AVEVA™ Gateway for 3D Data

Contents

Welcome to AVEVA™ Gateways

Welcome to the AVEVA Gateways Documentation!

AVEVA™ Gateways identifies and classifies consistent engineering data and their graphical representation into a format that is compatible with *AVEVA™ Asset Information Management* (AIM), previously known as AVEVA NET.

The Gateway extracts input data from various input sources such as Word, PDF, Oracle, SQL and so on. After transforming these extracted data, it exports them in CSV and intelligent Engineering Information and Workflow Model (EIWM) XML files into *AVEVA™ AIM.*

Benefits of the Gateway:

- Process information from data sources and add it to the common information model, which allows *AIM* to store a unified and cross-referenced model of all tags across projects or assets.
- Scan information sources for tagged items, extract them and automatically build relationships among them.
- Render accurately the source information and apply hotspots to tagged items, which allow you to easily navigate through the related information.

Legal Information

Publication date: Monday, May 25, 2026

Publication ID: 1564293

Technical Support and Training Information

Technical Support

If you encounter software issues or have suggestions for software improvements, contact the AVEVA Helpdesk (https://softwaresupport.aveva.com), and register a support incident.

After you register a support incident, we know your identity (from your customer ID) and the details of the incident, we endeavor to fix the software issue or consider the software improvement suggestion as soon as possible, and notify you about the status of the incident.

If you are an accredited user of the AVEVA products, you are entitled to use the AVEVA Helpdesk, for which you should have valid login credentials. For any other queries, contact the nearest AVEVA Regional Support Center (https://www.aveva.com/en/support).

Training Information

For information about product training courses, see the Product Training section of AVEVA's website (https://www.aveva.com/en/training), or contact the nearest AVEVA Regional Support Center (https://www.aveva.com/en/support).

Contact Information

AVEVA Group plc
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.

Gateways Copyright Information

Disclaimer

1.1 AVEVA does not warrant that the use of the AVEVA software will be uninterrupted, error-free or free from viruses.

1.2 AVEVA shall not be liable for: loss of profits; loss of business; depletion of goodwill and/or similar losses; loss of anticipated savings; loss of goods; loss of contract; loss of use; loss or corruption of data or information; any special, indirect, consequential or pure economic loss, costs, damages, charges or expenses which may be suffered by the user, including any loss suffered by the user resulting from the inaccuracy or invalidity of any data created by the AVEVA software, irrespective of whether such losses are suffered directly or indirectly, or arise in contract, tort (including negligence) or otherwise.

1.3 AVEVA's total liability in contract, tort (including negligence), or otherwise, arising in connection with the performance of the AVEVA software shall be limited to 100% of the licence fees paid in the year in which the user's claim is brought.

1.4 Clauses 1.1 to 1.3 shall apply to the fullest extent permissible at law.

1.5 In the event of any conflict between the above clauses and the analogous clauses in the software licence under which the AVEVA software was purchased, the clauses in the software licence shall take precedence.

Copyright

AVEVA Group Limited, High Cross, Madingley Road, Cambridge, CB3 0HB, United Kingdom.

Trademark

What is new in AVEVA Gateway for 3D Data?

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

This section describes the release rotes for the AVEVA Gateway for 3D Data.

Welcome to Gateway for 3D Data

A full release of *AVEVA Gateway for 3D Data 6.0.0*, which provides a new extractor for Navisworks files and replaces the full releases of:

- *AVEVA Gateway for AutoCAD 3D 5.1.2*
- *AVEVA Gateway for AutoCAD 3D* in *AVEVA Net Gateway Configuration Tool 5.0.11.6*
- *AVEVA Gateway for IFC 5.1.4* and *5.1.4.2*
- *AVEVA NET Gateway for PCF 5.1.3* and *5.1.3.2*

Automated upgrades from each of the above Gateway configurations is supported in this release.

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

Product Name	**Version Number**
AVEVA Gateway for 3D Data 6.0.0	6.0.0

Gateway for 3D Data

This Release Note about *AVEVA Gateway for 3D 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	Microsoft Windows Server 2019 (64-bit)

Prerequisites	Version No.
platform)	Microsoft Windows Server 2022 (64-bit) Microsoft Windows Server 2025 (64-bit)
Mandatory O/S Patches (for each supported platform)	Currently supported patch level, including all critical and security updates provided by Microsoft

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:

- **.Net Framework 4.8**
- **Microsoft Visual C++ Redistributable 2015-2019 (64-bit)**
- **Disk Space** – When the Gateway processes large models it requires disk space to store temporary graphics files. For example, to create a 7 GB ZGL it requires 100 GB of temporary space. Processing times will also be extended if the server isn't using a Solid-State Drive. Note that the size of the ZGL must be < 9 GB when using 3DVS in *AIM*.
- **RAM** - Huge input files may take significant time to process if the memory required by the Gateway process nears the limit of the RAM, as the Windows system will then start memory paging. For example:
  - Input NWD files up to 500 MB will require 32 GB of RAM and NWD files above 500 MB will require 64 GB RAM
  - Input IFC files up to 4 GB will require 64 GB RAM

    Therefore, if you expect to be processing such large files you should ensure the Gateway server has enough RAM to avoid memory paging.

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

Works (is compatible) with

A full release of *AVEVA Gateway for 3D Data 6.0.0* works (is compatible) with:

- *AVEVA Asset Information Management* (AIM - previously known as AVEVA NET) versions 5.1.11 and 5.1.12
- Input 3D files in the following formats:
  - Autodesk AutoCAD 3D .DWG and .DXF files up to version 2018 (version AC1032), produced by all versions of AutoCAD up to AutoCAD 2024
  - IFC STEP files conforming to the IFC 2x3 or IFC 4 schema definitions
  - PCF files created by piping design systems that support the use of Isogen®
  - XGL/ZGL files produced by AVEVA Gateways for UE (aka IE&D), E3D, PDMS, PDS 3D or SmartPlant 3D
    - Navisworks NWD files generated by Autodesk Navisworks Manager versions 2012-2024
- Access Database Engine 2010 or 2016 (if lookups to Excel files are needed)

- Microsoft SQL Server (64-bit) 2019 (if lookups to a SQL Server database are needed)
- Oracle database 12c Release 3 (12.1.0.2.0) (if lookups to an Oracle database are needed)

Supersedes

This release supersedes the previous full releases:

- *AVEVA Gateway for AutoCAD 3D 5.1.4* (release 71418) and *5.1.4.1* (release 71424)
- Gateway for AutoCAD 3D in *AVEVA NET Gateway Configuration Tool 5.0.11.6* (release 71455)
- *AVEVA Gateway for IFC 5.1.4* (release 71383) and 5.1.4.2 (release 71459)
- *AVEVA NET Gateway for PCF 5.1.3* (release 71354) and 5.1.3.2

To Install Product Release

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

Note: Before this *AVEVA Gateway for 3D 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 https://softwaresupport.aveva.com/ as '*AVEVA Gateway for 3D Data 6.0.0*'.

List of Enhancements

The Gateway's functions are described in the online User Guide: https://docs.aveva.com/bundle/gateways

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

- *AVEVA Gateway for AutoCAD 3D 5.1.4*
- *AVEVA Gateway for IFC 5.1.4*
- *AVEVA NET Gateway for PCF 5.1.3*

Incident Number	IMS Number	Story Number	Description
		1710456	All: Excluding objects for CSV and EIWM Loaders
		1735538	All: Support upgrade for ObjectMergeMapping extension
		1980622	All: Add support for relative paths

Incident Number	IMS Number	Story Number	Description
		2372276 2372340 2509191 2509204	New Extractor for Navisworks 3D
		2509145 2853165	All: Support JSON reporting
		2511395	Navisworks: Support for S3 in Extractor and Loader
		2511399	Navisworks: Support for ACS in Loader
		2588152	All: Separate process to control system errors
		2607268	All: Set ACL (Access Control List) for custom location installation
		2885706	All: Use Windows Integrated Security for Database lookups
		2913702	AC3D: Upgrade configurations from Gateway for AC3D and GCT
		2913726	IFC: Upgrade configurations from Gateway for IFC
		2913734	PCF: Upgrade configurations from Gateway for PCF
		3263171	Link objects based on attribute value while creating new associations
		3278268	Navisworks: Warn user about complexity/size of NWD

List of Fault Corrections

This release addresses defects arising from the following support incidents in the previous versions of the AVEVA Gateways:

Incident number	IMS number	Defect number	Description
		1821236	All: `InsertAfter` and `InsertBefore` options in transform are adding extra text
		2360704	All: Extraction process fails after cancelling XGL Loader processing
		2513999 2520752	AC3D: JSON report not generated by using Command line
960313560	2544401	2544666	IFC: No ZGL output created (Failed to import product/geometry)
		2655886	PCF: fails in XGL loader
		2664894	AC3D: fails in Transform
		2885195	All: Cannot load project due to Aveva Cloud Storage in the configuration
960368185	2971678	2972800	IFC: Does not fetch an attribute whose name contains parentheses "( )"
		2997517	PCF: XGL cannot be read
		3002542	IFC: Attributes with "( )" are extracted with additional " ' ' " in their names

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 https://softwaresupport.aveva.com/.

Known Issues

IFC 4 files containing these entities cannot be processed:

- IFCBEZIERCURVE
- IFCRATIONALBEZIERCURVE

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.

Layer attributes missing from Navisworks files derived from MicroStation 3D DGN files

When NWD files are derived from an import of MicroStation 3D models, the 'layer' attributes of objects are not extracted by the Gateway. Navisworks files derived from other 3D sources such as AutoCAD and IFC are unaffected by this limitation.

For the latest list of exceptions and other updates, including those for the for the original full release, please see the *Product Release Latest Update Note* on AVEVA's support web site https://softwaresupport.aveva.com.

Gateway for 3D Data 6.0.0

The Gateway extracts information from AutoCAD 3D .DWG and .DXF files, Naviswork .NWD files, IFC and PCF files and transforms the engineering and 3D graphics into output EIWM and .XGL/.ZGL files. These extracted files can be loaded directly into AIM enabling you to view the information in AIM. The Gateway is a stand-alone application (a single .exe file) 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.

The following are the input file sources from which the Gateway extracts the information:

- .NWD (**Navisworks Documents** files)
- AutoCAD 3D (AutoCAD .**DWG/**.**DXF** 3D models)
- IFC - (Industry Foundation Classes .**ifc** files)
- PCF - (Piping Component Files .**pcf**, .**XGL**, and .**ZGL** files)

These EIWM and .XGL/.ZGL files can be loaded directly into AIM enabling you to view the information in AIM. The Gateway is a stand-alone application that can be deployed independently.

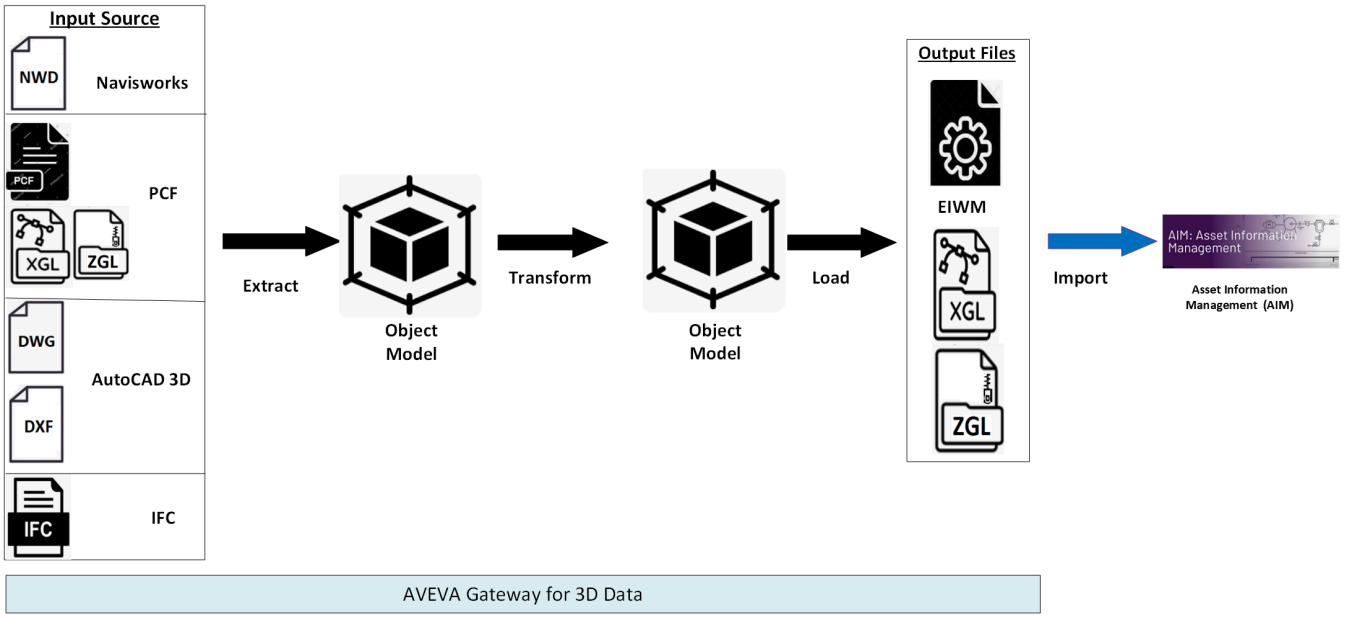
Characteristics of the Gateway

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 3D, Navisworks, IFC or PCF 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.
  - 3D renditions of the drawing graphics and object (tag) hotspots in .XGL/ZGL 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 for the Gateway to function. However, the Gateway may be configured to use the AutoCAD 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:



Get Started

This chapter 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 (RAM)	8 GB	32 GB
Hard Disk Drive	80 GB	150 GB
CPU	2 GHz or faster 64-bit compatible processor	3 GHz

Notes on RAM Usage

Huge input files may take significant time to process if the memory required by the Gateway process nears the limit of the RAM, as the Windows system will then start to start memory paging. For example:

- Input NWD files up to 500 MB will require almost 32 GB of RAM and NWD files above 500 MB will require almost 64 GB RAM.
- Input IFC files up to 4 GB will require 64 GB RAM.

