attributes from associated objects can also be used, as shown below:

```
<Object>
   ...
   <IncludeAssociatedObjectAttributes type ="^FillsVoids$" refID ="[Name]">
      <Conditions>
       <Attribute name = "ClassName" pattern = "^OPENINGELEMENT" />
       <Attribute name = "Name" pattern = "^element1$" />
      </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.

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>
```

Attributes from associated objects can also be used, as shown below:

```
<Object>
   ...
    <IncludeAssociatedObjectAttributes type ="^FillsVoids$" refID="[Tag]" >
     <Conditions>
      <Attribute name = "ClassName" pattern = "^OPENINGELEMENT" />
      <Attribute name = "Name" pattern = "^element1$" />
     </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`.

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

An 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 an 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 mapping entry to create a new attribute taking multiple attributes with their corresponding patterns to match a 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>
  </Value>
</Attribute>
</Attributes>
...
</Object>
```

Update:

The simplest form of an 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#.

Attributes from associated objects can also be used, as shown below:

```
<Object>

<IncludeAssociatedObjectAttributes type = "^FillsVoids$" refID = "[Tag] " >
    <Conditions>
      <Attribute name = "ClassName" pattern = "^OPENINGELEMENT" />
      <Attribute name = "Name" pattern = "^element1$" />
     </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 type ="^FillsVoids$" refID = "[associated:Tag]" >
    <Conditions>
     <Attribute name = "ClassName" pattern = "^OPENINGELEMENT" />
     <Attribute name = "Name" pattern = "^element1$" />
    </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 type = "^Materials$" refID =
"[associated{descendantLevel=2}:Name] ">
```

```
    <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>
```

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`

When only the path is set in the **Input Locator** field, then all the DWG/DXF files present in the top folder and sub-folders will be processed. If any DWG/DXF files contain internal references (XREF) to other DWG/DXF files, then they will automatically be included in the processing of the main DWG/DXF file. Therefore, if referenced files are placed in the same folder or sub-folders of the folder path selected for processing, then they may be processed more than once, for example, as a referenced file and as an individual file. This can be prevented by either ensuring all referenced files are not located in the input root folder or its sub-folders or by processing each specific DWG/DXF file representing the model as a single file in the input locator rather than setting it to the folder pathname.

DWG objects are read with all their internal information. This information is attached directly to these objects as attributes:

Attribute name for AutoCAD handle: "`handle`"

Attribute name for AutoCAD parent handle: "`parent handle`"

Attribute name for AutoCAD layer: "`layer`"

Attribute name for AutoCAD block name: "`block name`"

Attribute name for AutoCAD object's type: "`autocad type`"

Attribute name for main and each referenced ("XRefs") AutoCAD models: "`#MODEL_NAME#`"

Attribute name for AutoCAD layout name: "`#LAYOUT_NAME`"

When multiple layouts of the drawing are processed, then the `#MODEL_NAME#` attribute is a concatenation of the filename and the layout name separated by an underscore character.

Each of these attributes can be used for tagging or classification of objects.

When objects are extracted from reference files, the value of the `#MODEL_NAME#` parameter is set to the name of the reference file. If relevant, this name can then be used to define the object’s ID.

```
    <Object>
      ...
      <Conditions>
        <Attribute name="#MODEL_NAME#" />
      </Conditions>
      <ObjectID value = "[#MODEL_NAME#]"/>
      <ClassID value = "Equipment"/>
      <ContextID value="Avngate|XREF" />
     ...
</Object>
```

The "`MODEL_NAME`" attribute is used to set the tag (`ObjectID`) for all objects according to their source DWG file name. Additionally, the `ContextID` attribute is set with a nested name "`XREF`" to ensure uniqueness and it differentiates the main model's object tag from its child objects' tags.

DWG objects may have additionally associated two types of attributes:

- block attributes which can be attached only to objects of type "`block reference`".
- attributes represented by extended object data (`XData`) which can be attached to any kind of objects.

After reading this data from DWG, both groups of attributes are instantiated as two separate objects with attributes and are attached to main DWG object. DWG object refers to these objects by association of type: "`has dataset`".

Each object read from DWG has set `Handle` attribute; so objects created for keeping the additional attributes have always set `Handle` attributes with values, respectively: `"block dataset of <internal object handle>"` or `"XData dataset of <internal object handle>"`.

To export them to EIWM XML these objects representing groups of attributes (datasets) must be tagged by setting "`ObjectID`" attribute value and classified by setting "`ClassID`" attribute.

The following example first creates two new attributes: "`ObjectID`" and "`ClassID`" for both dataset objects and as a result they will appear in AIM. Last section of the mapping creates "`ObjectID`" and "`ClassID`" for each main DWG object which has associated dataset object.

```
    <!-- Set tag and classify Datasets created from block's attributes -->
<Object>
 ...
<Conditions>
<Attribute name="Handle" pattern=".*block dataset of.*" />
</Conditions>
<ObjectID value = "[handle]"/>
<ClassID value = "Equipment"/>
   ...
</Object>
<!-- Set tag and classify Datasets created from XData -->
<Object>
 ...
<Conditions>
<Attribute name="Handle" pattern=".*XData dataset of.*" />
</Conditions>
<ObjectID value = "[handle]"/>
<ClassID value = "Equipment"/>
    ...
</Object>
<!-- Set tag and classify objects -->
<Object>
 ...
```

```
<IncludeAssociatedObjectAttributestype="has dataset" refID = "[associated:Tag]"
" [associated:Type]" />
```

```
   <Conditions>
    <Attribute name = "Tag" />
   </Conditions>
 </IncludeAssociatedObjectAttributes>
 <ObjectID value = "[associated:Tag]"/>
 <ClassID value="[associated:Type]" />
```

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

In these cases "ClassID" attribute is created with constant values "Equipment".

"ObjectID" attribute is created with the value taken from attribute Handle.

Main DWG object has "ObjectID" and "ClassID" attributes created with values taken from one of the associated dataset which contains attributes names "Tag" and "Type".

Excluding Objects from Output Files

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_SVG# - excludes the object's hotspot (vnet:id) tagging information from the SVG 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 will not 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

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. To delete these objects, use 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.

An Association mapping example 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>
                 ...
   </Object>
```

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). 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 DTOtion 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, 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.

When extracting data from input sources that have hierarchies between objects, for example between block references, nested block references and drawing items (primitive graphics) in diagrams, these are automatically represented as associations, for example of type "is a part of". To retain these relationships, be careful when defining mappings for other types of associations to not then set "keep unmapped associations" to false.

Attributes from associated objects can also be used, as shown below:

```
<Object>
      <Conditions>
        <Attribute name="DrawingItemType" pattern="Connector" />
      </Conditions>
      <IncludeAssociatedObjectAttributes type="[a-z]{3} [a-z]{7}" refID="IAOA1">
        <Conditions>
          <Attribute name="ItemProperty" pattern="PipingMaterialsClass" />
        </Conditions>
      </IncludeAssociatedObjectAttributes>
      <IncludeAssociatedObjectAttributes type="has dataset" refID="IAOA2">
        <Conditions>
          <Attribute name="ItemProperty" pattern="[A-Z]{1}[a-z]{2}[A-Z]{1}[a-z]{7}[A-Z]{1}
[a-z]{1}" />
        </Conditions>
      </IncludeAssociatedObjectAttributes>
      <IncludeAssociatedObjectAttributes type="has dataset" refID="IAOA3">
        <Conditions>
          <Attribute name="ItemProperty" pattern="NominalDiameter" />
        </Conditions>
      </IncludeAssociatedObjectAttributes>
  <ObjectID value="[IAOA1:ItemValue]-[IAOA2:ItemValue]-[IAOA3:ItemValue]">
</Object>
```

For example, the above configuration might result in an Object, with an attribute named `"DrawingItemType"` and value of `"Connector`", having an `ObjectID = "A3B-6-150 mm`", where:

- `[IAOA1:ItemValue] = "A3B"`, from the value of the `"ItemValue"` attribute belonging to the associated Object whose association relationship is matching the pattern `"[a-z]{3} [a-z]{7}"` and its attribute `"ItemProperty"` has the value `"PipingMaterialsClass"`.
- `[IAOA2:ItemValue] = "6"`, from the value of the `"ItemValue"` attribute belonging to the associated Object whose association relationship is matching the pattern "has dataset" and its attribute `"ItemProperty"` has the value that is matching the regex `"[A-Z]{1}[a-z]{2}[A-Z]{1}[a-z]{7}[A-Z]{1}[a-z]{1}"`.
- `[IAOA3:ItemValue] = "150 mm"`, from the value of the `"ItemValue"` attribute belonging to the associated Object whose association relationship is matching the pattern `"has dataset"` and its attribute `"ItemProperty"` has the value `"NominalDiameter"`.

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 add one or more search blocks. Search is a sub-block where you 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.

After 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.

Notes:

- For tags associated with graphics, if the Search criteria matches then as many new tags as matched will be created, each inheriting the same attributes and associations as the original, but the original tag is then deleted. This is to ensure hotspotting by the new tags then works as expected.
- When using SearchCriteria successful pattern matches are not included in the annotated logs if the annotations level is set to **ObjectIDTracking** as this function only applies to BaseMapping matches. Hence, they also won't appear in CSV Reports of matched patterns, see **Appendix B: CSV Reporting** and **Appendix C: Tracking Changes in Data.**

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 present below:

```
<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. 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>
```

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 AutoCAD 2D models, the only hierarchy is objects within a block and between blocks which can be nested. The Gateway automatically creates the relevant "`is a part of`" associations between these objects.

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

This hierarchy is derived from the input file but is lost when graphical objects are grouped, see Patterns 2D.

Remove Non-Essential Layers From Drawing

Note: The MicroStation 'level' values are extracted as 'layer' attributes, hence, the following transformation applies to both AutoCAD 2D and MicroStation 2D drawings.

To remove non-essential layers from drawing, use the `<Remove>` tag but target object has to be found by Association relationship. The following example shows how to use `<Remove>` tag to remove layer by its attribute value.

```
    <Object>
      <Conditions>
        <Attribute name="layer" pattern="Non_Plot" />
      </Conditions>
      <Remove />
      <Associations>
        <Association relationship="GraphicalObjectAssociation">
          <Remove includeTarget="true" />
        </Association>
      </Associations>
    </Object>
```

Presentation Mapping

Presentation Mapping is used to adjust the output SVG and EIWM properties such as `materials` and `colors.` Use of the Presentation mapping extension in the main Transform Configuration file is optional.

The following example shows all the parameters of presentation mapping configuration.

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 2D 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 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 colors in *AIM Dashboard*. The syntax contains two ways to map colors: `fromAttribute` and `fromColour`. It is possible to map colors from any attribute value or specific `colour` attribute value to a `toColour` attribute.

The `fromColour` method maps the input color to a new color.

Example:

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

The following methods can affect the coloring used in Gateway renditions:

- RGB, for example, 128, 0, 128.
- Known colors (HTML color standard), for example, Red.
- HTML hexadecimal: `#E3D3D3`.
- Wildcard "*" in `fromColour` attribute to map all colors not affected by previous Colour mapping items.
- AutoCAD 2D DWG AutoCAD Color Index (ACI) mapping colors can be specified, for example,

  `<colour fromAttribute="autocad colour index" fromValue="71" toColour="red"/>`

- MicroStation 2D DGN Color Table (DCT) mapping colors can be specified in the `fromColour` attribute in the format `DCT<colourIndex>`, for example`,`

  `<colour fromColour="DCT170" toColour="blue"/>`.

Note: When using the extractor for 'Smart P&ID', the '`smconstant`' color definition system has been replaced by the standard RGB system, which should therefore be used for color mapping. Prior configurations in the Gateway Configuration Tool that specify '`smconstant`' mapping will be upgraded automatically to the equivalent RGB mapping, as `smconstant = R + G x 256 + B x 65536`.

Text Mapping

This section describes about the transformation settings defined for manipulating the Text objects in the 2D Graphics.

Text Mapping is used in 2D drawings to customize textual objects visible in the drawing, either by varying the font that is used or mapping specific characters/text strings to alternative values.

Note: Unlike Base mapping, text mapping changes the displayed content of the output drawing.

Example:

```
<textMapping xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://
www.w3.org/2001/XMLSchema"
sourceProductName="AVEVA™ Gateway for 2D Data" componentName="TextMapping"
componentVersion="2.10.0.0"
  <texts>
    <text from="1" to ="&#177;" font="ANSI_SYMBOLS" />
    <text from="%" to="^" font="*" regEx="false" />
    <text from="colour" to="colour" font=".*" regEx="true" />
 <text from="1024" to="0x400" font="*" />
 <text from="0x400" to="1 kilo" font="*" />
</texts>
   <fonts>
    <font from="Courier" to ="Arial,Lucida Console" />
    <font from="SansSerif" to="Lucida Console" italic="false" bold="true"/>
    <font from="#missingFonts#" to="Consolas"/>
  </fonts>
</textMapping>
```

Text Mapping Node

Text mapping exchanges specific characters or strings in text items. For example, adjust them to different mapping of character in fonts, exchange special characters not handled properly by a specific browser or even switch the language.

'text' attributes:

- `regEx`: If set to true, then values for "`from`" and "`font`" should be specified in Regular Expression format.
- `font`: The input font of the character or string to be changed. This can be used to select only the string values that are defined in a specific input font. If the font name is specified in RegEx format, then specifying a value ".*" will affect all graphical text labels with any font. If "`regEx`" is set to "`false`", specify "*" character to cover text items of any font.
- `from`: The input character or string to change.
- `to`: Replacement character or string to be used in the output SVG.

Note: In "`from`" and "`to`" values, the character can be specified in Unicode format as below: "`&#<code>;` "

Example code: "`&#177;`"

The output SVG would replace the input character with the plus-or-minus sign "±".

In the <`texts`> node, the following mappings are processed:

- Characters in the `ANSI_SYMBOLS` font will be searched for the character '1' which will be replaced with the unicode character ="`#177`", used to represent the ' ± ' sign.
- The "%" characters in any font or with no font will be converted to "^".
- All instances of the string "colour" in any font will be converted to "colour".

Multiple text mappings are applied in order to each text item that matches the mapping, even when that text has been updated by a previous mapping. For example, if an input text is "1024 colours", then the above mapping would result in an output text "1 kilo colours" as the third text mapping matches to "colour" and changes the text to "1024 colours", the fourth mapping then changes that to "0x400 colours" and the fifth mapping changes that to "1 kilo colours".

Note: To properly map SHX fonts, you need to include the 'shx' extension in the font name. If an extension is not provided, the Gateway assumes it is a TTF font. Mapping SHX to SHX or SHX to TTF is allowed. TTF to SHX mapping is not supported.

Examples:

```
   <font from="simplex.shx" to="complex.shx"/> (SHX to SHX mapping)
   <font from="simplex.shx" to="Arial"/> (SHX to TTF mapping)
```

Use regular expression, for example:

```
<font from=".*.shx" to="Times New Roman" regEx="true" />
```

Text mapping is not applied to the text if it is using SHX font because it is vectorized directly to graphics and cannot be modified in this form. To use Text mappings on SHX fonts, ensure they are first mapped to a TTF font.

Be sure that the Gateway has access to all required SHX fonts defined in the drawing and in the configuration, noting that if SHX to SHX font mapping is used then only the target SHX font is required. They must exist in one of the Font and Support Folders, see Extract Settings for more details regarding setting the locations of fonts.

Mapping of SHX fonts does not support 'sizeFactor', 'italic' nor 'bold'.

Font Mapping Node

When the input drawing uses fonts that are not available when viewing the SVG, a default font will be used. They may also be mapped to specific fonts, where such fonts are available on the machine where the SVG will be viewed, or converted to polylines when the input drawing's font is available to the Gateway. Some mappings may be specific to the input drawing type, for example, DWG drawings may contain SHX and TTF fonts, while DGN drawings may contain RSC, SHX or TTF fonts.

Font mapping also allows you to change the font size and how it is formatted to be varied compared to the input drawing.

In the <fonts> node, an input font definition of fontfamily="Courier, SansSerif" would be replaced in the output SVG with fontfamily="Arial, Lucida Console, Consolas". This is because the input definition means 'use Courier if available; otherwise, use SansSerif' which is replaced in the output to mean 'use Arial if available, otherwise, use Lucida Console if available; otherwise, use Consolas.'

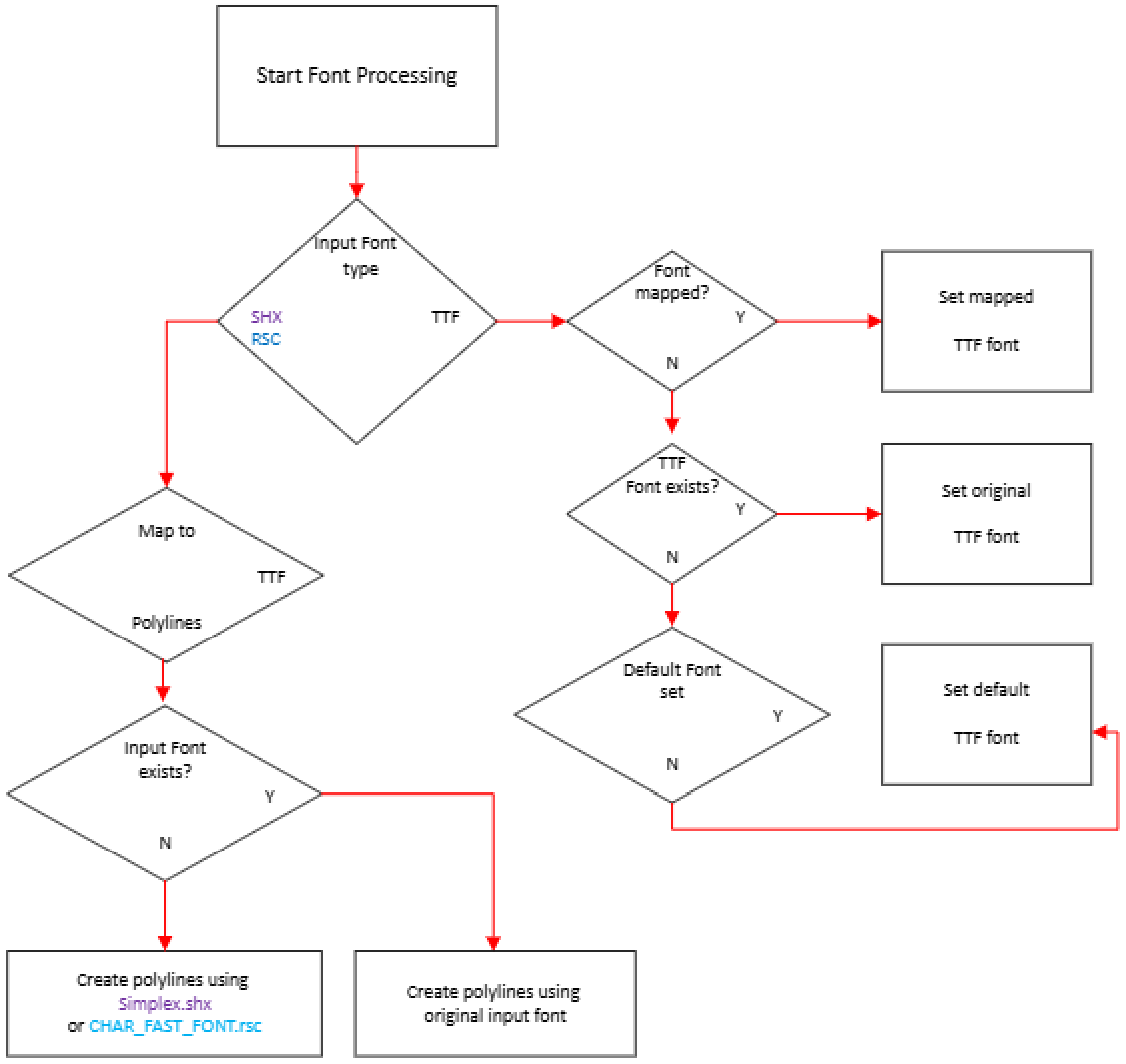
Multiple font mappings are applied to each text item in the order of the mappings until a match is made. Therefore, if the example mapping is used and an input text item has a 'Courier' font, then subsequent font mappings won't be applied to that text as the first font mapping matched its font.

'font' attributes:

- from/to: Maps a font in the input drawing (from) to another font(s) in the output SVG (to). It is possible to list more than one font in the "to" attribute value. When one font is not available, the next font will be used. The font name may be specified in RegEx format.
- from="#missingFonts#": The 'to' value becomes the default value in the output SVG when any of the output fonts are missing. This means that the last font on the prioritized list will be replaced.

- to="#equivalentTTF#": Supports automatic font mapping by the SHX or RSC font name specified in the from parameter, see Automatic font mapping from SHX or RSC to TTF fonts.
- sizeFactor: Used to vary the size of the font in the output SVG. Size factor value has to be between 0.01 and 100.0. Default sizeFactor = 1.
- italic/bold: Used to select italic or bold formats of the font, if available. If not specified, no change is made.

The following Process Flow diagram describes how font mapping and transformation are handled by the Gateway for input files of type DWG, DGN, IGR or PID:



Note: The IGR and PID files only contain TTF fonts.

When mapping SHX or RSC fonts to polylines the machine that hosts the Gateway should have the SHX font(s) installed, so it can accurately render the equivalent shape in polylines.

When mapped to TTF, the user's machine used to view the SVG should have the target font(s) installed, however, this is not required when SHX or RSC fonts are transformed to polylines.

The text size set in the input file applies to the input font. When it is being mapped to a different font, the output text size may be adjusted by the Gateway if the input font shape is a different size to the output font shape. If the input font is not available, the Gateway sets the output font size to be the same as the input font.

Notes:

If the default `Simplex.shx` or `CHAR_FAST_FONT.rsc` font is not available on the Gateway machine, then the input font will be used in the SVG, and when rendered on the user's machine the SVG viewer's default font will be used.

- Where possible choose an output font that has a similar shape to the input font, otherwise, it may not fit the same space as the text in the input drawing.
- For defining the font locations:
  - **SHX font**: For more information about SHX fonts, refer to section "Fonts and Support". If the location of the SHX fonts and shapes is not added to "ACAD" environment variable (it is not automatically added by Autodesk applications), and you want to use these fonts during Gateways processing, then add this path to the list of Paths in "Fonts and Support" Folder group of Extractor's settings.
  - **RSC font:** Add the path to the relevant font(s) in "Fonts and Support" Folder group of Extractor's settings.
    - RSC fonts must be available to the Gateway in the fonts folder, otherwise, the Gateway will not be able to recognize those RSC fonts, so it classifies all missing fonts as `CHAR_FAST_FONT.RSC` by default, and can be further mapped to any desired TTF font, for example:

```
          <fonts>
              <font from="CHAR_FAST_FONT.RSC " to="Tahoma" />
          </fonts>
```

      Note: This is different behavior from SHX fonts, where SHX fonts are recognized regardless of whether the SHX fonts are available to the Gateway in the fonts folder.

    - When relevant RSC fonts are not available in fonts folder and no mappings are available, the Gateway will convert the text to polylines in `CHAR_FAST_FONT.RSC` style.
  - **TTF font**: TTF fonts need to be registered in the Windows system.

Automatic font mapping from SHX or RSC to TTF fonts

It is possible to map SHX or RSC fonts using the #equivalentTTF# parameter, for example:

```
  <fonts>
    <font from="iso123.shx" to="#equivalentTTF#" />
  </fonts>
```

If an `iso123.shx` font was used in the input DWG drawing, or an `iso123.rsc` font was used in the input DGN drawing, then the relevant text in the output SVG file will have its font-family set to '`iso123`'.

Use of this parameter also supports regular expressions being used in the from field:

```
  <fonts>
    <font from=".*.shx" to="#equivalentTTF#" regEx="true"/>
  </fonts>
```

In this case, the Gateway maps all SHX fonts to equivalent TTF fonts. For example, '`romand.shx`' might be mapped to '`romand.ttf`' and '`ukraine.shx`' might be mapped to '`ukraine.ttf`', and so on.

Regular expressions can be used to filter which font families should be mapped:

```
  <fonts>
    <font from="roman..shx" to="#equivalentTTF#" regEx="true"/>
  </fonts>
```

In this case, if the input drawing contained text in `romans.shx`, `romand.shx`, `iso123.shx` and `simplex.shx` fonts, then the resultant mapping in the SVG file would be:

- `romans.shx` text is assigned a font-family romans.ttf.
- `romand.shx` text is assigned a font-family romand.ttf.
- `iso123.shx` text is vectorized to polylines.
- `simplex.shx` text is vectorized to polylines.

Note: The Gateway doesn't check whether the mapped TTF font file exists, it only uses the name of the SHX or RSC font to generate the name of the TTF font-family. If the SVG is viewed on a machine that doesn't have that TTF font then the SVG viewer will use a default font.

Rescale 2D

The transformation extension `Rescale2dTransformation` (see Transform Settings) enables you to adjust/scale the SVG graphics, OLE object resolution and line thickness. This may be needed when the drawing's resolution is not matched with the resolution of the monitor or printer where the SVG will be viewed.

Scale resolution can be provided for three elements:

- **Drawing resolution** – Rescales the entire drawing.
- **OLE resolution** – Manipulates the resolution of embedded elements. This is applicable for both MicroStation and AutoCAD input files.

  Limitation: Raster images in MicroStation drawings that are rotated are not rotated in the output SVG file.
- **Line resolution** – Manipulates line thickness.

Example:

```
<rescale2dTransformation guid="00000000-0000-0000-0000-000000000000"
sourceProductName="AVEVA™ Gateway for 2D Data" componentName="Rescale2dTransformation"
componentVersion="2.10.0.0"  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <resize>
    <drawingResolution factor="1.0" />
    <OLEResolution factor="automatic" />
    <lineResolution factor="2.5" type="text"/>
    <lineResolution factor="automatic" type="lines"/>
  </resize>
</rescale2dTransformation>
```

The three scale resolution elements are as follows:

- `<drawingResolution>` (required) – Rescales the drawing according by the factor, default value is 1.0.
- `<OLEResolution>` (optional) – Manipulates the resolution of embedded objects, either explicitly by a numeric value or 'automatic' (default). The 'automatic' mode derives the factor from the drawing resolution factor.
- `<lineResolution>` (optional) – Sets the line widths used for texts and lines types. Use both type at once, for example, `type="text|lines"` as well as two separate line resolution settings. The factor value is used to

multiply the line thickness. Only text lines generated as polylines (mapped from SHX fonts) are affected by this function and is not applicable to TTF fonts from input files. When set to 'automatic' (default), it derives the value from the drawing resolution factor.

When the drawing scale changes the line scale will also change if both are set to be rescaled, for example you use "automatic" for the line scale.

Note: SVG line thicknesses may be observed to be different when zoomed in compared to the original application view because they automatically update the line weight in real time. Rendering of SVGs does not have this feature, so thin lines appear to get thinner (or thicker) after zooming out (or in). In this case, automatic or manual line scaling in the Gateway can help to ensure the drawing content is still visible.

Chapter 6 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.

Note: If there are more than one million rows to be written then a warning is logged and the CSV output is segmented into multiple files, with an underscore plus index number added to the extra file names to keep them unique, for example: `<filename>_1.csv`, `<filename>_2.csv`, and so on.

- Reporting during transformation configured in Transform configuration file

  This feature generates the CSV report during transformation and customizes the type of the information written to the CSV file. 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>
```

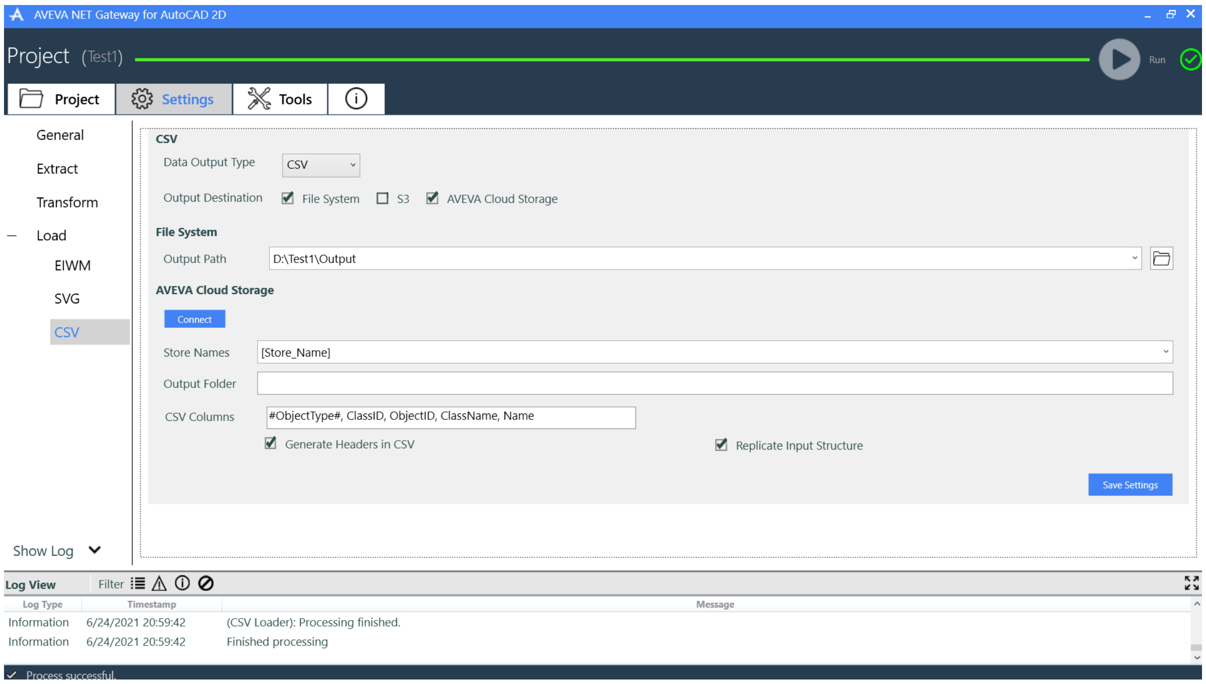
- **Example 2: CSV Reporting section with tracking attributes, from Transform configuration file:**

  ```
  <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>
  ```

Notes:

- 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.
- Multiple attributes and keywords are separated by commas, but multiple parameters for a # keyword are part of the keyword definition and so are separated by spaces or end of line characters, see Example 2 above.
- **Limitation**: Tracking attributes are not generated for expand and interpolate transformation used under ObjectID node.
- Reporting after transformation configured from GUI

  This CSV report is generated when **CSV** value is selected in **Data Output Type** combo list in **Load>CSV** panel. This report reflects the extracted data state after all transformations. Report file with .**CSV** extension is generated in the location set in **Output Path** field.
- GUI Settings in the Load > CSV panel:



CSV

Data Output Type: CSV

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

File System

Output Path: D:\Test1\Output

AVEVA Cloud Storage

Store Names: [Store_Name]

Output Folder:

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

☑ Generate Headers in CSV ☑ Replicate Input Structure

```
Log Type     Timestamp           Message
Information  6/24/2021 20:59:42  (CSV Loader): Processing finished.
Information  6/24/2021 20:59:42  Finished processing
```

✓ Process successful.

-

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 collect internal globally unique identifier (GUID) of the objects by writing as name:

  - #InternalID# and internal source ID reflecting ID of the object in source data format by writing as name: #SourceID#.
  - GUID and SourceID cannot be changed but may be helpful in tracking the history of the objects.
  - • #Attributes – (Optional) adds all attributes of the listed objects to the CSV report.
  - #ObjectType# - Specifies the internal type of the object, for example: EngineeringObject, GraphicalObject2D.
  - #SourceDataName# - Specifies the name of source data like input file.
  - 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 when the column 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. It is ignored when outputFilePath is set.
- 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. It is ignored when outputFilePath is set.
- 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.

- `append = "<true/valse>"`: (Optional) Adds current CSV data to file set by `ouputFilePath`. When option `addHeaderRow` is enabled, then file to append must contain columns with the same names in the header row. If `addHeaderRow` is disabled, then number of the columns in the file must be the same as in the data to append.

  In `append` mode, user cannot provide generic types of columns like `#Attributes` or `#Associations` because they may deliver different list of attributes and associations between files. In `append` mode, list of columns for all processed files is constant.
- `outputFilePath = "<output file path>"`: This attribute must be set when append mode is enabled. It can be provided absolute or path relative to Transform configuration location. When `outputFilePath` is set then attributes: `outputFolder` and suffix are ignored. This setting can be also used with disabled `append` mode.

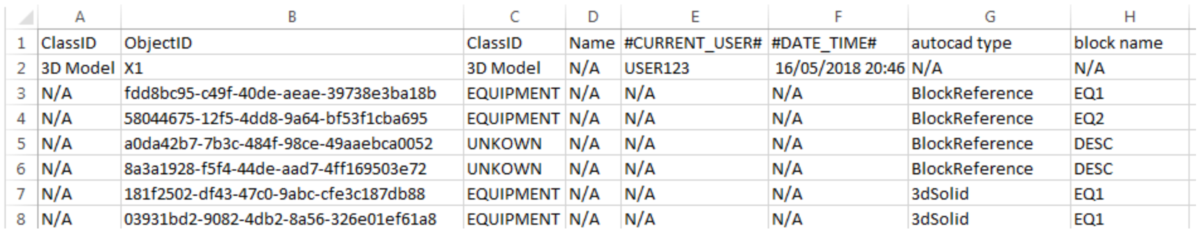
Tracking attributes names which can be used with keyword #Tracking:

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:

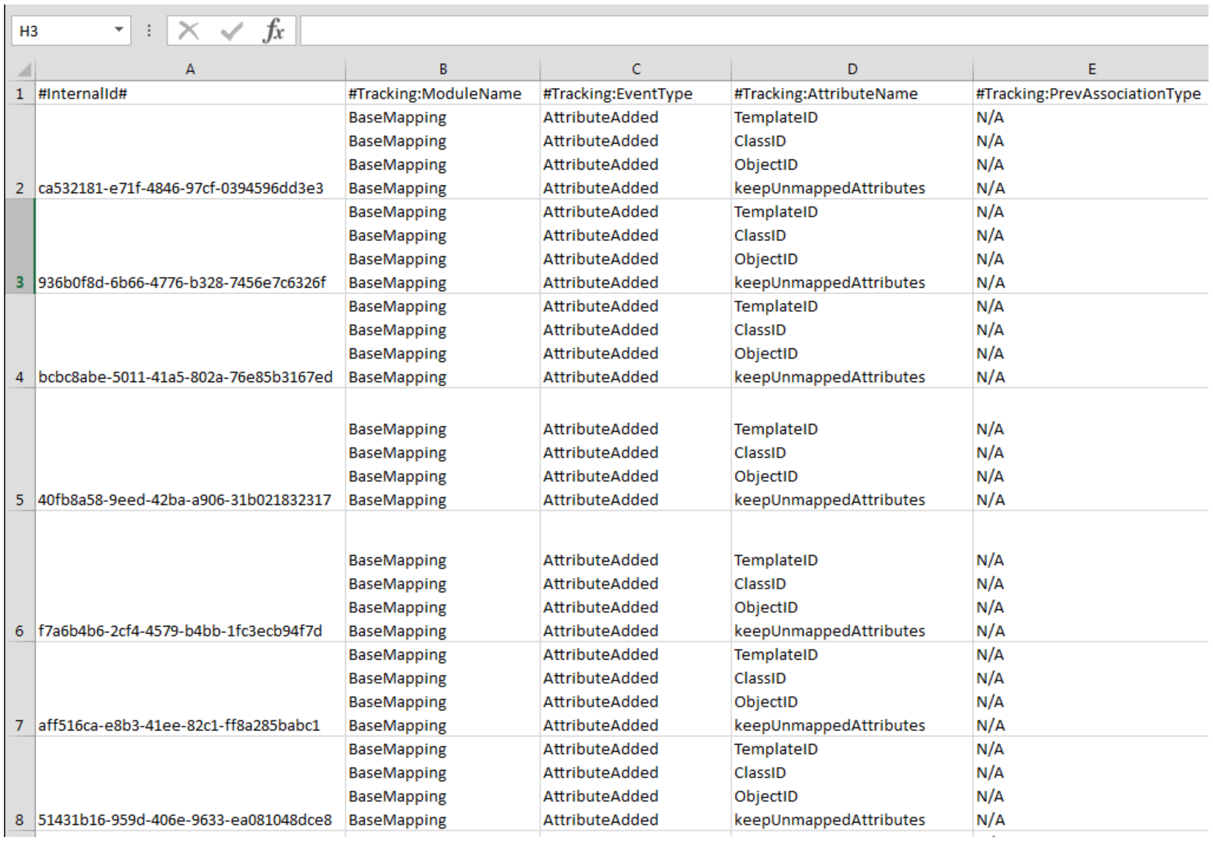
- `ObjectModified`, `ObjectAdded` and `ObjectRemoved`
- `AssociationAdded`, `AssociationChanged`, `AssociationReplaced`, `AssociationRemoved`, `AssociationMoved` and `AssociationsCleared`
- `AttributeAdded`, `AttributeChanged`, `AttributeReplaced`, `AttributeRemoved`, `AttributeMoved` and `AttributesCleared`