Therefore, if you expect to be processing such large files you should ensure that the Gateway server has enough

RAM to avoid page faults.

Usage of RAM

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

Usage of Virtual Machines

Running the Gateway on a Virtual Machine, even when it has the same hardware specifications as a physical machine, will in general take twice as long to process the same input files, compared to running it on a physical machine.

Software Requirements

You 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)
Visual C++ Redistributable Packages	Microsoft Visual C++ Redistributable 2015-2019 (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.
- The Gateway supports:
  - NWD files generated by Autodesk Navisworks version 2012-2024.
  - PCF (Piping Component File) files
  - AutoCAD 3D files
    - DXF (Drawing Exchange Format) - version AutoCAD 2024
    - DWG (Drawing File): version 2018

Note: The latest version of the DWG file format used by AutoCAD is the DWG 2018 format. This version is in use since AutoCAD 2018 and continues to be the standard for subsequent releases, including AutoCAD 2024.

- IFC (Industry Foundation Classes) files conforming to schema versions IFC 2x3 or IFC 4.

- You need to install ADE if you are using lookups in mapping.
- AVEVA Licensing System (ALS) and Licensing as a Service (LaaS) are no longer needed.

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:

Microsoft Access Database Engine 2016 Redistributable

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.
- If the Gateway is configured to read or write data into a shared location or to a file server, it is recommended

that users enable SMB encryption. This can be done by following the procedure mentioned under "**SMB Security Enhancements**" in the Microsoft learning page.

- 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 `3DDataGatewayInstaller.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 `3DDataGatewayInstaller.MSI /?` to see all options.

Example of quiet installation with passing folder:

```
3DDataGatewayInstaller.MSI INSTALLFOLDER="C:\Program Files\MyFolder" /quiet
```

Uninstall the Gateway

You can 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 3D 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 `3DDataGatewayInstaller`**.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.

Changing and Repairing the Gateway

You can 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 `3DDataGatewayInstaller.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.

Run the Gateway from the Project Explorer

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

execute projects using the **Run** button.

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

1. Type **3D Data Gateway** in the Search box on the taskbar in your computer. Alternatively, you can click the **AVEVA Gateway For 3D 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 3D 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, you can connect to servers and other workstations without mapping to a drive. The UNC path syntax is as follows:

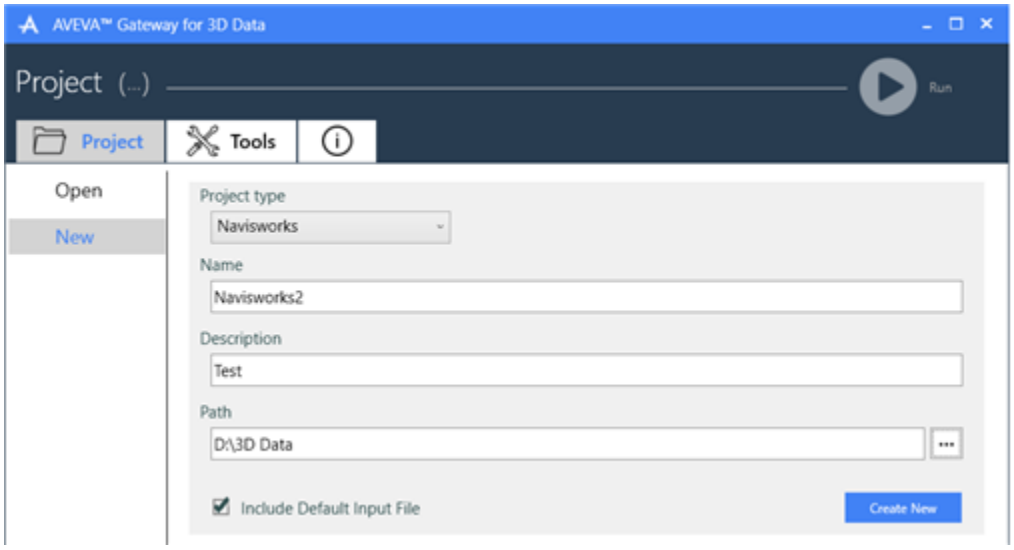
```
\\servername\sharename\directory
```

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

Create Projects

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

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



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

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

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

that the project configuration files are loaded into the user interface.

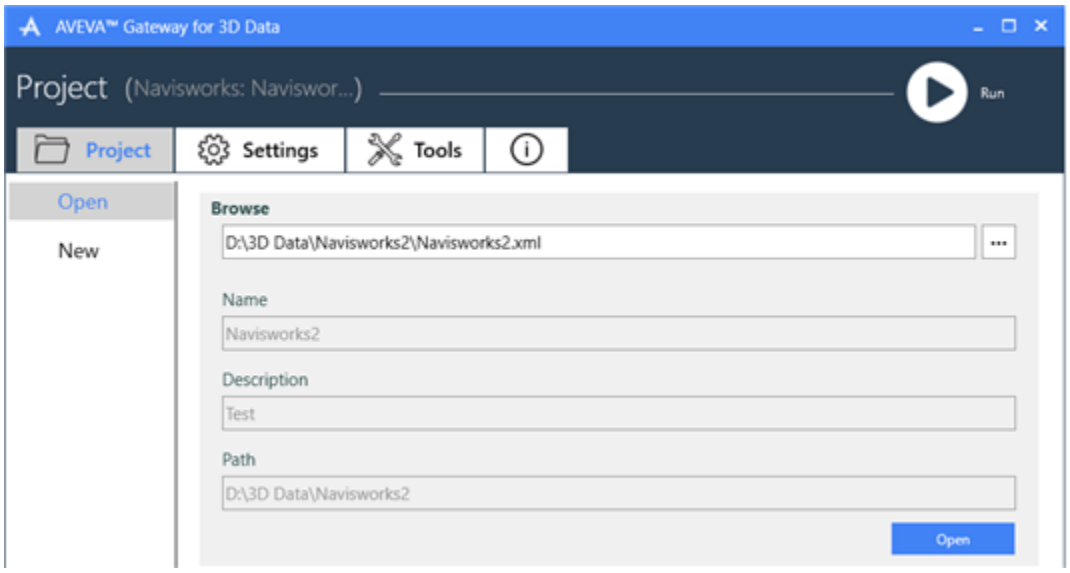
5. Select the **Include Default Input File** box to create some example input files.

Open Projects

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

To open an existing project:

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



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

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

Define the Project Settings

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

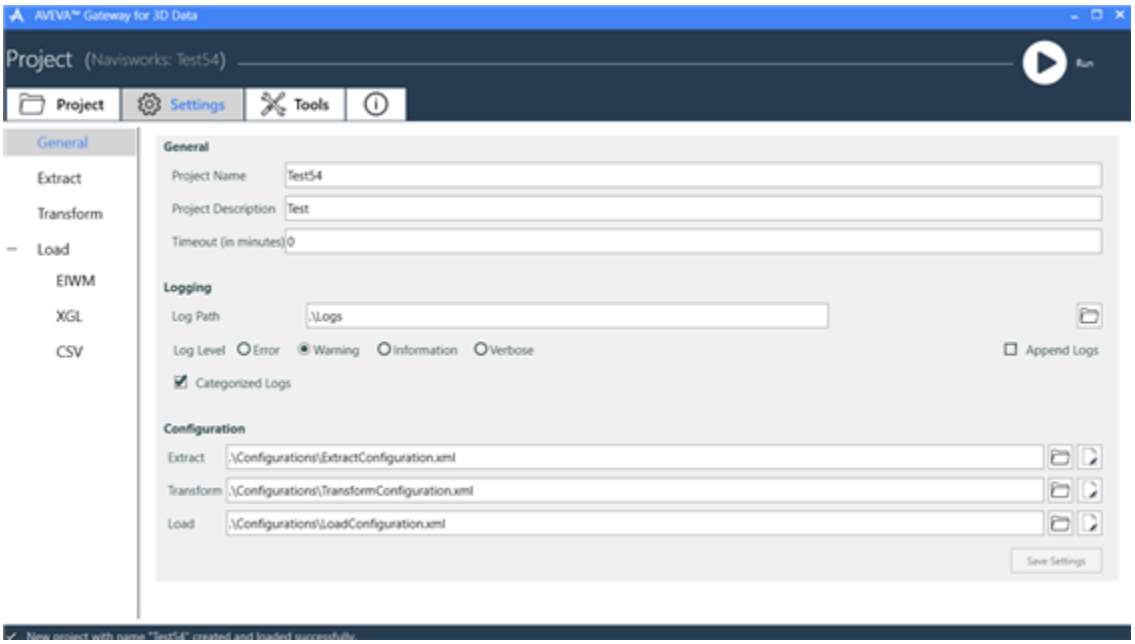
The **Settings** menu contains the following panels:

- **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 panel 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.

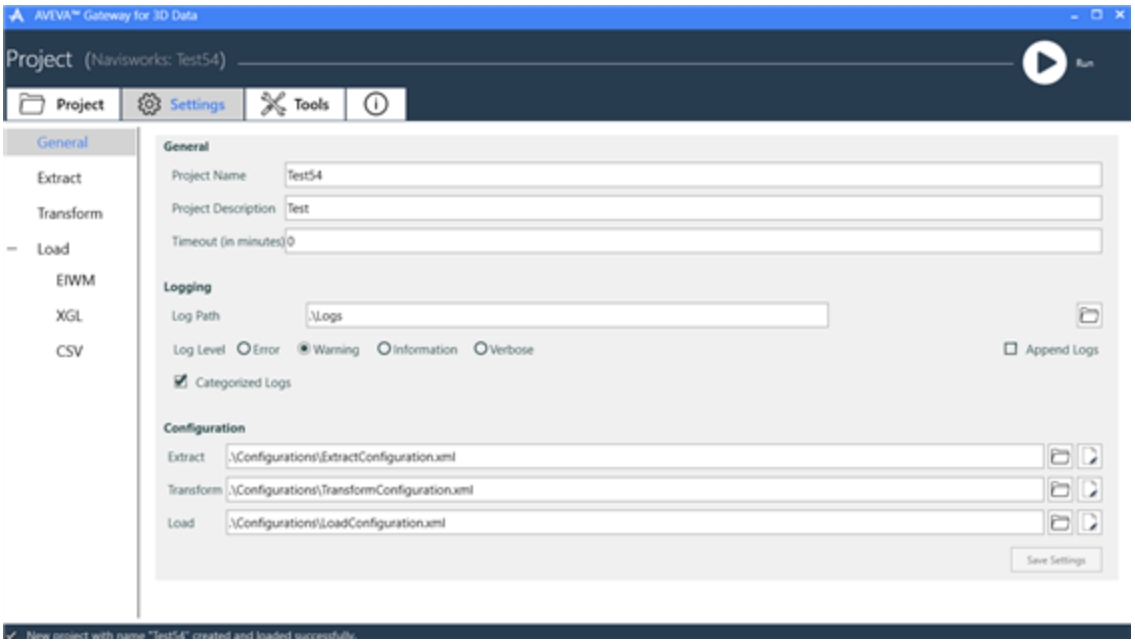
Using Relative Paths in the Configurations

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" />
```

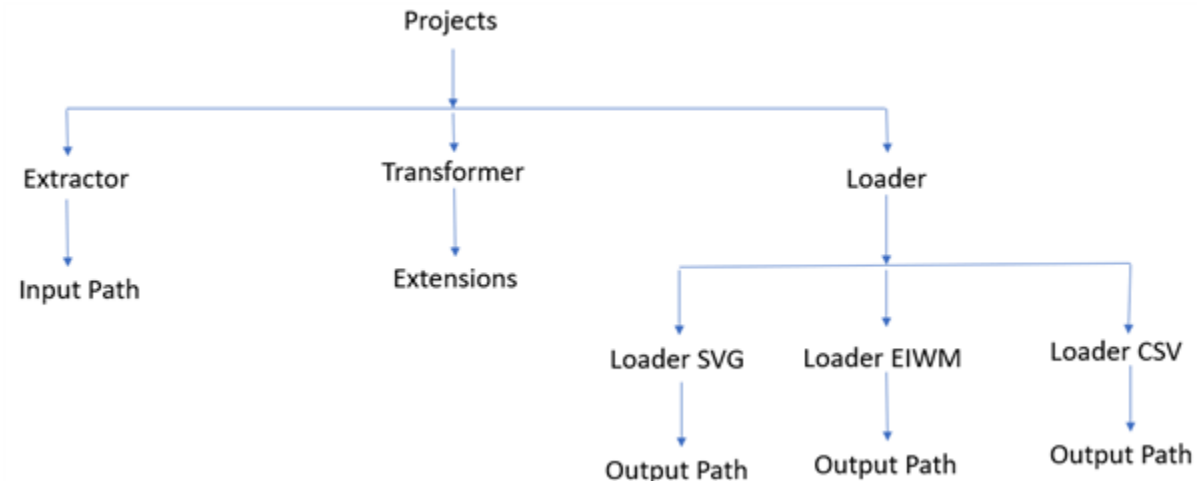
Example of Relative paths in the GUI:



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`

Review from here : https://dev.azure.com/AVEVA-VSTS/Interoperability/_workitems/edit/3947814

General Area

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

To complete the project related information:

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

   Note: The **Project Name** is 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. When processing a folder of more than one file, the timeout applies to the entire processing of all files in the folder. This behavior is modified when Separate Processing (see below) is configured to be true. In this case, the timeout applies to the time taken to process *each file*, and if a timeout fails on one of the files, the Gateway will continue to process any remaining file(s) in the folder.

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

   End Review here

Review from here: https://dev.azure.com/AVEVA-VSTS/Interoperability/_workitems/edit/4346780/

Logging Area

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

If the **Categorized Logs** option is selected the Gateway will also create a JavaScript Object Notation (JSON) report file to display the same logging information as contained in the log text files, but groups the log entries by warnings, errors, severity, type, and so on. For more details about Categorized JSON reports, refer to Status Messages from the Gateway Processing.

Review ends here:

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.
Information	Indicates logs for an error, warning and other informational message. An Information is a message with a concise report of a object processing, timestamp, problem trigger, processing details and working of software modules
Verbose	Indicates logs at all levels, that is, error, warning and informational messages. A verbose is a message that provides additional details about the activity start and end, dump data and the values of the variables and arguments that are used. This extra information can be helpful during troubleshooting, but as it creates large log files this option should be avoided in normal runs.

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

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

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

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

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

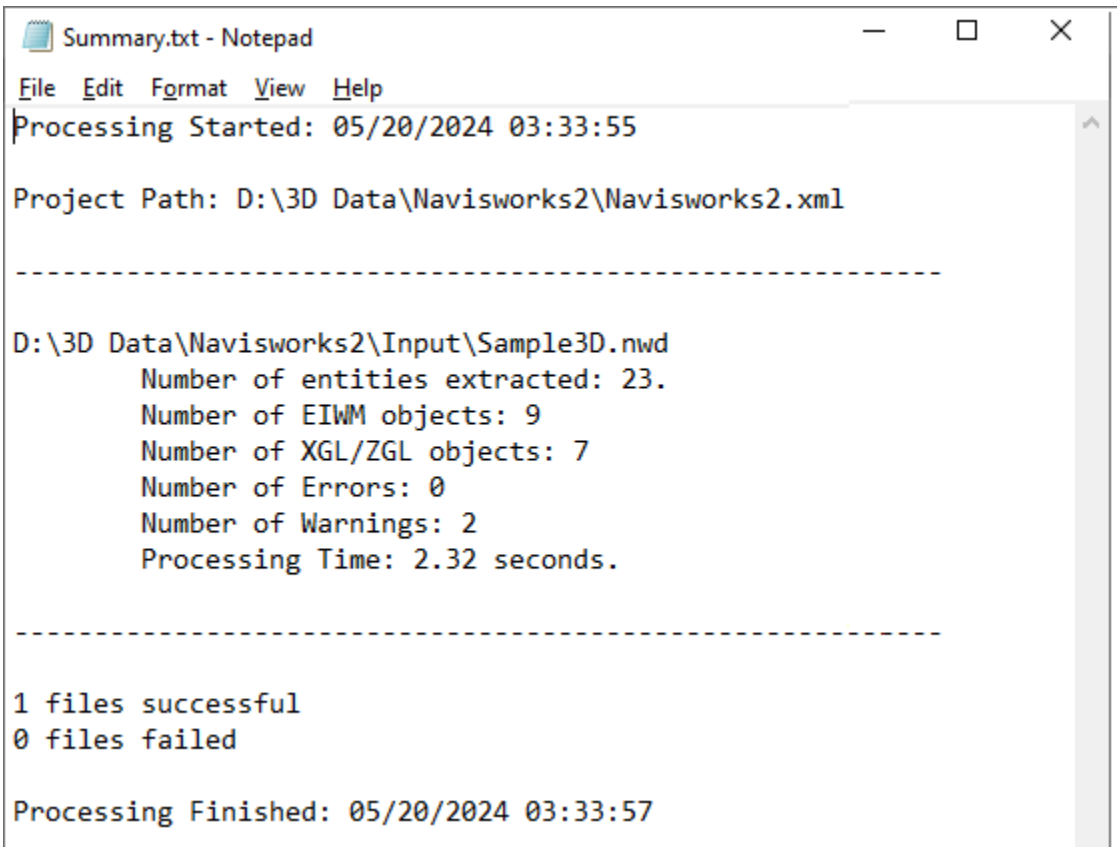
   After processing, the Gateway produces a `Summary.txt` file in the Project Logs folder, which contains statistics about the amount of extracted and loaded objects, project path, file processing information, number of errors and warnings and processing time.

Statistic	Source	Description
Number of entities extracted	Extractor	Number of engineering objects in Object Model after extraction
Number of XGL/ZGL objects	XGL Loader	Number of 3D graphical objects in Object Model that were loaded into the output XGL/ZGL file
Number of EIWM objects	EIWM Loader	Number of engineering objects in

		Object Model that were loaded into the output EIWM file. This may be different from the number of extracted engineering objects if transformation mapping has removed some objects or created new ones.

Example:

Example:



Configuration Area

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

To define the configuration settings:

1. Type the configuration details in the **Extract**, **Transform**, **Load** fields, respectively.
   - **Extract**: Provide the configuration file details to extract the engineering and graphics data from the 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, you can type the path of the configuration file (`.xml` file) or click **File Selector** at the end of the box. To modify the respective configuration files, click  .

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. You can save the settings after filling the required information in **General** settings.

Separate Processing

The processing of large files from Navisworks or IFC sources can often consume too many server resources which causes processing issues. To log these types of issues the Gateway can be run in two separate processes, where one process loads the configurations and then launches a separate process that executes the configurations. If the executing process fails then the launch process is able to log the issue.

This function is enabled by setting `<SeparateProcessing>true</SeparateProcessing>`. As this is mainly relevant for only large input Navisworks and IFC files, the default value is false and this function isn't available when processing PCF files.

Extract Navisworks

The Gateway can extract data from a variety of input sources located in either a local File System or an Amazon Web Services (AWS) in an S3 Bucket. The details of the particular source 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`".

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

- Navisworks (.NWD files)
- AutoCAD 3D (AutoCAD .DWG/.DXF 3D models)
- IFC - (Industry Foundation Classes)
- PCF - (Piping Component Files)

The Gateway can extract .NWD (Navisworks Document) input files.

Note: An NWD file serves as a snapshot of the current state of a model within Navisworks. It contains all model geometry along with Navisworks-specific data, such as review markups. It compresses the CAD data, reducing the file size by up to 80% compared to the original. These files are compact, snapshot-like representations of models, making them ideal for sharing and collaboration in the context of Navisworks projects.

Navisworks files are highly compressed, hence a moderate-sized NWD file can contain an extreme number of elements which require a lot of memory to process and may result in a large ZGL file that is too big to be

imported into AIM. When this is likely, two types of warnings may be logged, therefore when processing large files, it is good practice to open the log file to decide whether the Gateway process is likely to take too long to process due to not enough RAM or should be canceled if the expected ZGL is too big to be imported into AIM.

- When the estimated maximum memory level is too large the warning will be of the form:

  "The input NWD model has 909 million faces which will require about 182 GB of free RAM to process normally. It might still process with less free RAM but will then take much longer to process, or it might fail to process."
- When the estimated output ZGL file size is likely to be too big:

  "The output ZGL model will be about 15 GB which may have issues loading into AIM."

  Review from here: https://dev.azure.com/AVEVA-VSTS/Interoperability/_workitems/edit/4359930

Notes:

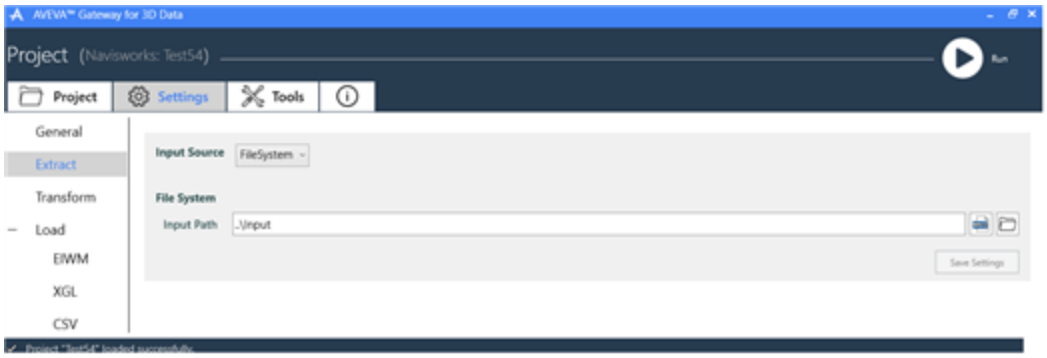
- You need to remove any password protection from NWD files, via Navisworks Manage, before processing in the Gateway.
- Missing XDATA from original AutoCAD models: Navisworks doesn't support the import of XDATA attributes while importing AutoCAD models, therefore they are also absent when importing the Navisworks models into AIM.
- Units displayed in Navisworks may be different: in Navisworks Manage the displayed units can be controlled through the 'Options Editor' tab, but this doesn't change the units stored in the NWD file, hence when extracted by the Gateway they will have the same equivalent value but may be in units different to those displayed in Navisworks.
- Conversion of hexadecimal values to decimal: Navisworks values that are in hexadecimal format (for example, handle values imported from AutoCAD models) are imported as their equivalent decimal values when imported by the Gateway.
- User Data Attributes in Navisworks files: In Navisworks, users can add their own custom attributes to any item in the model scene, as User Data tabs. They are extracted like any other attribute.

  Review from here: https://dev.azure.com/AVEVA-VSTS/Interoperability/_workitems/edit/4079071/

- Navisworks creates and maintains a set of attributes for each element within an NWD model, categorized by tabs. The ones most pertinent to tag identifications are `Name` and `PersistentId` under the `"Item"` attributes tab. To enhance the uniqueness of these attributes (for example, another tab may also have a `Name` attribute) the tab name "`Item_`" is added as a prefix to these two attributes, resulting in attribute names `"Item_Name"` and `"Item_PersistentId"`. In addition, when any other attribute name (for example, `Type`) exists under multiple property type tabs (for example, `Item`, `PDMS`), these also have their tab name added as a prefix to ensure uniqueness (for example, `Item_Type`, `PDMS_Type`).

To specify how to extract the .NWD data:

1. Click **Extract** to display the **Extract Navisworks: <ProjectName>** field.
2. Select the type of **Input Source** from the drop-down box.



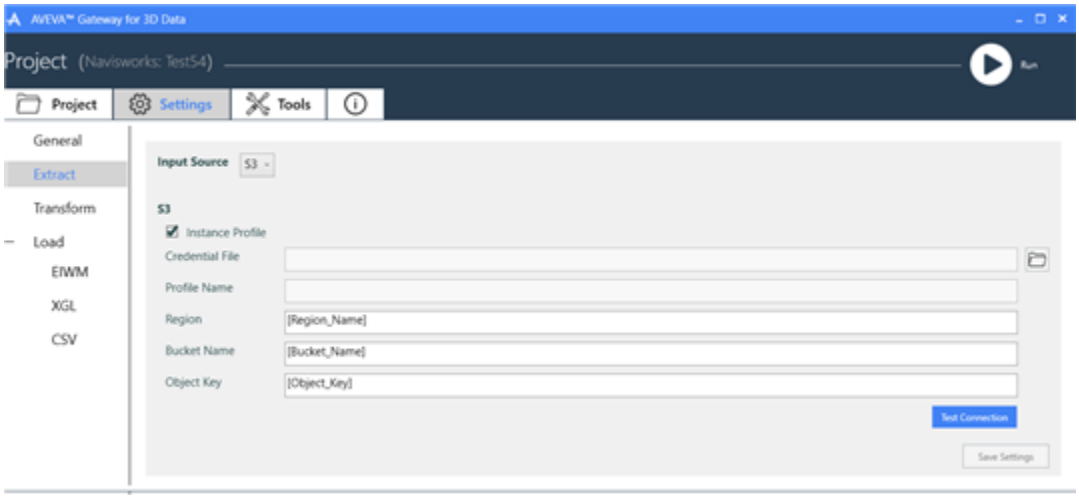
For FileSystem:

3. Define the **Input Path** for the location of the input files (.NWD 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.
4. Select **Save Settings**.

   For S3:

   To configure the extractor for S3 Bucket as input source:

5. Define the elements in S3 Credential Details section. For more information, see Accessing an AWS S3 Bucket.
6. If you want to test connection to S3 Bucket, click **Test Connection**. The result of the test is displayed in the bottom left status bar.
7. Click **Save Settings** to save the settings.



Review from Here: https://dev.azure.com/AVEVA-VSTS/Interoperability/_workitems/edit/3955735/

Conditional Processing

Large NWD models can cause problems during processing and loading results to target systems. Functionality allows you to stop processing NWD files if the number of graphical elements is likely to produce an output ZGL file that is too large to be imported into AIM (currently 9 GB, due to a limitation with 3DVS).

Before reading the entire file, the Navisworks extractor calculates the expected number of graphical elements. This calculation is then used to estimate the size of the output ZGL file and the RAM usage needed to process the model. Note that these predictions are uncertain due to the varying complexity of different models.

The Gateway can be configured to stop processing at this early stage, based on the predicted values, by setting

the 'ProcessingConditions' node options in the Extractor configuration file, these are:

- maxFacesNumber - natural numbers (0, 1, 2, ...)
- maxZglSizeGB - positive floating point numbers
- maxMemoryUsageGB - positive floating point numbers

For example:

```
<ProcessingConditions maxFacesNumber="1000000" />
```

You can set any combination of conditions in one configuration, for example:

```
<ProcessingConditions maxFacesNumber="1000000" maxZglSizeGB="9" />
```

Further processing will stop if at least one of configured values will be predicted to be exceeded. An appropriate error will be reported.

By default, the settings are not added to the configuration. This means that processing will not be interrupted.

Note: The project setting 'timeOut' works independently, meaning that if processing continues it can still be interrupted after a specified time has passed.

Conditional processing can also be set in batch mode. In this case, you can add the appropriate parameters to the command line:

- For maximum number of faces: -maxFacesNumber or -mf.
- For maximum output ZGL file size (in GB): -maxZglSizeGB or -mz.
- For maxium memory usage (in GB): -maxMemoryUsageGB or -mm.

An example:

```
AVEVA.NET.Gateways.Data3D.exe -cp “project.xml” -mf 1000000 -mz 9 -mm 60
```

The configuration is not available in the GUI, so must be added by editing the extractor configuration file.

Note: If you only want to generate a report of the input file structure, set one of the values to '0'. This will interrupt processing, but you will get a log file with useful information about the model. Example setting:

(Extractor configuration file) `<ProcessingConditions maxFacesNumber="0"/>`

(batch mode) `AVEVA.NET.Gateways.Data3D.exe -cp “project.xml” -mf 0`

Extract AutoCAD 3D

The Gateway can extract different data types from .DWG/.DXF input files. 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">. This Extract component can read the data from File System or S3 Bucket sources.

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

To specify how to extract the data:

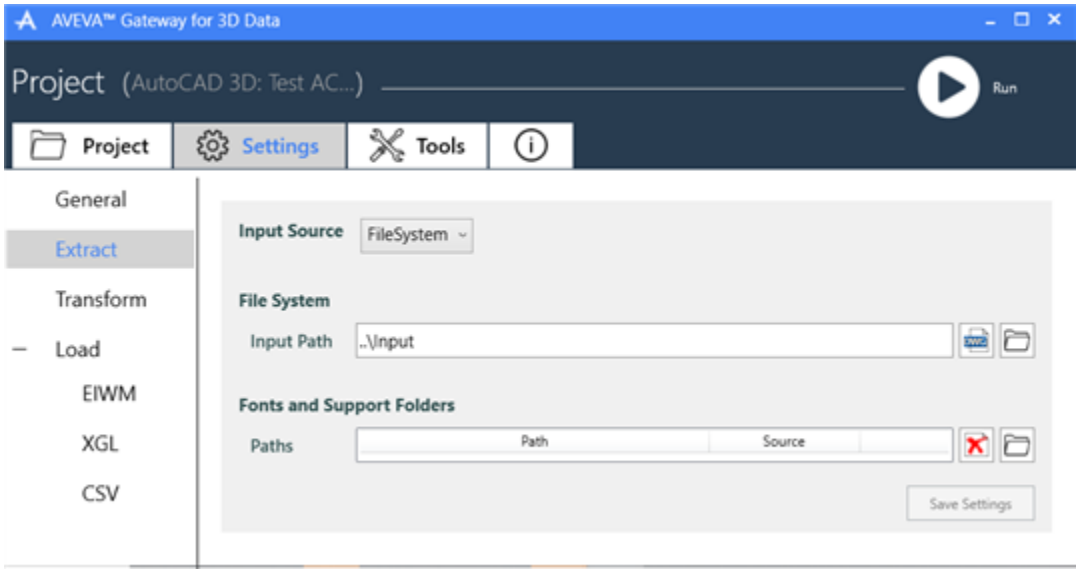
1. Click **Extract** to display the **Extract AutoCAD 3D** field.
2. Select the type of **Input Source** from the drop-down box, either FileSystem or S3.

   For FileSystem:

3. From the **File System Details** section, type the location of the input files (.DWG/.DXF files) from the box. Alternatively, you can use the **Browse DWG/DXF File Path** and **Browse Path** options to enter the input file location.

   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 this support folder, either in the project's configuration or in a system variable.

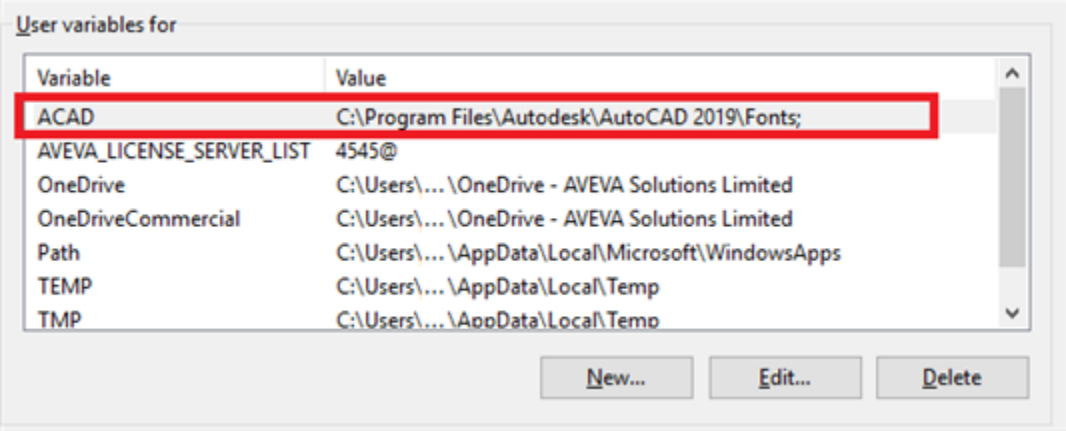
   You can specify the support folder location for the project in the Extract page, as shown below:



Fonts and Support Folders

As it is common to locate all font files in a common system folder, you can 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:



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

You can also 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`.

Review from here: https://dev.azure.com/AVEVA-VSTS/Interoperability/_workitems/edit/4405492/

Note: The input file can sometimes be designed to reference data existing in other files. These references are defined with either relative or absolute paths. In both cases, before starting the processing, ensure that all referenced files are placed in the relevant locations so that the Gateway will include these files in the processing of the input file.

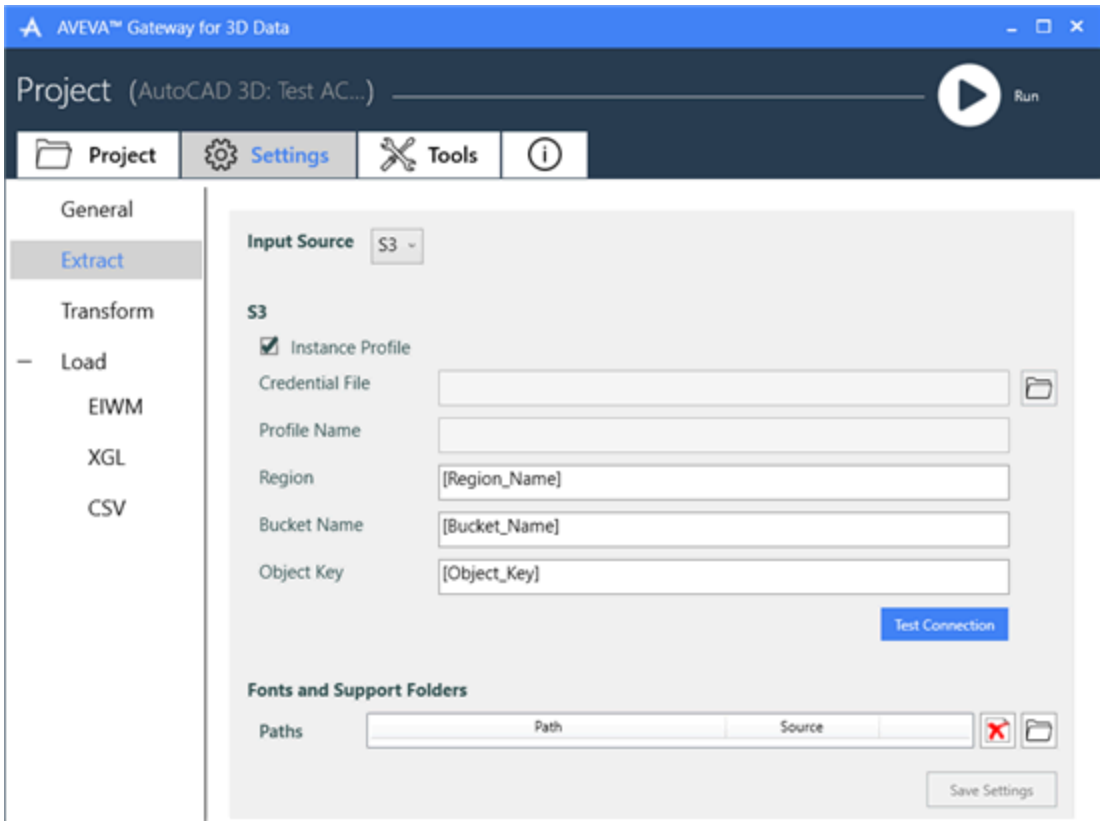
Limitation for S3 and AVEVA Cloud Services (ACS):

When processing an input file stored in either S3 or ACS, the Gateway first downloads a copy of the file to the local file system. Hence if this file contains links to referenced files, they won't be available for processing. To ensure that the input file will be processed successfully, both the file and all referenced files must be stored locally and the Gateway should be configured to use the Filesystem.

For S3:

To configure the extractor for S3 Bucket 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 Bucket, click **Test Connection**. The result of the test is displayed in the bottom left status bar.
3. Click **Save Settings** to save the settings.



Notes:

- You can set tracking of changes on Extraction of data using annotations feature. For information about annotations, see Appendix C: 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.

Extract IFC

Industry Foundation Classes (IFC) is the format designed to exchange Building Information Model (BIM) data between contract entities in the building industry. AIM with access to both the models and their engineering data allows the impacts between building and plant to be assessed directly, both for planning new work and for maintaining existing facilities.

BIM is a process standard to manage complex projects where project, engineering and 3D geometry data are shared between various entities involved in defining project scope, delivery, handover verification and subsequent operations facilities management. BIM is being recognized generally as a more efficient way to manage projects that rely on 3D models. While not every BIM project mandates the use of IFC as the data transfer format, it is the format most widely associated with BIM. It is therefore a common assumption or requirement that products in a BIM project support the IFC format.

IFC (Industry Foundation Classes):

- **Definition**: IFC is an open, non-proprietary file format used for exchanging and sharing data in the field of Building Information Modeling (BIM).
- **Purpose**: It facilitates data exchange between professionals involved in design, construction, management, and maintenance of assets, regardless of the software they use.
- **Format**: IFC files are XML-based and contain information about spatial elements, materials, shapes, and more.
- **Standardization**: IFC is an international standard (ISO 16739) and promotes interoperability across various BIM tools.

The Gateway can extract data either from IFC input files located in a local File System or from files stored in Amazon Web Services (AWS) in an S3 Bucket. This is defined in the extract configuration in the `<input>` element, via the `source` parameter, for example, `<Input source="FileSystem">`.

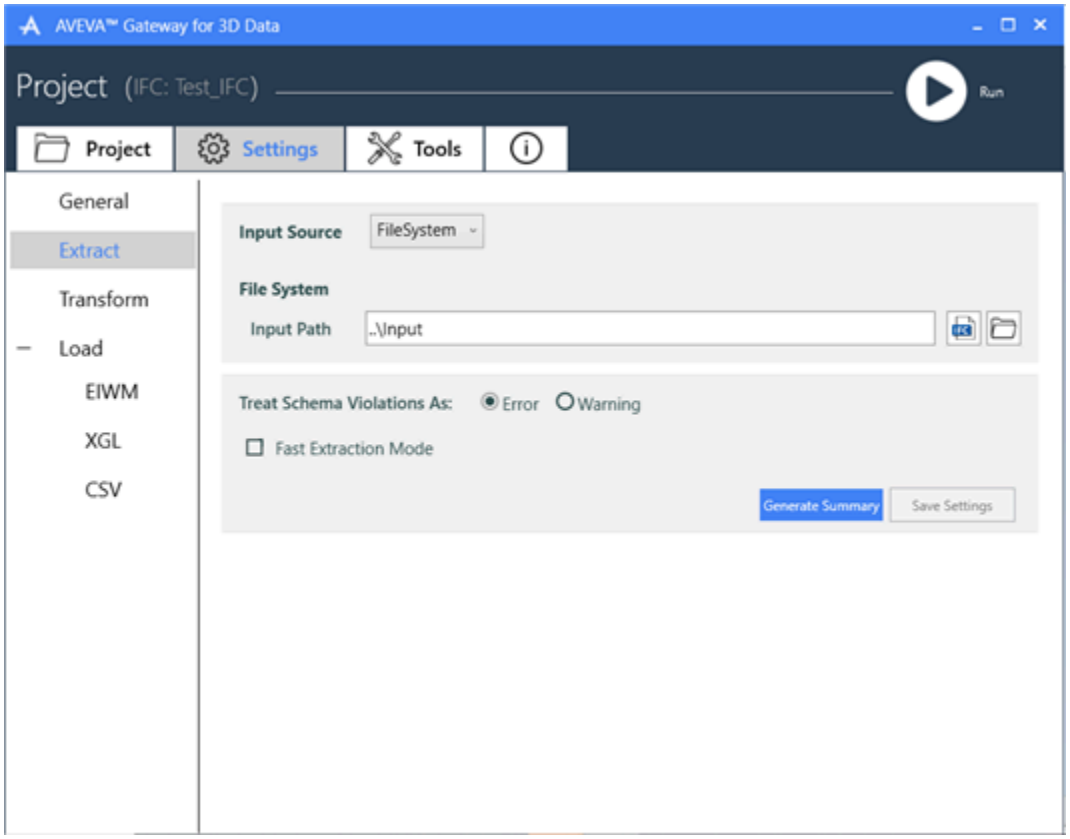
Note: The values of the `source` attribute can be a case-insensitive representation of either "`FileSystem`" or "`S3`".

To specify how to extract the data:

1. Click **Extract** to display the **Extract page**.
2. Select the type of **Input Source** from the drop-down box.

   For File System:
3. From the **File System** section, type the location of the input files (`IFC` files) from the **Input Path** box. Alternatively, you can use the **Browse IFC File** and **Browse Path** options to enter the input file location.

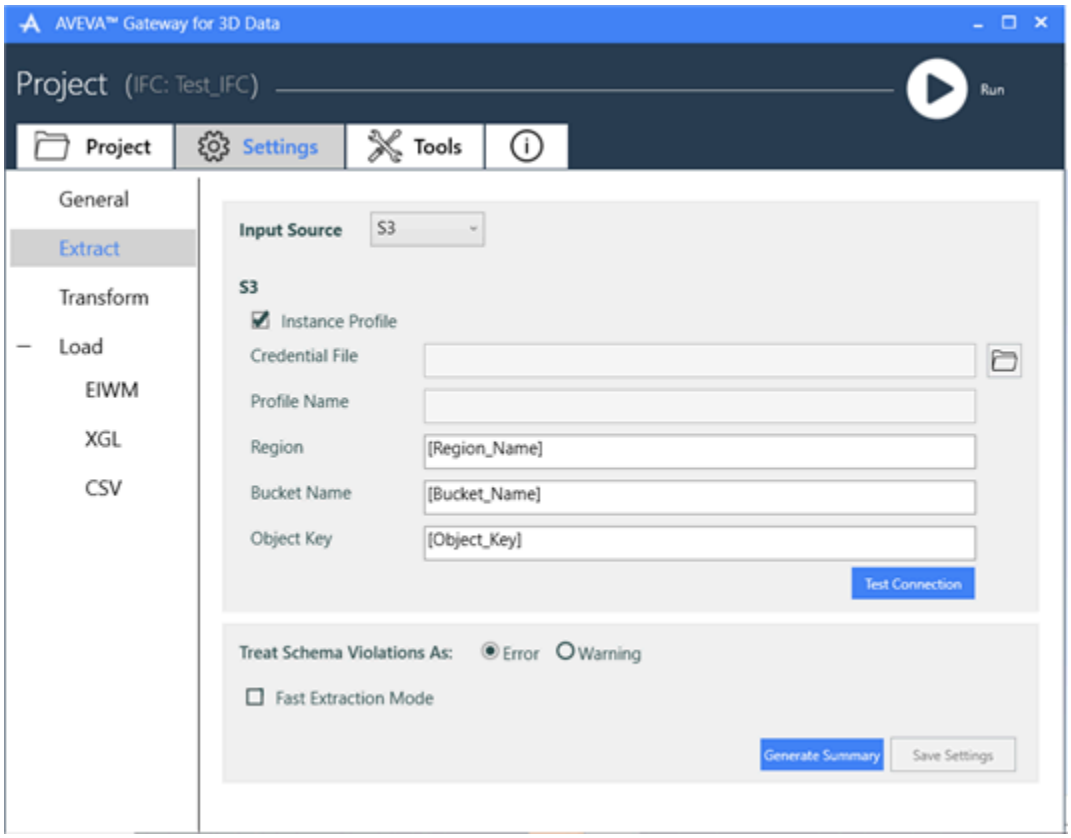
   Note: The **Input Path** value must point to a valid folder location having one or more valid input (.**ifc**) files.



For S3:

To configure the extractor for S3 Bucket as input source:

4. Follow the procedures mentioned in Accessing an AWS S3 Bucket to use the **S3 Credential** Details. If you want to test connection to S3 Bucket, click **Test Connection**. The result of the test is displayed in bottom left status bar.



Other Settings:

5. Select either `Error` or `Warning` to continue the processing of the project, which do not comply with schema definitions in the Treat Schema violations as section. For more information about treating these schema violations, see Requirements for the Thermo Library Generation Tool. You can configure the same using `apply` attribute of the element `<treatSchemaViolationAsWarning apply="1" />` in the Extractor configuration file.

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.

6. Select `FastExtractionMode` option for better performance while extracting objects. See Requirements for the Thermo Library Generation Tool.
7. Click **Save Settings**.
8. Click **Generate Summary** to get a configurable summary of each IFC file.

Note: An **IFC Summary** report provides the statistics and statuses from specific IFC fields. Based on this report, you can decide if any specific file should be imported into AIM. This Summary report is usually in the form of a CSV file containing a section summarizing IFC file names, view definition, creation date, import date and IFC schema version. It also contains counts for each entity types present in the IFC file. This report can be used to troubleshoot any extraction error, if any file fails to be extracted.

Conditional Processing

Large IFC models can cause problems during processing and loading results to target systems. Functionality allows you to stop processing IFC files if the number of graphical elements is likely to produce an output ZGL file that is too large to be imported into AIM (currently 9 GB, due to a limitation with 3DVS).

The IFC extractor reads the entire file and counts the number of graphical elements. This is then used to estimate the size of the output ZGL file and the RAM usage needed to process the model. Note that these predictions are uncertain due to the varying complexity of different models.

The Gateway can be configured to stop processing at this early stage, based on the predicted values, by setting the `'ProcessingConditions'` node options in the Extractor configuration file, these are:

- `maxFacesNumber` - natural numbers (0, 1, 2, ...)
- `maxZglSizeGB` - positive floating point numbers
- `maxMemoryUsageGB` - positive floating point numbers

For example:

```
<ProcessingConditions maxFacesNumber="1000000" />
```

You can set any combination of conditions in one configuration, for example:

```
<ProcessingConditions maxFacesNumber="1000000" maxZglSizeGB="9" />
```

Further processing will stop if at least one of configured values will be predicted to be exceeded. An appropriate error will be reported.

By default, the settings are not added to the configuration. This means that processing will not be interrupted.

Note: The project setting `'timeOut'` works independently, meaning that if processing continues it can still be interrupted after a specified time has passed.

Conditional processing can also be set in batch mode. In this case, you can add the appropriate parameters to the command line:

- For maximum number of faces: `-maxFacesNumber` or `-mf`.
- For maximum output ZGL file size (in GB): `-maxZglSizeGB` or `-mz`.
- For maximum memory usage (in GB): `-maxMemoryUsageGB` or `-mm`.

An example:

```
AVEVA.NET.Gateways.Data3D.exe -cp “project.xml” -mf 1000000 -mz 9 -mm 60
```

The configuration is not available in the GUI, so must be added by editing the extractor configuration file.

Note: If you only want to generate a report of the input file structure, set one of the values to '0'. This will interrupt processing, but you will get a log file with useful information about the model. Example setting:

(Extractor configuration file) `<ProcessingConditions maxFacesNumber="0"/>`

(batch mode) `AVEVA.NET.Gateways.Data3D.exe -cp “project.xml” -mf 0`

Treat Schema violations as Error or Warning

The Gateway uses the relevant IFC schema (2x3 or 4) to generate the property names for the property values of each entity in an IFC file. Normally an Error is logged if the number of properties in the entity is different to the schema definition. An option is available on the **Extract** tab to treat such schema violations as warnings, so that the issue can be logged but subsequent entities will be processed.

You must check the log file for any such warnings and decide whether to correct the IFC file and re-process or allow the schema violation:

- If **Error** is selected (default), the Gateway stops processing any file which violates the schema definition and logs the Error.
- If **Warning** is selected, the Gateway proceeds with processing all entities in the file, and logs the Warning for any schema violations. For example, if N is the number of attributes (IFC properties) for an entity in the file and M is the number of attributes defined in the IFC schema, then the following two scenarios occur:
  - If N > M, the Gateway reads the first M attributes and logs a warning.
  - If N < M, the Gateway assumes that the first N attributes are correct and the missing M-N attributes have a value of $ and logs a warning.

    If an attribute has a relationship with another (target) entity but the class of the target entity violates the schema definition, the Gateway processes the target entity based on the class defined in the IFC file and logs a warning.

    If the above assumptions result in a processing error, the Gateway ignores that entity and its dependent entities, logs a warning and processes the next entity.

    Note: The optional `'annotations'` feature allows you to store additional data about how each object, attribute or association has been changed due to a transformation step. These tracking attributes store data about creation, modification and deletion operations and can be inspected via CSV reporting. This can be achieved by adding XML `<annotations>` node to the module configuration. Available `annotations level` values are `'None'` and `'Basic'`. For example:

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

For more information, see Appendix B: CSV Reporting.

Note: The Gateway can identify some attributes as key attributes, as these attributes will be exported into EIWM file even without mapping in configuration. For example, ObjectID, ObjectName, Revision and so on. For more information about these key reserved attributes, see Appendix E: Reserved Attributes Used in the Gateway.

FastExtractionMode

Normally relationships between IFC objects are determined by querying the source object for all of its inverse and direct relationships, which is time-consuming. If FastExtractionMode is enabled, then inverse relationships are created after all objects have been extracted via a separate process, which is a lot faster. It does however result in different association types as they are then derived from the inverse attribute name, rather than the inverse relationship type name from the source object's schema definition.

For example, if part of the input IFC file is defined as:

```
#380= IFCDOOR('3$pEhtFpv31QFndkpGikoC',#41,'M_Single-Flush:0915 x 2134mm:197506',$,'0915 x 2134mm',#698,#374,'197506',2134.,915.);
#385= IFCPROPERTYSINGLEVALUE('Reference',$,IFCIDENTIFIER('0915 x 2134mm'),$);
#386= IFCPROPERTYSET('3$pEhtFpv31QFnbHpGikoC',#41,'Pset_DoorCommon',$,(#247,#385));
#388= IFCRELDEFINESBYPROPERTIES('10D4T6dArDZePbheZhDXiI',#41,$,$,(#380),#386);
```

Then the normal method for creating relationships uses the IFC schema definition of `IFCDOOR` which contains an inverse attribute called `'IsDefinedBy'` and its type is `'IfcRelDefines'`.

Note: `IfcRelDefines` is a base type having subtypes like `IfcRelDefinesByProperties`,

IfcRelDefinesByType. If FastExtractionMode is enabled, then the association type is derived from the IFC file as "IfcRelDefinesByProperties", that is, it takes whatever the inverse relationship entity is defined as in the input file.

Without any association transformation, the EIWM will be:

```
<Object>
<ID>3$pEhtFpv31QFndkpGikoC</ID>
<Context>
<ID>IPE</ID>
</Context>
<ClassID>IFCDOOR</ClassID>
<Association type="IFCRELDEFINESBYPROPERTIES">
<Object>
<ID>3$pEhtFpv31QFnbHpGikoC</ID>
<Context>
<ID>IPE</ID>
</Context>
<ClassID>IFCPROPERTYSET</ClassID>
</Object>
</Association>
</Object>
```

If you disable the FastExtractionMode, then the same IFCDOOR object will have an association type 'IsDefinedBy' in EIWM.

EIWM:

```
<Object>
<ID>3$pEhtFpv31QFndkpGikoC</ID>
<Context>
<ID>IPE</ID>
</Context>
<ClassID>IFCDOOR</ClassID>
<Association type="IsDefinedBy">
<Object>
<ID>3$pEhtFpv31QFnbHpGikoC</ID>
<Context>
<ID>IPE</ID>
</Context>
<ClassID>IFCPROPERTYSET</ClassID>
</Object>
</Association>
</Object>
```

Limitations:

The following related types in certain inverse relationships cannot be processed:

- **IfcRelConnectsPathElements**:
  - RelatingPriorities: Priorities for connection. It refers to the layers of the RelatingObject.
  - RelatedPriorities: Priorities for connection. It refers to the layers of the RelatedObject.
  - RelatedConnectionType: Indication of the connection type in relation to the path of the RelatingObject.
  - RelatingConnectionType: Indication of the connection type in relation to the path of the RelatingObject.
- **IfcRelConnectsWithRealizingElements:**

  - RealizingElements: Defines the elements that realize a connection relationship.
- **IfcRelSpaceBoundary**:
  - PhysicalOrVirtualBoundary: Defines, whether the Space Boundary is physical (Physical) or virtual (Virtual).
  - InternalOrExternalBoundary: Defines, whether the Space Boundary is internal (Internal), or external, that is, adjacent to open space (that can be a partially enclosed space, such as terrace (External).

Extract PCF

The Gateway can extract data from XGL, ZGL and PCF input file sources. Click Extract to populate the two subsections: Input Source XGL/ZGL and Input Source PCF.

Notes:

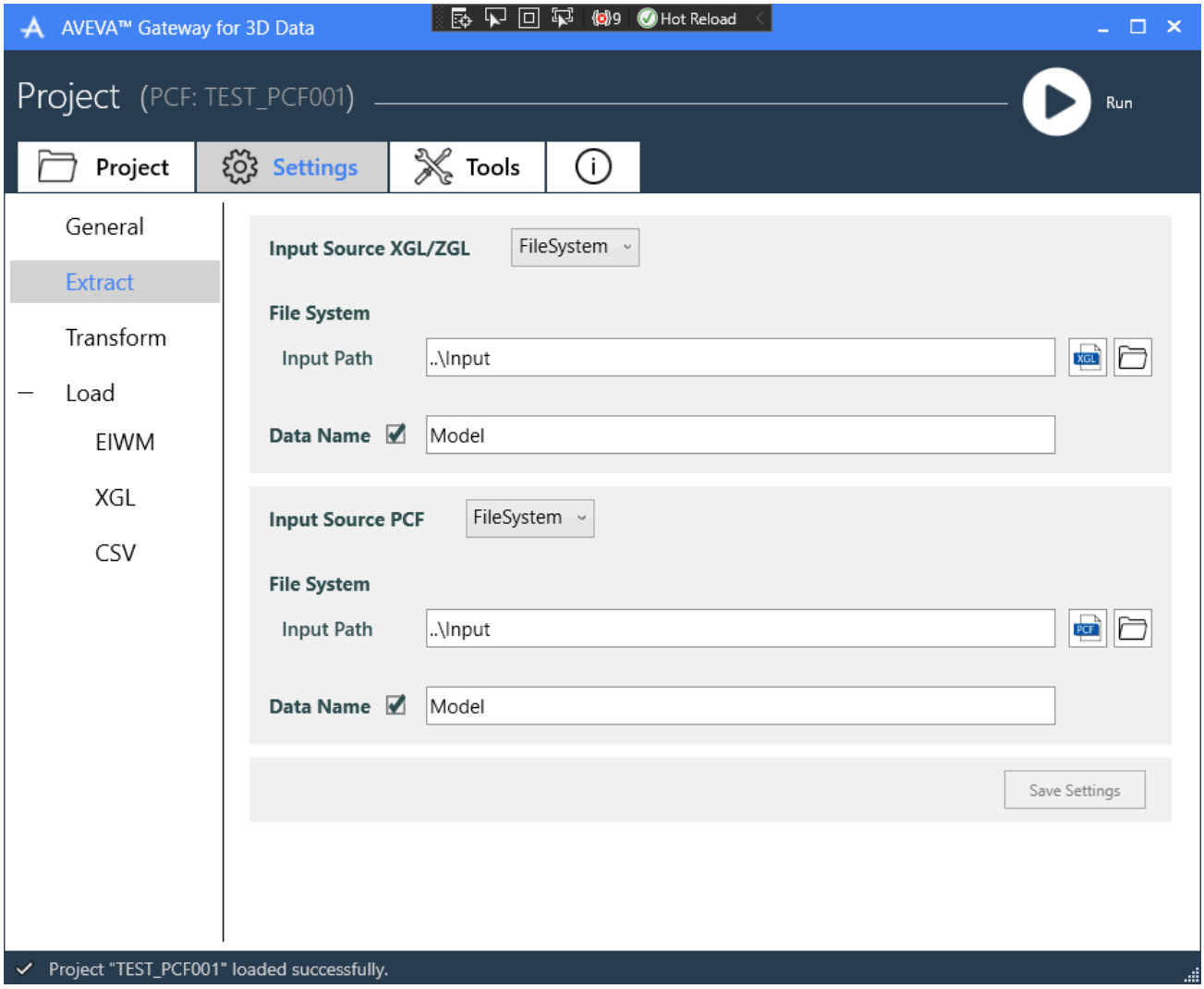
- **XGL (XML-based Graphics Language)** is an XML-based file format used to represent 2D vector graphics. XGL files store graphical elements like shapes, paths, colors, and transformations in a structured XML format, making them easily readable and editable.
- **ZGL (Z-Buffer Graphics Library)**: ZGL files store 3D models and their associated data. This includes geometry, texture information, material settings, and other attributes. XGL file means graphics with XML format or ZGL file means compressed XGL format.
- **PCF (Piping Component File)** are used specifically in the context of piping and process plant design. They contain information related to piping components, such as valves, fittings, and pipes. PCF files serve as an intermediate format for exchanging piping data between different software applications. PCF files are not XML-based, but have a simpler structure.

To Extract XGL and ZGL Source file

To specify how to extract the data from **XGL** and **ZGL** input files:

1. Select **Input Source XGL/ZGL** section under **Extract** to open the **XGL** setting window.
2. Select the type of **Input Source** from the drop-down box. You can select **FileSystem**, **S3** or **None** to extract data from the input sources.

   Note: Selecting **None** means the extractor does not extract any data. As a result, there will be no complete data model.



When Input Source: FileSystem

To specify how to extract data from the **File System** as input source:

3. From the **File System** section, type the location of the input XGL files in the **Input Path** box. Alternatively, you can use the **Browse 3D Model File** icon  to enter **XGL/ZGL** input files. You can use the Browse Path  option to enter the input file location.

Note: The details of the particular input source 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">. This Extract component can read the data from File System or S3 sources.

4. Select the **Data Name** field to set the user name for the processed data. For more information about **Data Name** field, see the Data Name section.

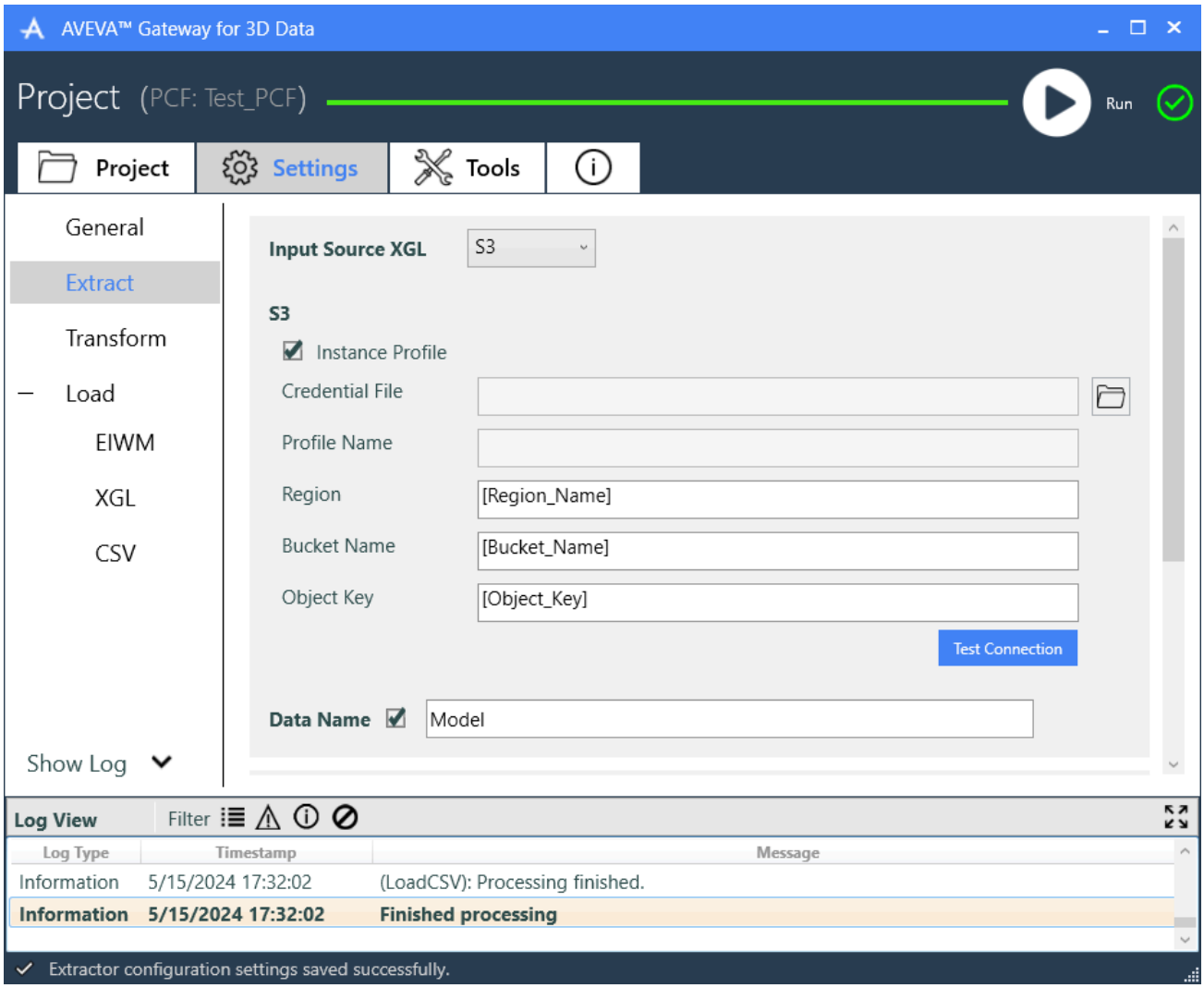
When Input Source: S3

To specify how to extract data from the S3 as input source:

5. Select the type of Input Source from the drop-down box.
6. Define the respective fields in S3 section. For more information about these fields, see Accessing an AWS S3

Bucket.

7. If you want to test connection to S3 Bucket, click Test Connection. The result of the test is displayed in the bottom left status bar.



8. Select the **Data Name** field to set user name for the processed data. For more information about Data Name field, see the Data Name section.
9. Click **Save Settings** to save the settings.

Notes:

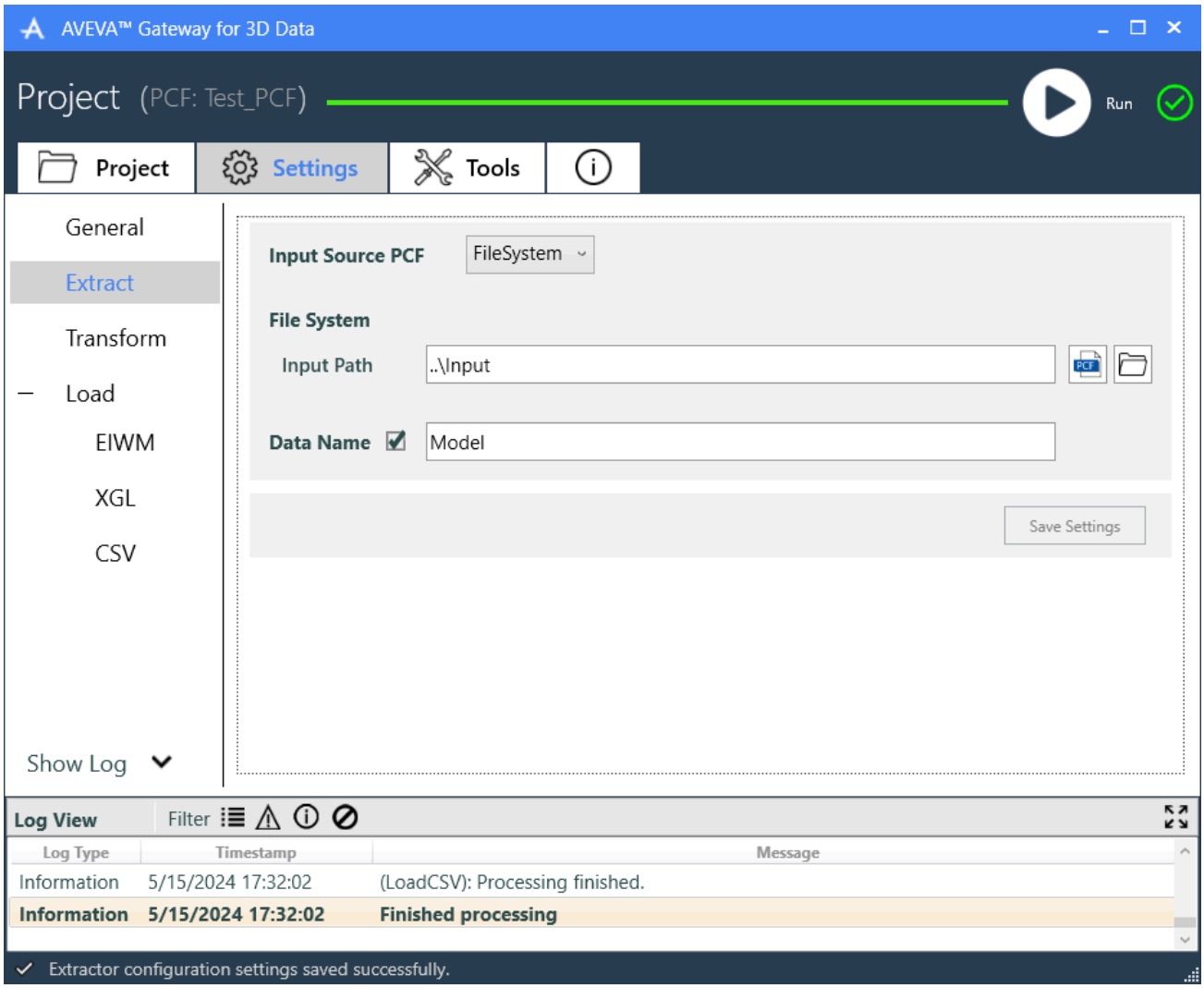
- You can set tracking of changes on Extraction of data using annotations feature. For information about annotations, see Appendix C: Tracking Changes in Data.
- The Gateway does not extract any OVERLAID XREF files in models.
- The Gateway can identify some key attributes, for example, ObjectID, ObjectName, Revision and so on will be exported into EIWM file even without mapping in configuration.

To Extract PCF Source file

To specify how to extract the data from PCF Input files:

1. Select Input Source PCF section under Extract to open the XGL setting window.

2. Select the type of Input Source from the drop-down box. You can select FileSystem, S3 or None to extract data from the input sources.



When Input Source: FileSystem

To specify how to extract data from the File System as input source:

3. From the File System section, type the location of the input PCF files in the Input Path box. Alternatively, you can use the Browse 3D Model File  icon to enter PCF input files. You can use the Browse Path  option to enter the input file location.

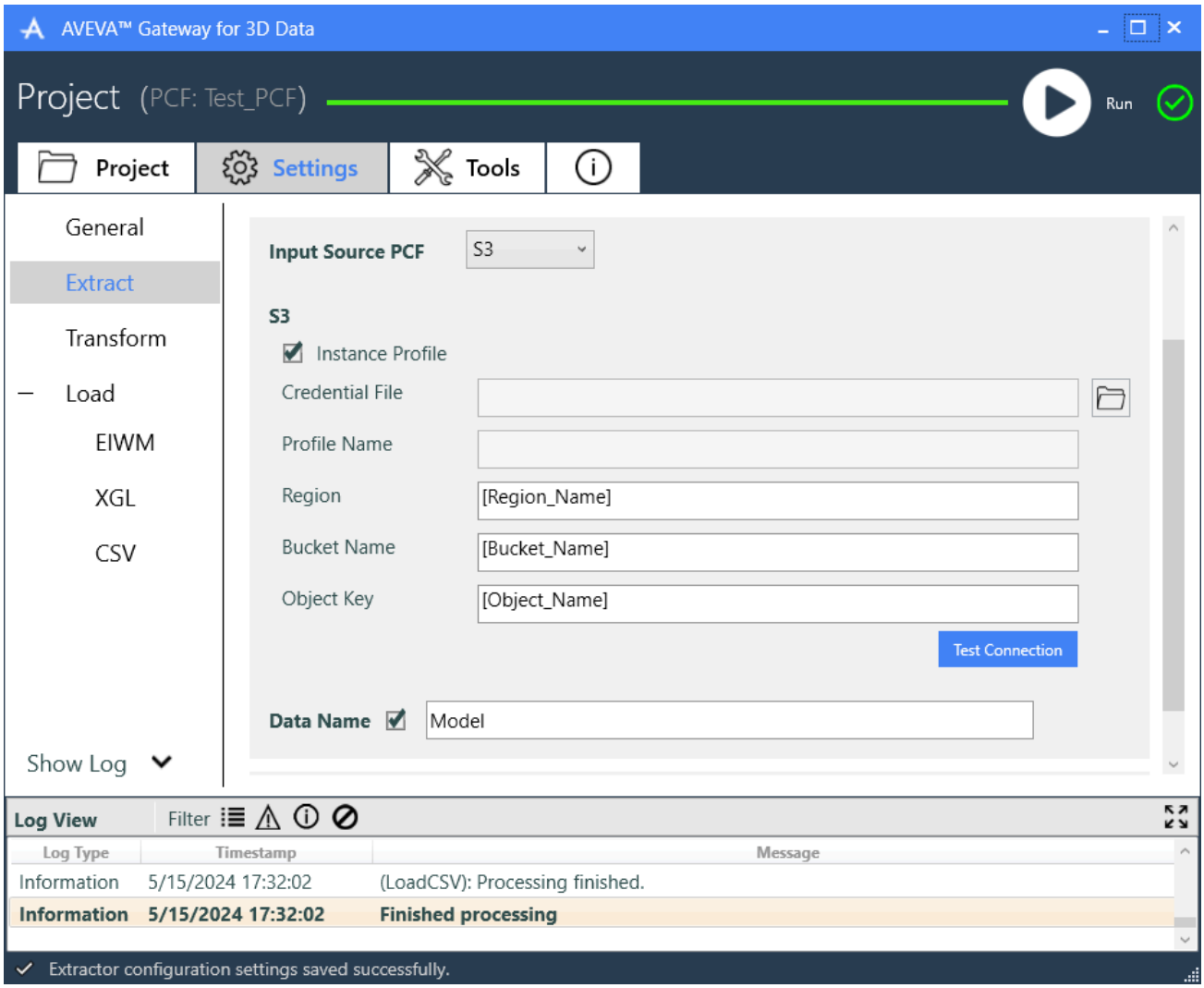
Note: The details of the particular input source to be read by the project are defined in the extract configuration in the <input> element, via the source parameter, for example, `<Input Source="FileSystem">`. This Extract component can read the data from File System or S3 Bucket sources.