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



H3

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

Example 3: CSV Reporting the Base Mapping patterns used for defining ObjectID's

```
<extension name="CSVReporting">
  <csvReport suffix="_AfterTransformation" addHeaderRow="true" columns="ObjectId, Label">
    <column value="#Tracking: AttributeName
                                                   BMExpression
                                                   BMChanges
                                                   #Filter: AttributeName: ObjectID " />
  </csvReport>
</extension>
```

Note: "BMExpression" and "BMChanges" contain the patterns used in the conditions and definition of the ObjectID. These tracking attributes are available to the CSV report when the BaseMapping annotation level is set to "`ObjectIdTracking`", see Appendix C: Tracking Changes in Data.

Chapter 7 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. 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' and 'ObjectIDTracking'.

- When the annotations level is set to `None`, no tracking information is collected.
- When the annotations level is set to `Basic`, then all tracking information connected with creating, updating and deleting performed on objects, attributes and associations are recorded.
- When the annotations level is set to `ObjectIdTracking`, then the patterns that were used in ObjectID conditions and definitions are recorded.

Example for Extractor and Loaders:

```
<configuration ...>
  ...
  <annotations level = "None" />
</configuration>
```

Example for Transformer's Extensions:

```
<extension ...>
  ...
  <annotations level = "Basic" />
</extension>
```

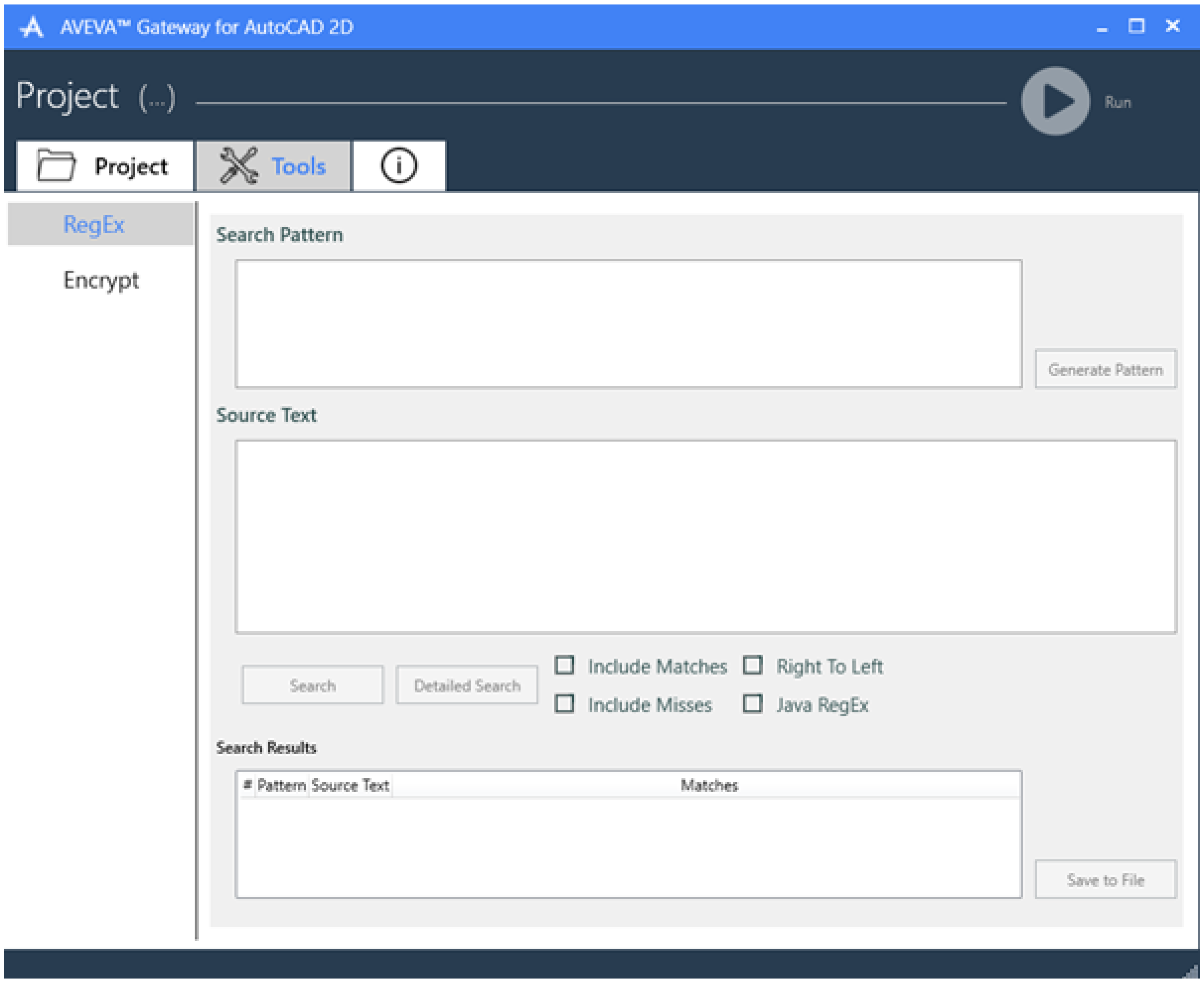
Example of using ObjectIdTracking level in BaseMapping Extension Node:

```
<extension name="BaseMapping">
   <mapping locator="..\Mappings\DefaultBaseMapping.xml" />
   <annotations level="ObjectIdTracking" />
</extension>
```

For more information about passing report of tracking data to CSV, see Appendix B: CSV Reporting.

Chapter 8 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 s**ource 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.

RegEx Utility Options	**User Action**
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.
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.

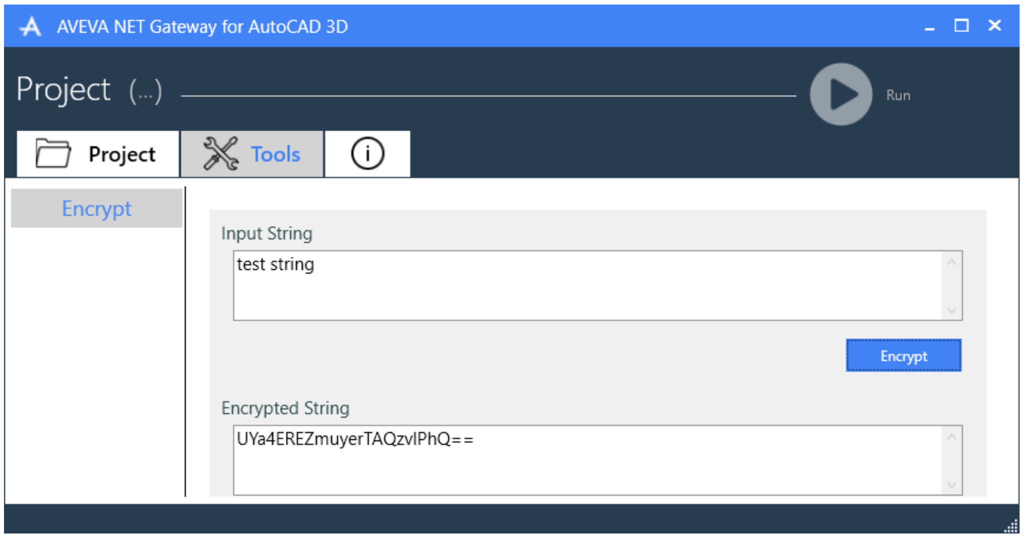
Chapter 9 Appendix E: Encrypt Utility

For the lookup data sources that have connection strings, 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**.

   The encrypted string appears below.
4. Copy the encrypted string and paste it in place of the connection string in all the supported mapping configuration sections.



Input String: test string

Encrypted String: UYa4EREZmuyerTAQzvlPhQ==

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. Get an encrypted password using the encryption utility.

Chapter 10 Appendix F: 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.
- `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.
- `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

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. 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 11 Appendix G: Upgrade from the Gateway Configuration Tool Project Settings

Available upgrades for the Gateway:

- Gateway for AVEVA P&ID -> Gateway for 2D Data / extractor for AutoCAD 2D with AVEVA P&ID option selected
- GCT Gateway for AutoCAD 2D -> Gateway for 2D Data / extractor for AutoCAD 2D
- Gateway for DEXPI -> Gateway for 2D Data / extractor for DEXPI
- GCT Gateway for MicroStation 2D -> Gateway for 2D Data / extractor for MicroStation 2D
- GCT Gateway for SmartPlant P&ID -> Gateway for 2D Data / extractor for Smart P&ID

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 *Gateway for 2D Data,* 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	Rescale2dUnitsDef.xml - drawingResolution Rescale2dUnitsDef.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	Rescale2dUnitsDef .xml - lineResolution type="lines"	Copy value
rscFonts	ExampleTextMappi ng.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 Utitlity" is not upgraded as not supported in Gateway for 2D Data.
viewPort	N/A	Log Warning: Setting: "viewPort" is obsolete.
hiddenLevels	Extract	Copy value
constructioneleme nts	Extract	Copy value
removelowpriority lines	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 <font from="#missingFonts#" to="[font]" />
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,	BaseMapping_[original_GCT_pattern_mapping_file_name].xml	Conversion with respect both GCT and new user guides

separator, modifiers, replacespace, prefix, suffix		
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 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	ExamplePresentationMapping.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 + extractallblockreferencetags (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 <font from="#missingFonts#" to="[font]" />
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 following table lists the full upgrade information with respect to **Smart P&ID** syntax:

GCT Project - Smart P&ID Setting	**New Gateway Target**	**Comment**
digitsInFloatValues	N/A	SVG coordinates have maximum float precision