4. Select the **Data Name** field to set user name for the processed data. For more information about **Data Name** field, see **Data Name** section.
5. Click Save Settings to save the settings.

For S3 Input Source:

6. Select the type of **Input Source** from the drop-down box.

7. Define the respective fields under S3 section. For more information about these fields, see Accessing an AWS S3 Bucket.
8. If you want to test connection to S3 Bucket, click Test Connection. The result of the test is displayed in the bottom left status bar.



9. Select the **Data Name** field to set user name for the processed data. For more information about Data Name field, see Data Name section.
10. Click Save Settings to save the settings.

Data Name

To define the base name for the LOG, XGL, EIWM and CSV Report files, a DataName parameter is used to define the base name, independent of the input file names. When the Input Path is a folder only, the data name is required. When the full file path is defined, data name is optional. If the Data Name is not defined, then the Log file name and CSV Reports name are derived from the PCF Extractor name. The following table lists the base names for all input file combinations:

PCF Input	XGL Input	EIWM	XGL Output	Log and CSV File	Comment
PCF-ABC.pcf	XGL-XYZ.xgl	PCF-	XGL-XYZ.xgl /	Takes name	

		ABC_avngate.xml / <PCF Data Name>_avngate.xml	<XGL Data Name>.xgl	from EIWM	
PCF-ABC.pcf	Folder path		<XGL Data Name>.xgl		Extractor merges XGL files
PCF-ABC.pcf	-		-		
Folder path	XGL-XYZ.xgl	<PCF Data Name>_avngate.xml	XGL-XYZ.xgl / <XGL Data Name>.xgl		Extractor merges PCF files
Folder path	Folder path		<XGL Data Name>.xgl		Extractor merges PCF and XGL files
Folder path	-		-		Extractor merges PCF files
-	XGL-XYZ.xgl	Takes name from XGL output	XGL-XYZ.xgl / <XGL Data Name>.xgl		
-	Folder path		<XGL Data Name>.xgl		Extractor merges XGL files

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:

- `Base Mapping`: Modifies the engineering data of a model's objects present in the extracted data. For more information, see Base Mapping.
- `TransformGeometry`: Transforms the coordinates of the geometry of the model via scaling, offsets and rotation functions, either defined explicitly in the configuration or derived from three matching points in the source and target models.
- `PresentationMapping`: Adjusts the output graphics properties, such as materials and colour. It can also be used to segment the input model to be included in the XGL rendition.
- `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:

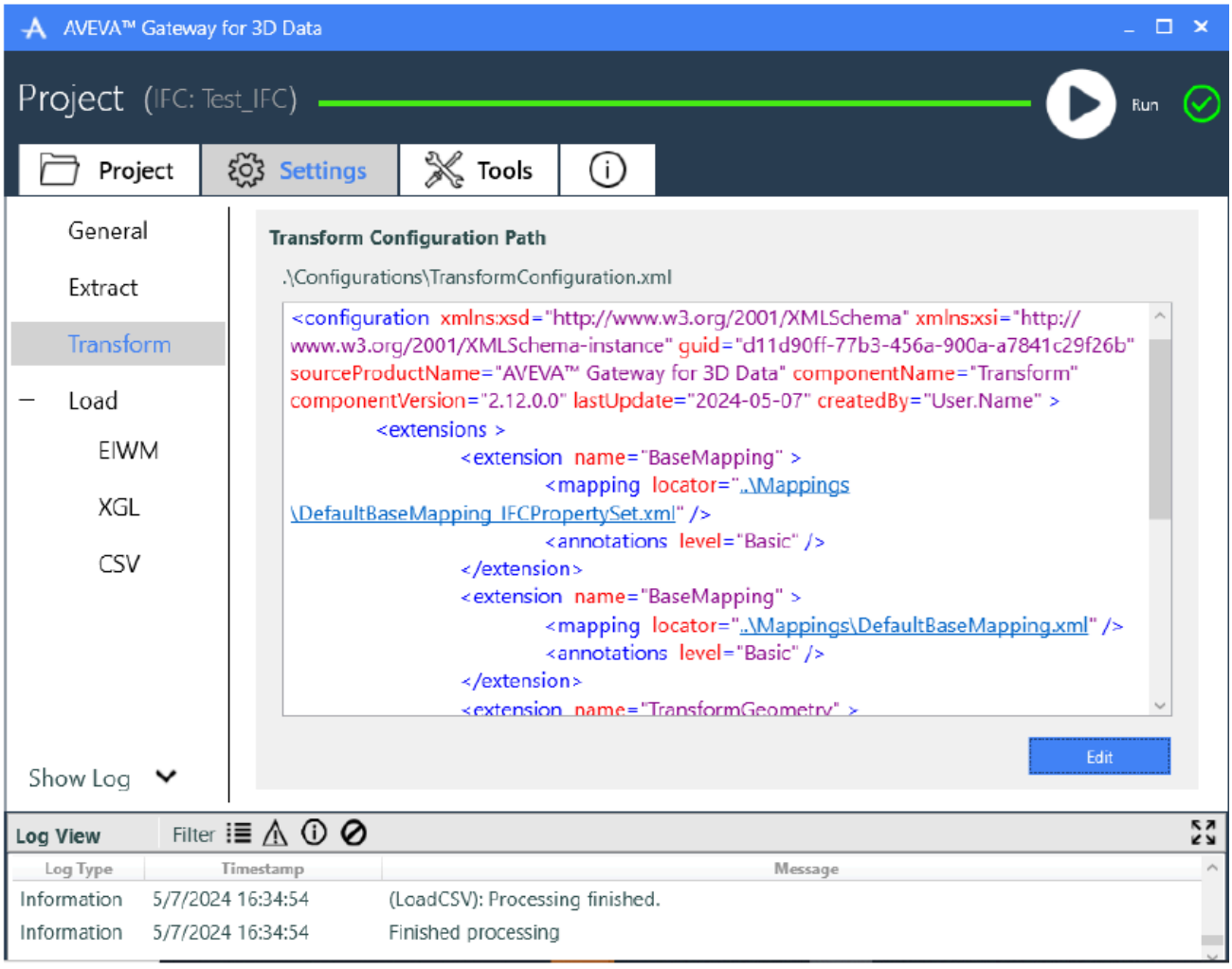
- As a number of mapping configuration files can be associated to a project configuration, order of execution of these files is determined as follows:

  - 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 its mapping 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** tab in the **Project Settings** page to open the **Transform** details page.



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

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

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

Notes:

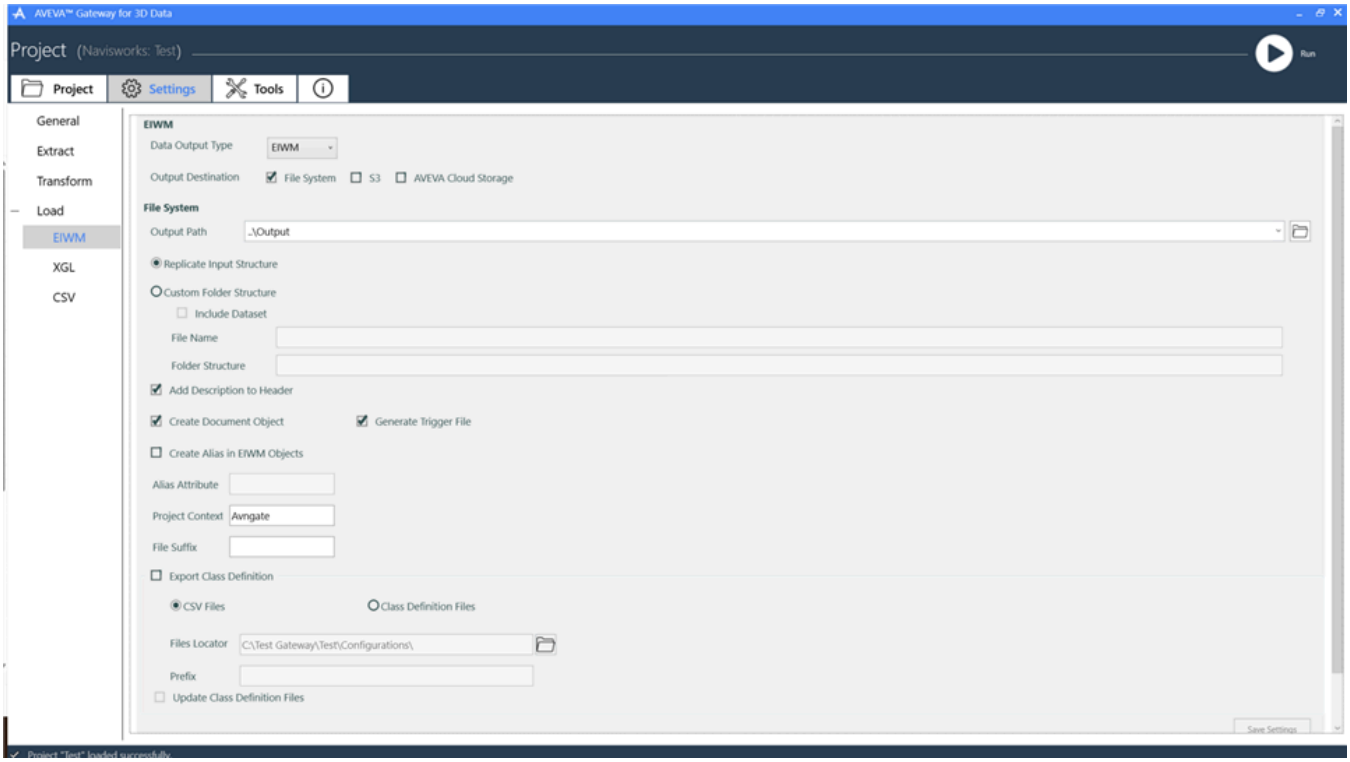
- Any update in the configuration document must be saved before running the process.
- You can set tracking of changes provided by each Transform extension using annotations feature as shown in the above XML configuration. For more information about annotations, see Appendix C: Tracking Changes in Data.
- A CSV Report that is defined in the Transformation configuration file can have its output path defined by the `outputFolder` parameter. This parameter is optional, and when it is not set, the CSV file is written to the EIWM location by default. 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 F: Handling of System Attributes.

Load Settings

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

Load EIWM

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



Review from Data Output Type

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

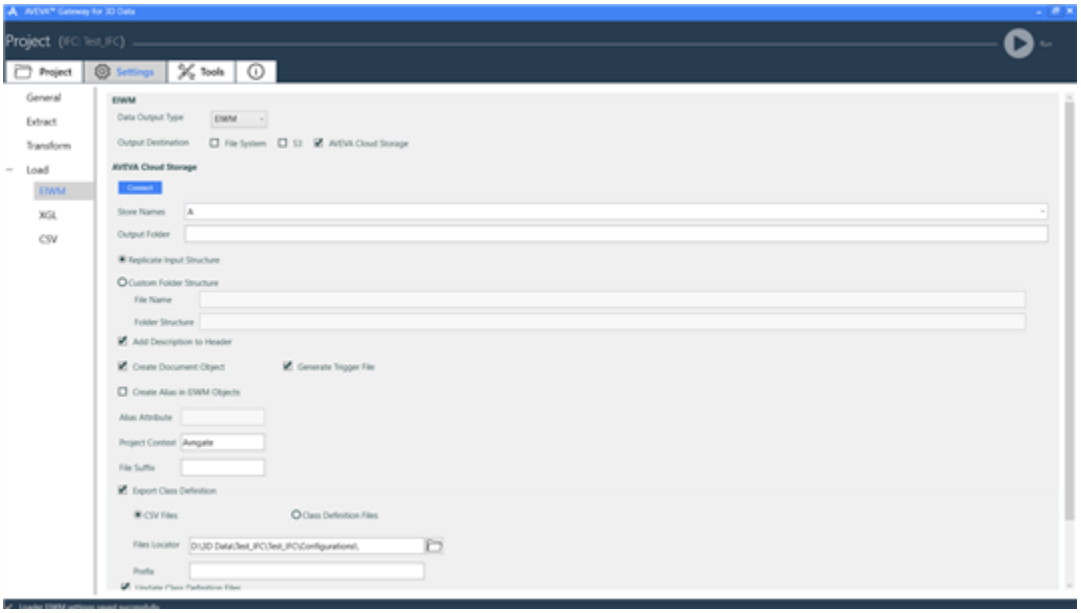
- **Data Output Type**: Normally set to EIWM where the output Engineering data will be written to an EIWM XML file. Selecting **NONE** means that no Engineering data is written, for example, when only a CSV report is needed. You can configure this in the Load EIWM Configuration file using the `dataType` element, for example, `<dataType value="EIWM" />`.
- **Output Destination**: Select whether the output will be written to a File System or an S3 Bucket 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">`.

  Note: The `Destination` values can be a case-insensitive representation of either "`FileSystem`" or "`S3`", or "`ACS`".

  - **File System:** Under **Output Path** option, type the output file path or click the **Browse for Folder** icon to search for the output folder path.

    Note: The Output EIWM Location is reported in the OutputData section of the Categorized Logs JSON report, see General Settings.

  - **S3**: For more information while using S3 as the output destination, see Accessing an AWS S3 Bucket.
  - **AVEVA Cloud Storage**: For more information while using ACS as the output destination, see Accessing 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 and the **IncludeDataset** checkbox. To write all output files to the **Output** Folder, leave the **File Name** and **Folder Structure** fields empty.

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

Custom File Name Structure

File Name is configurable and the expected entries in a **File Name** 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>
```