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>
      <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 12 Appendix H: Handling of 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
	Occurs in Object Type	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
`#KEEP_FOLDER_TREE#`	manifest	R	R	-
`#TARGET_LOCATION#`	manifest	R	R	EXPORT
`#FILE_TIME_STAMP#`	manifest	R	R	-
`InfoLocator`	manifest	R	R	EXPORT
`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 need to 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>
```

The example above will add the engineering attribute named `GRAPHICS_FORMAT`.

Chapter 13 Appendix I: Known Limitations

The following are some of the limitations of Gateway for 2D Data:

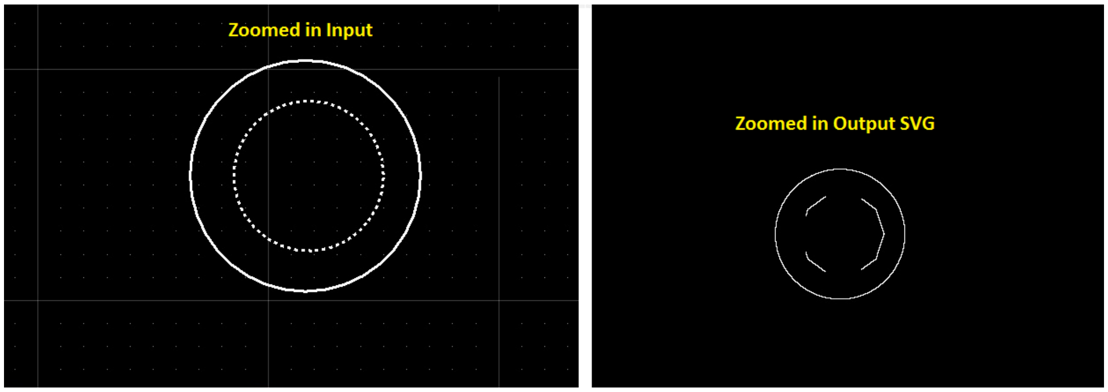
Limitation for MicroStation 2D

The Gateway does not support the extraction of referenced Excel files from MicroStation 2D.

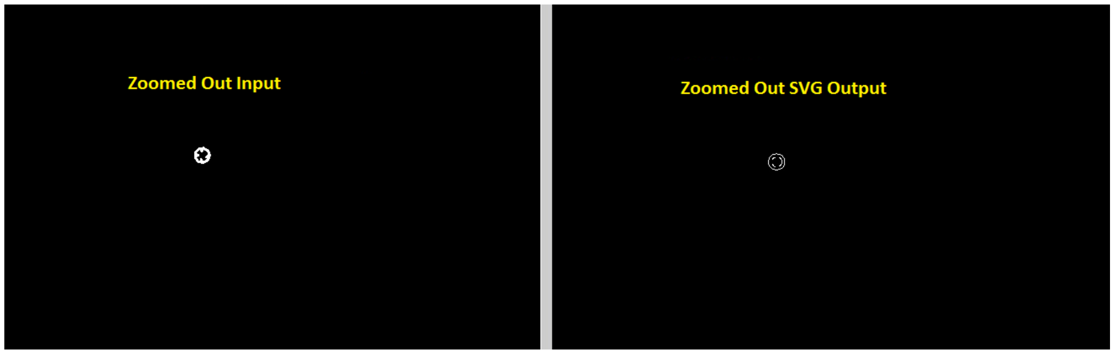
Limitations for both AutoCAD 2D and MicroStation 2D

The AutoCAD and MicroStation applications can scale line styles as you zoom in or zoom out in a drawing. In contrast, the SVG file does not support dynamic line styles.

- A DGN input file containing a circle and dotted circle is zoomed in MicroStation 2D Gateway. Because of the line style limitations in the SVG, the Gateway does not export the input file in SVG accurately. The output SVG shows a broken circle.



- A DGN input file containing circle with four inner lines is zoomed out in MicroStation 2D Gateway. Because of the line style limitations in the SVG, the Gateway does not export the input file in SVG accurately. The output SVG shows a hexagon in place of a circle.



Limitation 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. This also applies to DEXPI extractions.

Extracting from Drawing/Model Issues for AutoCAD 2D Extractor

Input drawings or models are usually not exported correctly if they contain custom/proxy objects. Processing these input drawings/models results in missing graphics in SVG and engineering information for parts represented by these custom/proxy objects.

To process DWG files containing custom/proxy objects:

1. Save these drawings/models to the native AutoCAD DWG format.

   You can use an Object Enabler (program supplied by the creator of the objects), to read (open) these custom objects in AutoCAD. You can also use specific commands of the Object Enabler to convert the proxy objects to native objects before writing out to a new DWG file.

Extracting from Drawing/Model Issues for MicroStation 2D Gateway

It is observed that the processing of a few MicroStation 2D drawings is failing due to the following reasons:

- In some drawings the default view is not set to any of the 8 views and therefore the output is a blank SVG.
- In some drawings there are large entities (with large extents) so that when the default view is '**zoom to extents**' the drawing appears very, very small, so in the SVG the drawing is shown as a very small dot and it is difficult to zoom into it.

To process such drawings using the MicroStation 2D extractor, you need to manually correct these drawings:

1. Open the file in the MicroStation application.
2. Fit the drawing contents manually within the MicroStation level view window or use the 'Fit View' button, whichever is applicable, so that all entities fit within the view.
3. Click **File > Save Settings**.
4. Click **File > Save**.
5. Click **File > Save As** and save as '**MicroStation DGN files**'.
6. Rename the file if needed and click **Save**.
7. The drawing file may now be processed using the Gateway.

SVG Rendering Issues

You may observe performance issues during the rendering of SVG content on Internet browsers while processing large graphics files. After loading the graphics to the SVG viewer, slow response is observed for zoom and pan operations. In some cases, this may lead to a lack of response or a crash of the application. If hardware acceleration is not fully supported by the computers, you may observe these issues.

Hardware acceleration within a browser passes most of the graphics intensive tasks to the Graphical Processing Units (GPU). You can modify the browser settings in the following way to fix the SVG rendering issues:

- Select the **Use software rendering instead of GPU rendering** option under Internet Options, Advanced, Accelerated Graphics section.
- Deselect **Use hardware acceleration when available** option under Internet Options, Advanced, Systems section.

Chapter 14 Best Practices - Drawing and Design Recommendation

This section describes the best practices for drawing and design recommendations in AVEVA Gateway for 2D Data.

Introduction

The aim of this guide is to provide MicroStation and AutoCAD drawing authors with some general guidance on DGN and DWG file creation and technical tips for the quality of hot spotting in order to gain the best results when using the AVEVA Gateway for 2D Data. Drawings that conform to these guidelines will result in better quality hot spotted Gateway outputs and avoid a number of Gateway errors.

The extractors in the Gateway are able to read almost all the data in standard DGN and DWG files, including attributes of cells and reference blocks, to avoid any need to explode them. However, data associated with non-standard objects, such as entities created by proxy extensions to MicroStation and AutoCAD products may not be readable. Once extracted, tags are identified and graphics hotspotted by transformation Patterns 2D and Base Mapping configurations. These functions will be briefly described to provide context to the later sections.

Patterns 2D

The Patterns 2D transformer extension in the Gateway provides a flexible solution for grouping and hotspotting sets of graphical objects with tag text, both multiline text ('mtext') and single line text ('text') to be included in the Group. Various filters can be defined based on the size of the graphical objects, attribute values, number of objects in the group and types of graphical objects to include in a group, as well as how multiple text elements should be concatenated to form a single string for identifying tag(s) to be associated with the group. These tag definitions can be further refined using base mapping.

The following Patterns 2D configuration shows two such filters.

```
  <Object>
    <Conditions>
      <ObjectType ObjectType="symbol, table, oval, all_lines_closed" />
      <Size min="0" max="20" />
    </Conditions>
    <Group Separator="-" TextControlPoint="center" JoinTextFrom="topleft"
IncludeParts="false" GrabTypes="text, mtext" KeepTextOnly="false">
      <IncludeArea AllDirections="1" />
    </Group>
  </Object>
  <Object>
    <Conditions>
      <ObjectType ObjectType="text, mtext" />
    </Conditions>
    <Group Separator="-" TextControlPoint="center" JoinTextFrom="topleft"
IncludeParts="false" KeepTextOnly="false" />
  </Object>
```

The first filter defined by the `<Conditions>` node selects graphical objects of various types (in this case symbol, table, oval, or all_lines_closed) that are smaller than 20% of the area of the drawing. The Gateway classifies MicroStation cells and AutoCAD block references as graphical symbol objects (a full list of Symbol types can be found in the Gateway for 2D Data User Guide). Any text or mtext boxes within these graphical objects' extents are then assigned as the initial tag value. Multiple texts are concatenated in the order of their central positions (`TextControlPoint="center"`) from top left to bottom right (`JoinTextFrom="topleft"`). Concatenated texts are separated by hyphens, as defined by `Separator="-"`. In this example only text that is fully within the group's extent will be part of the group as `IncludeParts="false"`.

The second filter selects graphical objects of types text or mtext that were not grouped by the first filter. These would normally be isolated texts not located within symbols or closed line objects.

Base Mapping

The ObjectID’s derived from the graphics processed by the Patterns 2D configuration can be modified by further base mapping processing. The User Guide section on ObjectID mapping describes this in detail and includes replacing the value with a lookup or another attribute, Transforms functions to keep just the part that matches a pattern or add a prefix or suffix, and so on. Search Criteria can also be used to recursively search through the same text string for multiple tags, as described in the Search Criteria Mapping section of the User Guide..

Drawing Design to Enhance Tag Identification

Each MicroStation or AutoCAD drawing designer creates DGN and DWG files differently dependent upon their requirements and individual content creation technique. The Gateway's extractors for MicroStation 2D and AutoCAD 2D provide many features to identify tags but may be constrained by those original design decisions. It is therefore recommended to consider the following points when creating DGN and DWG files.

Text Boxes within Graphical Objects

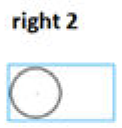
Where graphical objects and text objects need to be highlighted in the AIM dashboard as a single object then it's best practice to create them as a cell or a block reference, heareafter referred to as a symbol.

Example 1: Symbol contains two items of text 'P' and '101' along with graphical objects. Here the complete text would be "P 101". Suitable pattern mapping would match this text and 'P 101' would be the hot spotted text.

Example 2: Symbol contains two text labels 'Text 1' and 'P-101' along with graphical objects. Here the complete text would be "Text 1 P-101". Suitable pattern mapping [A-NNN] that matches any or all of this concatenated string would result in hot spotted text 'P-101'.

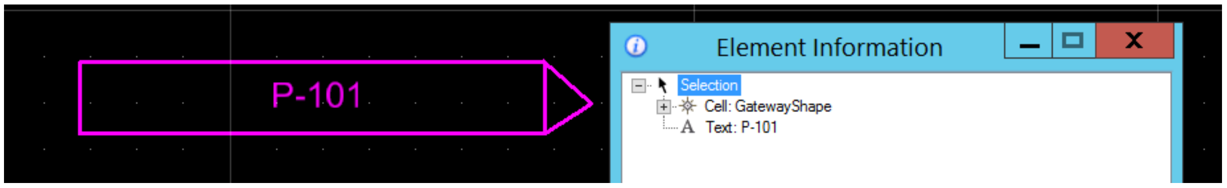
Text Boxes nearby Graphical Objects

If the designer has decided not to include nearby text as part of a symbol, then there are parameters in the Group configuration that enable areas based on the symbol's extents to be used to define which nearby text will be related to the graphical symbol. The IncludeArea parameter directs the Gateway to also include other objects whose position falls within an area defined by the positions of the grouped objects. For example, `<IncludeArea AllDirections="1" />` will include all objects which are within the X and Y extents of the graphics of each of the objects in the group, so if a greater area needs to be searched for nearby text this parameter can be increased, for example, by 10% if set to 1.1. If the designer has been consistent in adding text to say the right of each symbol, then you can restrict the search area accordingly, by using 'right=2.0' which extends the search box to the right.



Independent Text Box Located with Graphical Objects

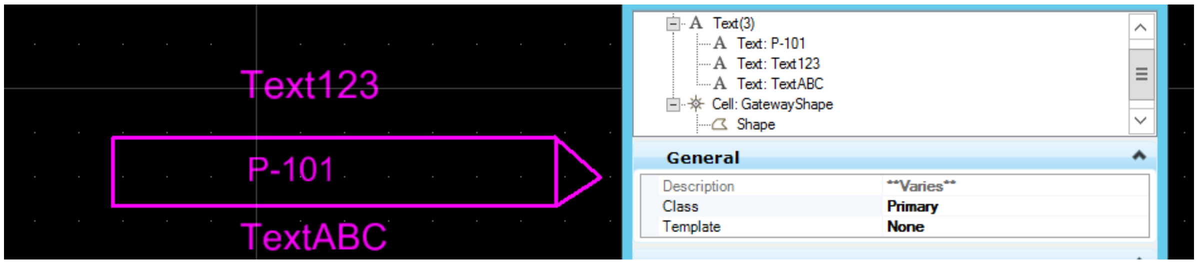
If **IncludeArea** has been configured then if only graphical objects are created as a symbol and a text object is independent, but the text object is placed within the extents of the symbol, the Gateway will group the text with the graphical object as it lies within the symbol object's extent. If pattern mapping recognizes this text, then the matched text will be created, in this case 'P-101'.



Note: Here 'P-101' is independent text which is placed on top of the symbol.

Multiple Independent Text Boxes Located with Graphical Objects

In the following example, two of the extents of the three text objects are outside the extents of the symbol. In this case all of the independent text objects 'text123' and 'P-101' and 'TestABC' can be grouped with the symbol using an `<IncludeArea>` configuration, and any tags derived from that text will result in a hotspot that highlights all graphical elements of the group when the tag is selected.



Description	**Varies**
Class	Primary
Template	None

If the area to be searched as defined in Patterns 2D was top=2 and bottom=2, then the concatenated text assigned to the graphics would be 'Text123-P-101-TextABC', which will be the initial Object ID of the graphical object. If pattern mapping matches this text or any sub-string, for example, with this base mapping configuration:

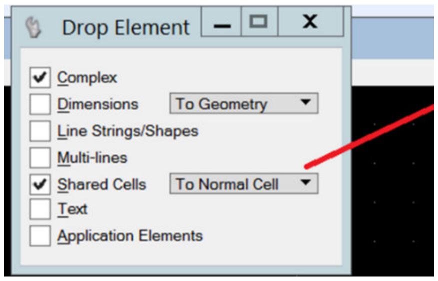
```
<ObjectID>
  <Transforms>
    <Keep pattern="[A-Z]-\d{3} " />
  </Transforms>
</ObjectID>
```

then the tag will be renamed and the matched part of the string text will be created as a hot spot for the graphical object. Here 'P-101' matches the pattern and a mouse hover on the graphical object or any of the three texts will display a highlight value of 'P-101'.

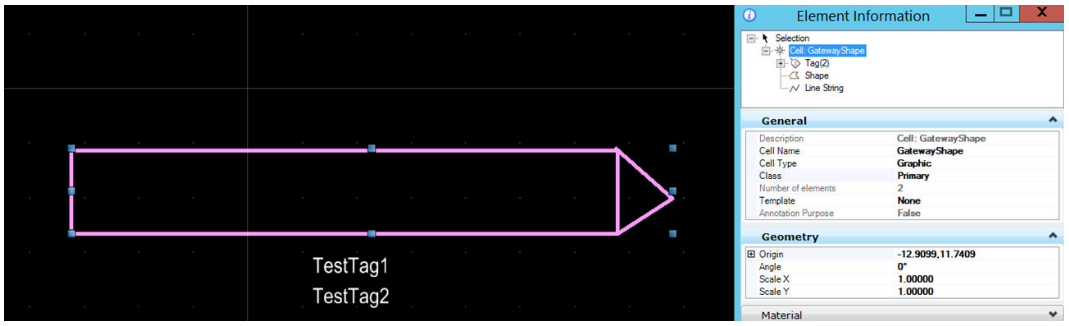
Note: If the nearby text location is not within the search area defined by the Group parameters then the text objects are treated as individual texts, for example, 'Text123', 'P-101' and 'TextABC'. If pattern mapping matches any of these texts, then the matched part of the text will be created as a hot spot for the corresponding text object only. In this example 'P-101' is highlighted when hovering over only the 'P-101' text object.

Unexpected Extents of Symbols

If in a DGN a cell is created with an attached library, it can be converted to a shared cell. The extents of shared cells may be affected by whether some tags in the library have been deleted. This might then impact the tag identification as the undeleted tags might lie outside the cell's extents. This can be fixed in MicroStation by converting the shared cell to a normal cell, which restores the tags in the library and updates the cell with the correct extents.



- [✓] Complex
- [ ] Dimensions — To Geometry
- [ ] Line Strings/Shapes
- [ ] Multi-lines
- [✓] Shared Cells — To Normal Cell
- [ ] Text
- [ ] Application Elements



General

Description	Cell: GatewayShape
Cell Name	GatewayShape
Cell Type	Graphic
Class	Primary
Number of elements	2
Template	None
Annotation Purpose	False

Geometry

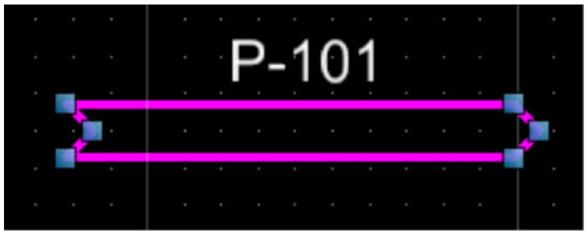
Origin	-12.9099,11.7409
Angle	0°
Scale X	1.00000
Scale Y	1.00000

Material

Independent Text Object is Placed outside the Extents of a Symbol

In this case the Gateway can only associate the text with the graphics if the Group area search parameters are increased, for example, `IncludeArea allDirections="1.1".`

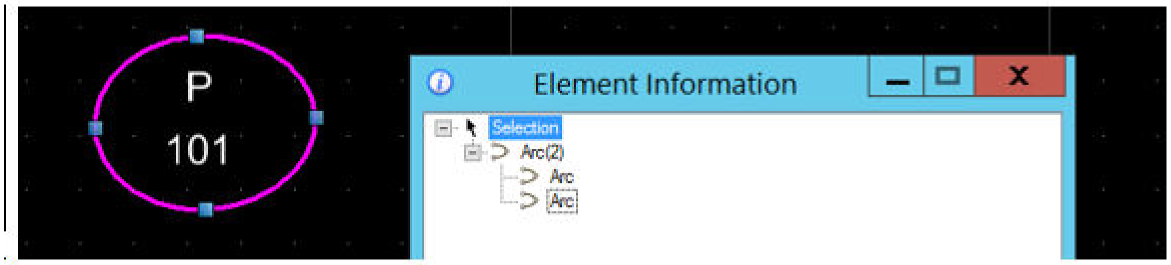
If the graphical object from the below example is a symbol and the 'P-101' text falls within the increased extents of the graphical object, then the 'P-101' text will be associated with the graphical object and will be created as a hot spot when hovering over the extended area.



When Multiple Arcs are used to Create a 'Circle'

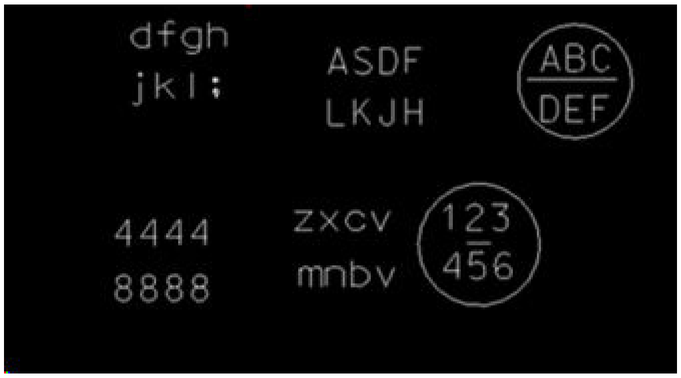
In the below example, the oval looks like it's a single object but is actually four separate arcs. If the graphical objects are not within the same symbol, then the Gateway will not associate the text with the arcs, unless `<IncludeArea>` for arcs or for the text is defined to pick up additional objects near their location.

Recommendation: Instrument bubbles and tag objects should always be constructed as a single fully enclosed drawing object.



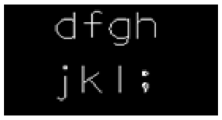
Tag Identification – MultiLine Feature

Multiline text is a text object that can be displayed over more than one line, for example when there are too many characters in the text that it would overlap with another graphical object. Multiline Text can be placed anywhere in the drawing, including inside a symbol or outside a symbol. Consider the following drawing:



which was designed as three separate graphical objects:

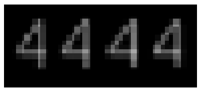
Example 1: Multi-line text:



grouped by the second filter in the Patterns2D example configuration, hence tagged as single line objects

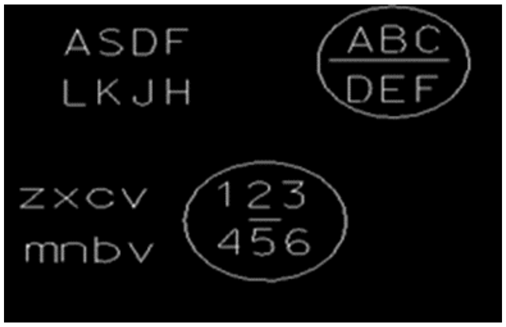
This is multiline text which is not enclosed in any symbol, so both lines are identified and grouped by the second filter in the Patterns2D example configuration, hence tagged as single line objects. A mouse hover on
'dfgh' or 'jkl;' will show 'dfgh-jkl;' as the highlighted text. If subsequent base mapping was applied with a pattern of [AA-AAA] then the tag and highlighted text will be changed to 'gh-jkl'.

Example 2: Single Line text:



This is a single line text that will also be identified by the second filter, so will be captured as '4444'.

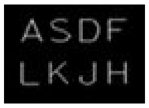
Example 3: All text objects created in a single symbol (for example, Complete_cell):



This was designed as a complete symbol but it's larger than 20% of the drawing size so will not be selected by the first filter and the tagged items from Patterns 2D would be produced by the second filter, as if the symbol had been exploded.

This large symbol consists of nested objects:

- **Example 3.1**: **Multiline Text**

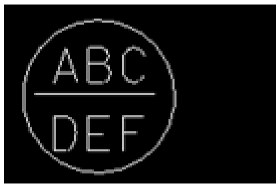
  This is a single line text that will also be identified by the second filter, so will be captured as 'ASDF-LKJH'.

- **Example 3.2**: **Single Line Texts**

  and

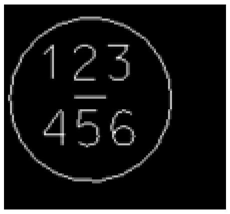
  This is a single line text that will also be identified by the second filter, so will be captured as two tags 'zxcv' and 'mvbv'.

- **Example 3.3**: **Multiline_Circle** where 'ABC' and 'DEF' is multiline text. (one text object)

  Multiline_Circle is a nested symbol that will be caught by the first filter, so when a mouse hover over this area will display a hot spot as 'ABC-DEF'.

- **Example 3.4**: **IndividualLines_Circle** where '123' and '456' are individual text objects.

  IndividualLines_Circle is also a nested symbol caught by the first filter, so a mouse hover in this cell area will display '123-456'.

Individual MText boxes

If mtext is not included in either of the Group definitions, then they will be ignored in Patterns 2D. For the above examples the results would be:

Example 1: This is multiline text so won't be tagged.

Example 2: Single line text so will be tagged separately.

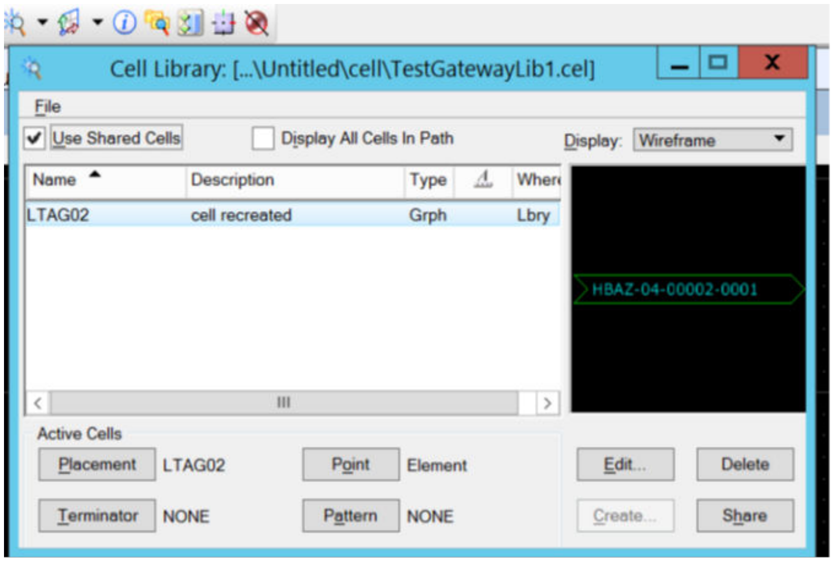
Example 3: ASDF and LKJH won't be tagged but zxcv and mnbv are two normal texts so they will be tagged separately. The **Multiline_Circle** only contains mtext so won't be tagged but the **IndividualLines_Circle** will be tagged with 123-456.

Converting Cells in MicroStation

If tag objects on drawings fail to hot spot it is possible that the construction of the drawing object is different to other objects on the same drawing that do hot spot. The following MicroStation techniques may allow you to change this behaviour.

Convert Normal Cell to Shared Cell

Click on the cell library window from the tool bar. From the menu bar click on **File** and attach the cell library. In the Cell Library window, select '**Use Shared cells'** option which will enable the **'Share'** button at the bottom of the window.


Cell Library: [...\Untitled\cell\TestGatewayLib1.cel]

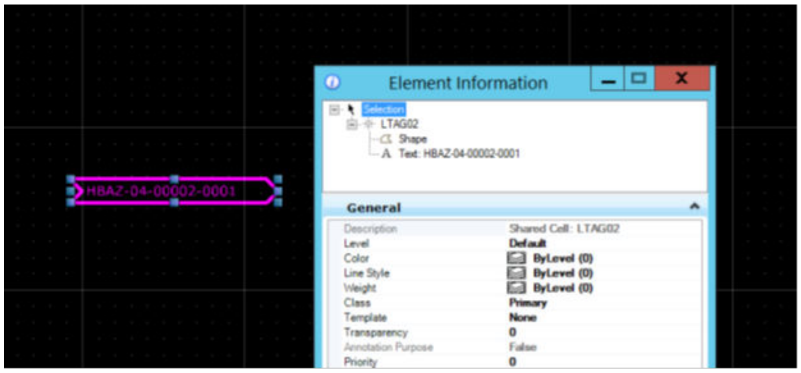
[✓] Use Shared Cells   [ ] Display All Cells In Path   Display: Wireframe

Name	Description	Type	Where
LTAG02	cell recreated	Grph	Lbry

Active Cells

Placement	LTAG02	Point	Element
Terminator	NONE	Pattern	NONE

Clicking the **'Share'** button then converts the cell to a shared cell. This can be verified using the element information option as shown below.


Element Information

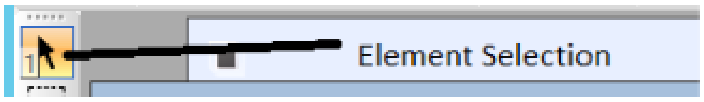
Selection
LTAG02
Shape
Text: HBAZ-04-00002-0001

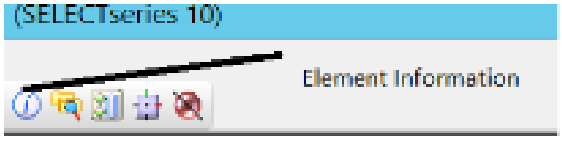
General

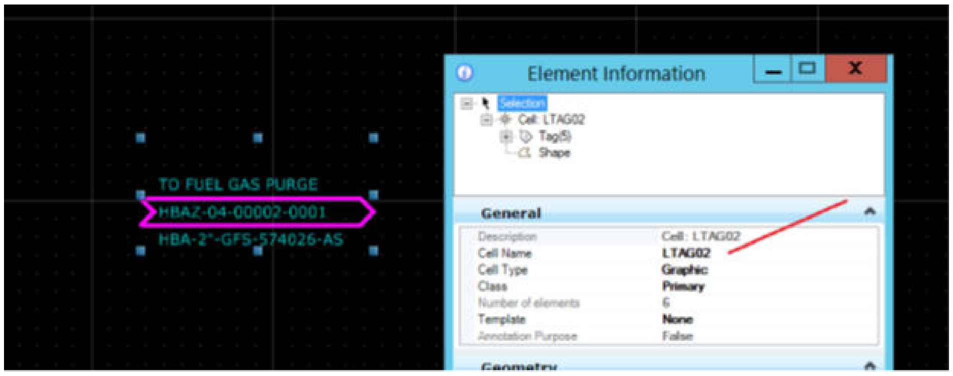
Description	Shared Cell: LTAG02
Level	Default
Color	ByLevel (0)
Line Style	ByLevel (0)
Weight	ByLevel (0)
Class	Primary
Template	None
Transparency	0
Annotation Purpose	False
Priority	0

Drop a Cell to Individual Graphics

Click on the element selection arrow and select the relevant shape, then click on the element information as shown below:

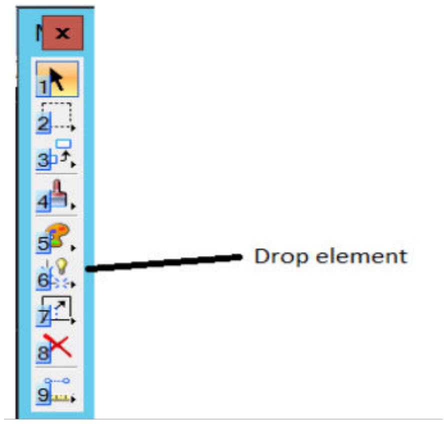


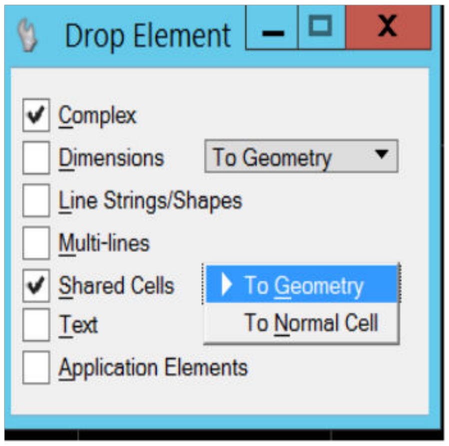




Description	Cell: LTAG02
Cell Name	LTAG02
Cell Type	Graphic
Class	Primary
Number of elements	6
Template	None
Annotation Purpose	False

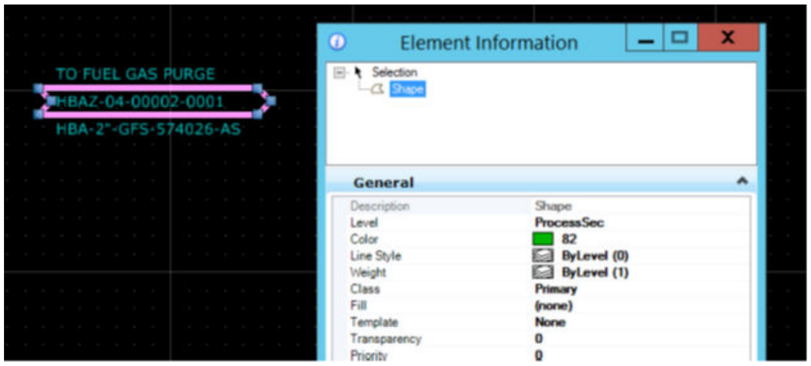
The element information confirms it is a cell. To drop the cell, go to ->**Tools** -> Click on **Main**, which displays the **Main** toolbox. Click on drop element as shown below which opens Enable Shared cells. Select the "To Geometry" option. Then finally click on the shape once again to apply the changes.





[✓] Complex
[ ] Dimensions — To Geometry
[ ] Line Strings/Shapes
[ ] Multi-lines
[✓] Shared Cells — [✓ To Geometry] To Normal Cell
[ ] Text
[ ] Application Elements

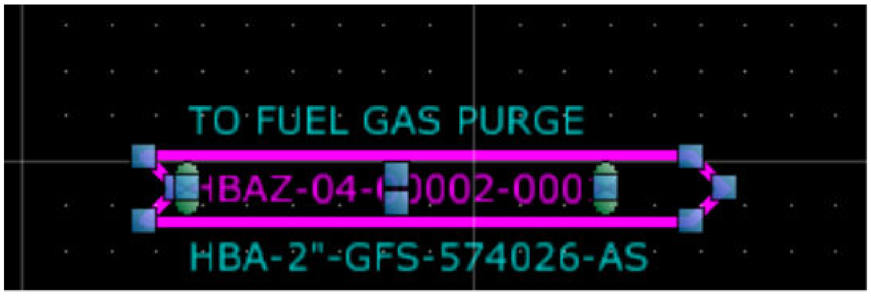
Verify via the element information to make sure the cell is exploded/dropped. If it is exploded then you'll see only shape information as shown below.



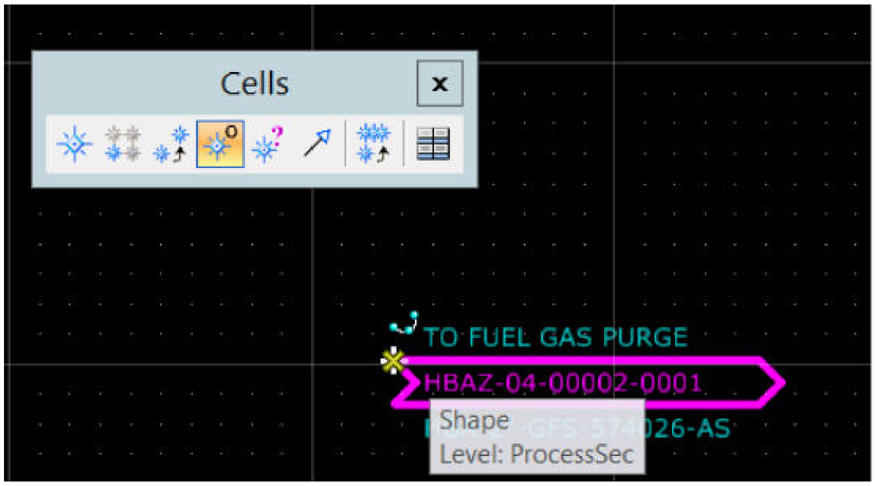
Description	Shape
Level	ProcessSec
Color	🟩 82
Line Style	ByLevel (0)
Weight	ByLevel (1)
Class	Primary
Fill	(none)
Template	None
Transparency	0
Priority	0

Create a Cell from Multiple Graphics

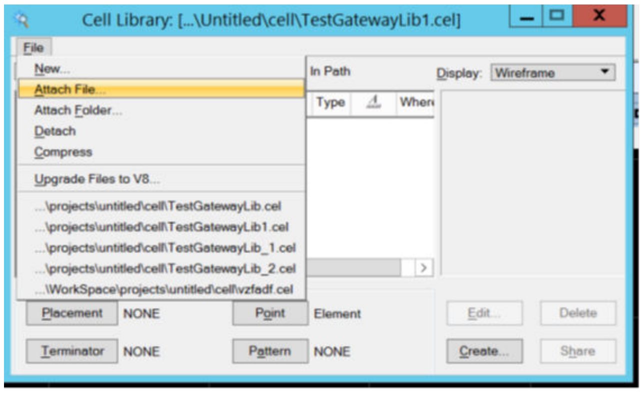
1. Click on the shape, select the text inside the shape and go to **Tools->Cells->Open as Tool Box**.

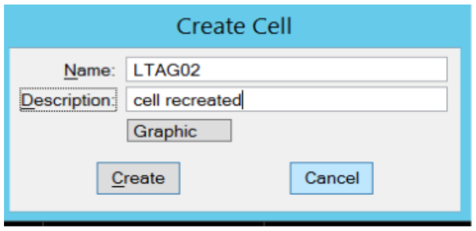


2. Click on define cell origin and click on selected shape to define the origin as shown below.



3. Once the origin is defined click on **Cells** from the tool bar, which will show another window to create a new cell.
4. Go to **File** and click on attach file to attach the cell library.
5. Select **Create** at the bottom of the window to create a new cell. Name it and click on **Create**.


Cell Library: [...\Untitled\cell\TestGatewayLib1.cel]
File
New...
Attach File...
Attach Folder...
Detach
Compress
Upgrade Files to V8...
...\projects\untitled\cell\TestGatewayLib.cel
...\projects\untitled\cell\TestGatewayLib1.cel
...\projects\untitled\cell\TestGatewayLib_1.cel
...\projects\untitled\cell\TestGatewayLib_2.cel
...\WorkSpace\projects\untitled\cell\vzfadf.cel
In Path
Display: Wireframe
Type
Where
Placement NONE
Point Element
Edit...
Delete
Terminator NONE
Pattern NONE
Create...
Share

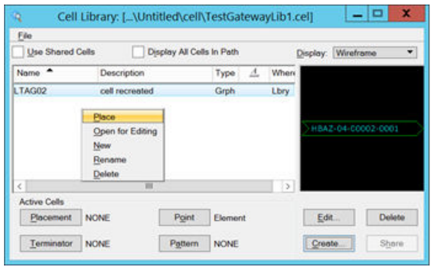

Create Cell

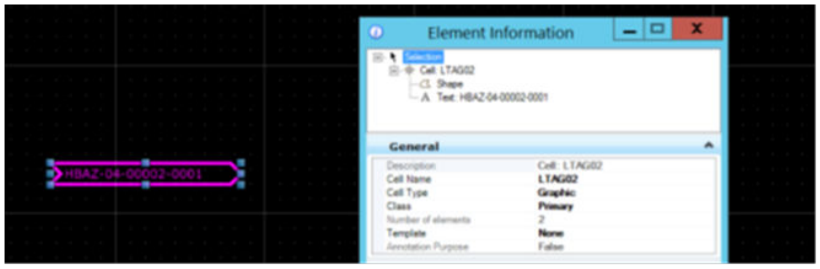
Name: LTAG02

Description: cell recreated

Graphic

6. The new cell is created as shown below. Right click on window to see menu options and click on **Place** menu bar item, place it anywhere in the drawing and view the element information to confirm the cell details.





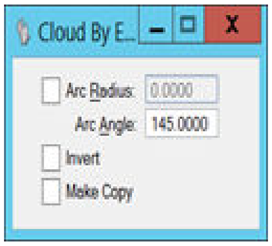
Description	Cell: LTAG02
Cell Name	LTAG02
Cell Type	Graphic
Class	Primary
Number of elements	2
Template	None
Annotation Purpose	False

Creating Cloud Shapes in MicroStation

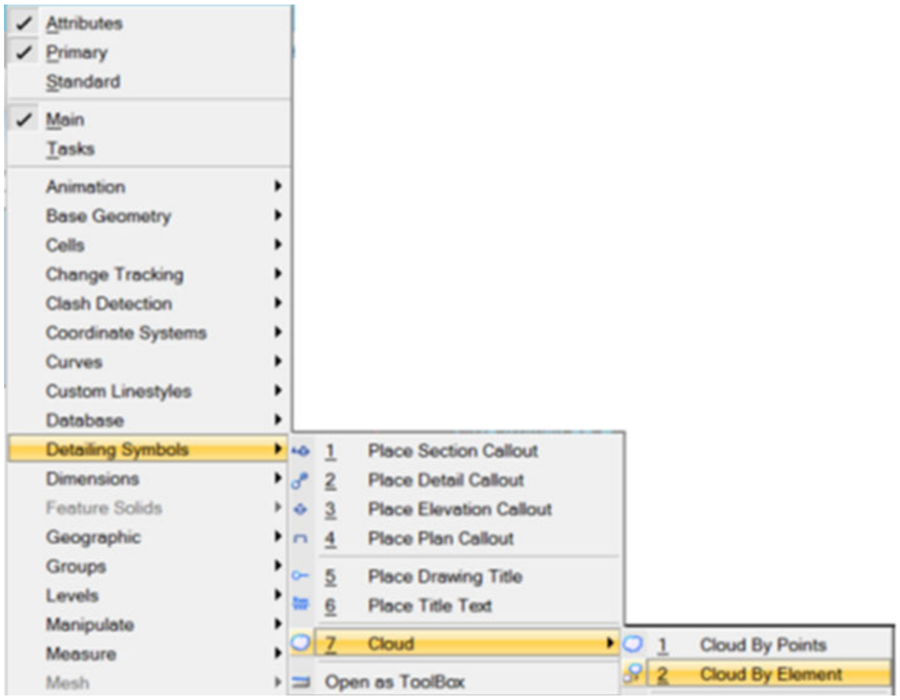
When using cloud shapes in MicroStation files consideration must be given to the Gateway technology which will interpret these in a specific way. The following section provides advice on how these shapes are created and what to do to resolve an issue with a cloud shape to correct any hot spotting issues that may have resulted after the file has been processed through the Gateway.

Creation of Cloud Shape

1. Create any shape in MicroStation and click on menu options **Tools -> Detailing Symbols**.

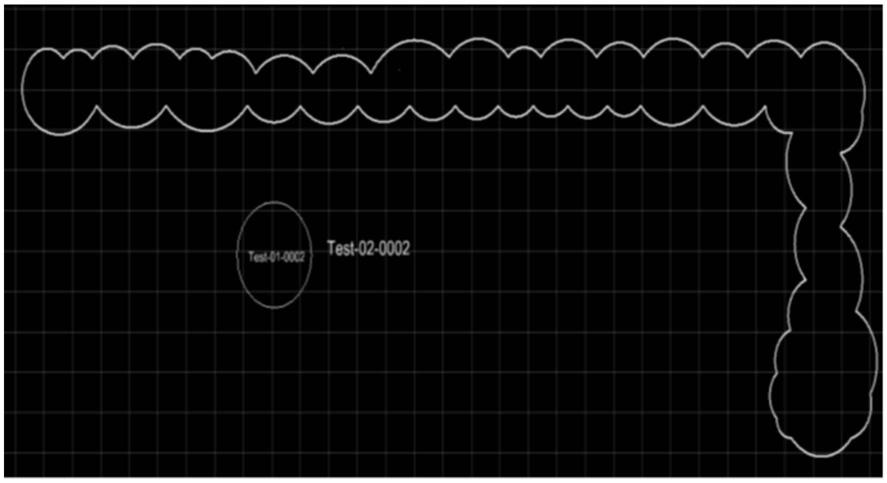


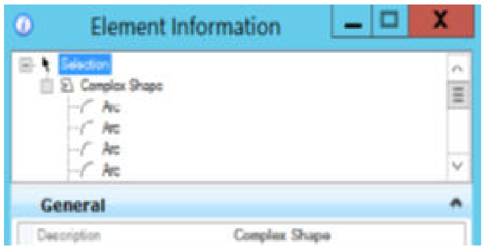
2. Select **Detailing Symbols->Cloud->Cloud By Element**, which will display another window, where you select arc radius and finally click on the shape which you want to convert to a cloud shape. Then the normal shape will be converted to a cloud shape.



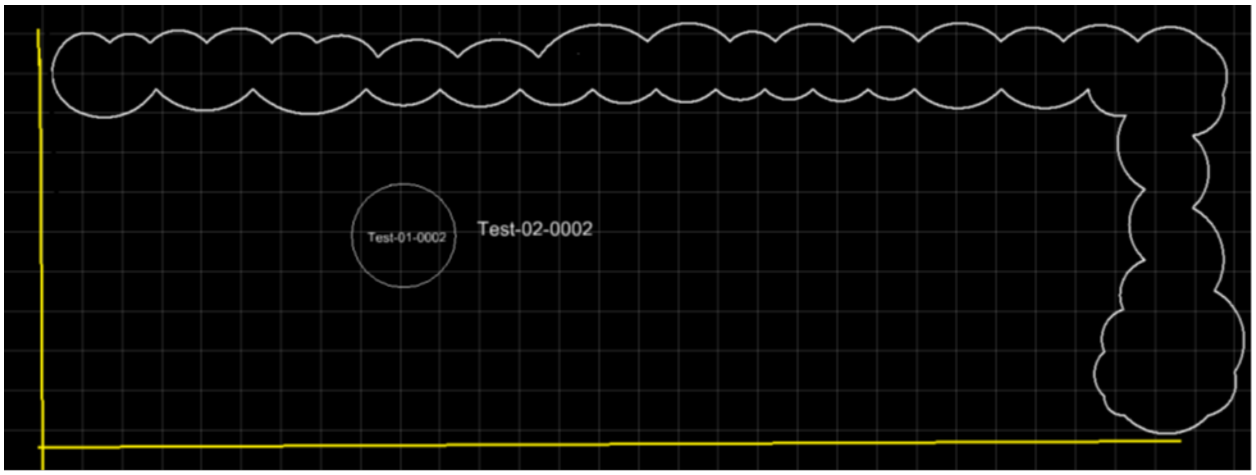
Gateway behaviour for extensive Cloud shapes

In the cloud shape below, any text objects which are within the extents of the cloud shape will be concatenated and created as a single hotspot text associated with the cloud shape. The extents are calculated based on min-x, min-y, max-x, and max-y, as shown below.





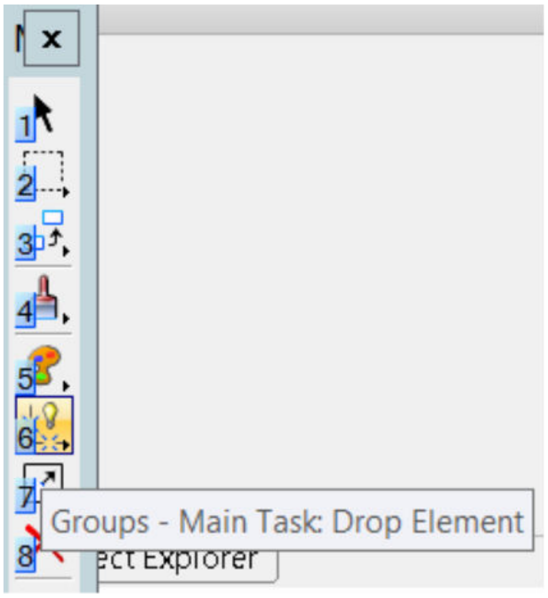
From the following example, the hot spot text will be 'Test-01-0002 Test-02-0002'. This representation may not be the desired output, in which case you'll need to drop the cloud shape and replace it with individual arc elements.



Drop Shape

To drop the Shape:

1. Select the 'Drop Elements' option, go to menu bar **Tools->Main -> drop**.



The cloud shape will now be ungrouped and display as individual graphical elements. Re-processing the drawing through the Gateway will now resolve any hot spotting issues.

Drawing Design Recommendations

When creating DGN or DWG drawings it is recommended that the following conventions be followed:

- Use closed shapes for Tag Objects, that is, Instrument Bubbles.
- Best to include one tag value in each closed shape.
- Create objects to be hot spotted as a symbol (cell or block reference).
- Cloud shapes should be constructed as individual arcs so they are not considered by the Gateway as a single big object whose extent may cover other objects that need to be tagged separately. This will prevent erroneous tag capture.
- Do not replace vectorized object’s symbols by raster images because any text label or attributes which belonged to this symbol will be lost when converted to a raster image, that is, bitmap symbols cannot be properly tagged.
- **Specifically for AutoCAD drawings:**
  - If the Gateway extraction fails due to non-standard (native) DWG format. Try opening the DWG in AutoCAD and if it reports errors, use the Audit function which often cleans up the discrepant/non-standard objects. Resave the DWG before processing again with the Gateway.
  - If the DWG file contains custom/proxy objects they may not be extracted by the Gateway, resulting in missing graphics in the SVG and engineering information for parts represented by these custom/proxy objects. To process DWG files containing custom/proxy objects:
    - Save these drawings/models to the native AutoCAD DWG format. You can use an Object Enabler (program supplied by the creator of the objects), to read (open) these custom objects in AutoCAD. You can also use specific commands of the Object Enabler to convert the proxy objects to native objects before writing out to a new DWG file.
  - Supported versions: DWG and .DXF files up to version 2018 (version AC1032), produced by all versions of AutoCAD up to AutoCAD 2024
- **Specifically for MicroStation drawings:**
  - Scaling should be set to a ‘normal’ level so that when the file is opened in MicroStation it can be viewed without the need to zoom in. Scaling is adjusted by using the Active Scale Settings of the file.
    - In some drawings there are hidden objects that are positioned outside the main drawing, resulting in large overall extents, so that when the default view is ‘zoom to extents’ the drawing appears very small and so the SVG will also be displayed very small. In this case, open the DGN in MicroStation and fit the drawing's visible contents manually within the level view window or use the ‘Fit View’ button, so that all visible entities fit within the view. Re-save the file and reprocess with the Gateway.
  - Supported versions: DGN files produced by MicroStation V7, V8 XM (aka V8.9), V8i (aka V8.11)
  - Most V7 files can be processed without any issues, but if not then they should be opened in MicroStation and saved to a V8 copy.