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

- **Produce Revisions in Output File Names:**

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

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

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

```
<OutputDestination Destination="FileSystem"
FolderStructure="CustomFolderStructureForOutput">
```

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

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

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

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

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

Custom Folder Structure

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

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

Dataset Objects

If an object has datasets (generated from 'has dataset' or 'is a dataset of') and the Include Dataset option is checked, then EIWM files containing Dataset objects are created within the same directory as their related objects. The suggested combination while using this check box is File Name as "eiwm:id" and Folder Structure as "eiwm:classid". This will result in one object or dataset per EIWM file, with each set of files located in their respective ClassID folders.

Notes:

- When the Include Dataset option is checked then no output folder will be created with a ClassId set of Dataset Objects.
- When Filename is empty and and FileStructure is "eiwm:classid", the dataset objects are merged with the related objects in the one EIWM file per class.

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: [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

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

  Note: The `apply` value must be an `xsd` schema supported Boolean value. The valid values for `xsd:boolean` are `true`, `false`, `0` and `1`. Attribute values that are capitalized (for example, `TRUE`) or abbreviated (for example, `T`) are not valid.
- **Create Document Object:** A Document Object is the metadata object present in the EIWM that describes the input data source. It holds the information about the data source for extraction like filename, date and time of extraction and so on. Default value of `createDocumentObject` option is `true`.

  The configuration entry looks as follow.

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

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

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

  Note: You can modify the same setting from both configuration and the EIWM Loader GUI page.
- **Generate Trigger File**: After output file generation, to trigger the Import Controller, a **trigger.start** file is

created in the staging area. It passes through three stages: start, process and finish. When the file reaches the finish stage, the output files are imported into AIM.

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

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

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

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

Notes:

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

  Examples:

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

  Load Configuration file:

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

  Mapping configuration is defined as:

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

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

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

```
<Object>
<ID>3$pEhtFpv31QFndkpGikoC</ID>
<Context>
<ID>Avngate</ID>
```

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

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

  Load configuration file:

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

  Mapping configuration is defined as:

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

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

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

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

  Load configuration file:

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

Mapping configuration is defined as:

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

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

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

- **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.
- **Export Class Definition**: Select this option to generate AIM bootstrap files that can be imported into AIM by the ImportController to generate or update the AIM schema definition for: classes, characteristics, properties, associations and permissible associations. You can configure this by selecting which files to generate, either Class Definition files (aka bootstrap files) or CSV files that contain the same values that might be used in a post process. The default structure of **Class Definition Configuration File is** shown as follows:

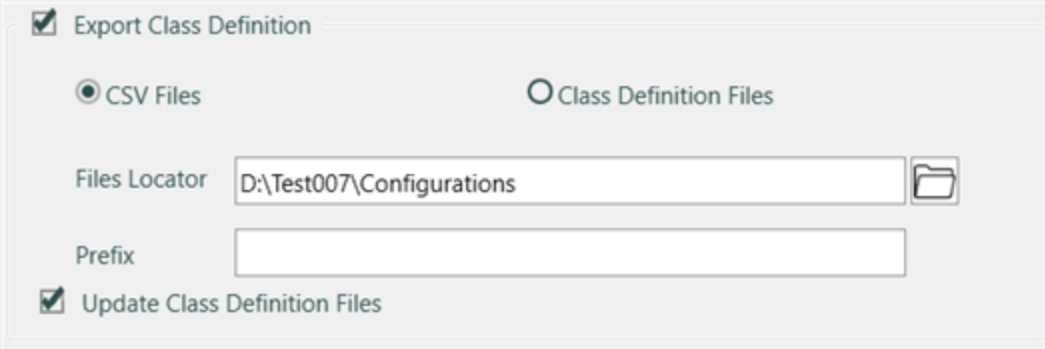
ClassDefinitionConfiguration.xml - Notepad

File Edit Format View Help

```
<?xml version="1.0" encoding="UTF-8"?>
<ClassDefinitionConfiguration>
  <!--[please provide existing class definition file path for Class to create or update]-->
  <Class path="[path]">
    <ClassID From="ClassID" />
    <ParentClassID Value= "[UserDefined ParentClassID]" />
    <ClassName From="ClassID" />
    <PluralName From="ClassID" Append ="" />
    <ClassIcon Value=""/>
    <Visible Value=""/>
    <Hidden Value=""/>
  </Class>
  <!--[please provide existing class definition file path for CharacteristicClass to create or update]-->
  <CharacteristicClass path="[path]">
    <ClassID From="Name" />
    <ParentClassID Value="[UserDefined ParentClassID]" />
    <ClassName From="Name" />
    <DataType Value=""/>
    <AllowInvalidDataType Value=""/>
  </CharacteristicClass>
  <!--[please provide class definition file path for PropertyClass to create or update]-->
  <PropertyClass path="[path]">
    <ClassID From="Name" />
    <ParentClassID Value="[UserDefined ParentClassID]" />
    <ClassName From="Name" />
    <Definition Value="" />
    <MeasureClassID From ="Unit" />
    <AllowInvalidDataType Value=""/>
  </PropertyClass>
  <!--[please provide class definition file path for AssociationClass to create or update]-->
  <AssociationClass path="[path]">
    <ID From="type" />
    <Name From="type" />
    <SourceRole From="type" />
    <TargetRole From="type" />
  </AssociationClass>
  <!--[please provide class definition file path for PermissibleAssociation Class to create or update]-->
  <PermissibleAssociation path="[path]">
    <ClassID From="ClassID" />
    <AssociationType From="type" />
    <TargetClassID From="AssociatedObject[ClassID]" />
  </PermissibleAssociation>
</ClassDefinitionConfiguration>
```

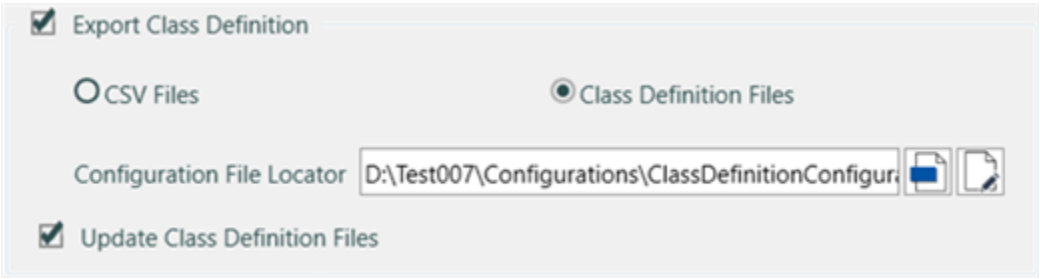
Warning: Importing bootstrap files into AIM should always be done with caution as an incorrectly configured bootstrap file can corrupt the AIM schema and hence your access to AIM data. You should be thoroughly familiar with bootstrap functionality as described in AIM's documentation before using this Gateway feature. If in any doubt seek guidance and advice from the AVEVA helpdesk.

- **CSV Files:** When you select the CSV Files, two configurable options appear:
  - **Files Locator**: Selects the folder where the output bootstrap CSV files will be generated. By default, this path is set to Configurations folder of the select project.
  - **Prefix**: Sets the prefix to be used in naming the output CSV files.



When you select Class Definition Files, one configurable option appears:

- **Configuration File Locator:** Selects the location for the Class Definition Configuration xml file. Select this option to generate AIM bootstrap files. The Class Definition Configuration file can be selected by using the Browse path button or edited by selecting the Edit button. By default, the ClassDefinitionConfiguration.xml file is present in the Configurations folder of the selected project.



The default structure of Class Definition Configuration File is shown as follows:

```
<ClassDefinitionConfiguration xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" sourceProductName="AVEVA™
Gateway for 3D Data" componentName="ClassDefinitionConfiguration"
componentVersion="2.12.0.0" >
<!--[please provide existing class definition file path for Class to create or
update]-->
<Class path="[path]">
<ClassID From="ClassID" />
<ParentClassID Value="[UserDefined ParentClassID]" />
<ClassName From="ClassID" />
<PluralName From="ClassID" Append="" />
<ClassIcon Value="" />
<Visible Value="" />
<Hidden Value="" />
</Class>
<!--[please provide existing class definition file path for CharacteristicClass to
create or update]-->
<CharacteristicClass path="[path]">
<ClassID From="Name" />
<ParentClassID Value="[UserDefined ParentClassID]" />
<ClassName From="Name" />
<DataType Value="" />
<AllowInvalidDataType Value="" />
</CharacteristicClass>
<!--[please provide class definition file path for PropertyClass to create or
```

```
update]-->
<PropertyClass path="[path]">
<ClassID From="Name" />
<ParentClassID Value="[UserDefined ParentClassID]" />
<ClassName From="Name" />
<Definition Value="" />
<MeasureClassID From="Unit" />
<AllowInvalidDataType Value="" />
</PropertyClass>
<!--[please provide class definition file path for AssociationClass to create or
update]-->
<AssociationClass path="[path]">
<ID From="type" />
<Name From="type" />
<SourceRole From="type" />
<TargetRole From="type" />
</AssociationClass>
<!--[please provide class definition file path for PermissibleAssociation Class to
create or update]-->
<PermissibleAssociation path="[path]">
<ClassID From="ClassID" />
<AssociationType From="type" />
<TargetClassID From="AssociatedObject[ClassID]" />
</PermissibleAssociation>
</ClassDefinitionConfiguration>
```

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

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

- **Load EIWM configuration file format**: 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.Gateways.3DData" componentName="LoadEIWM"
componentVersion="2.10.0.0" >
<OutputDestination Destination="FileSystem">
<S3>
<Authentication instance="true">
<CredentialFile path="default" profileName="default" />
</Authentication>
<Region>Mumbai</Region>
<BucketName>S3Bucket</BucketName>
<OutputDirectory>Output</OutputDirectory>
</S3>
<FileSystem>
<OutputPath>D:\Tabular\Test1\Output</OutputPath>
</FileSystem>
</OutputDestination>
<dataType value="EIWM" />
<keepInputFolderStructureForOutput apply="true" />
<generateTriggerStart apply="true" />
<addDescriptionToHeader apply="true" />
<projectContext value="Avngate" override="false" />
<createAlias apply="true" attribute="GlobalID" />
<file maxObjectCount="0" suffix="_part_" suffixNumberFormat="DDD" />
```

```
<annotations level="Basic" />
</configuration>
```

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:

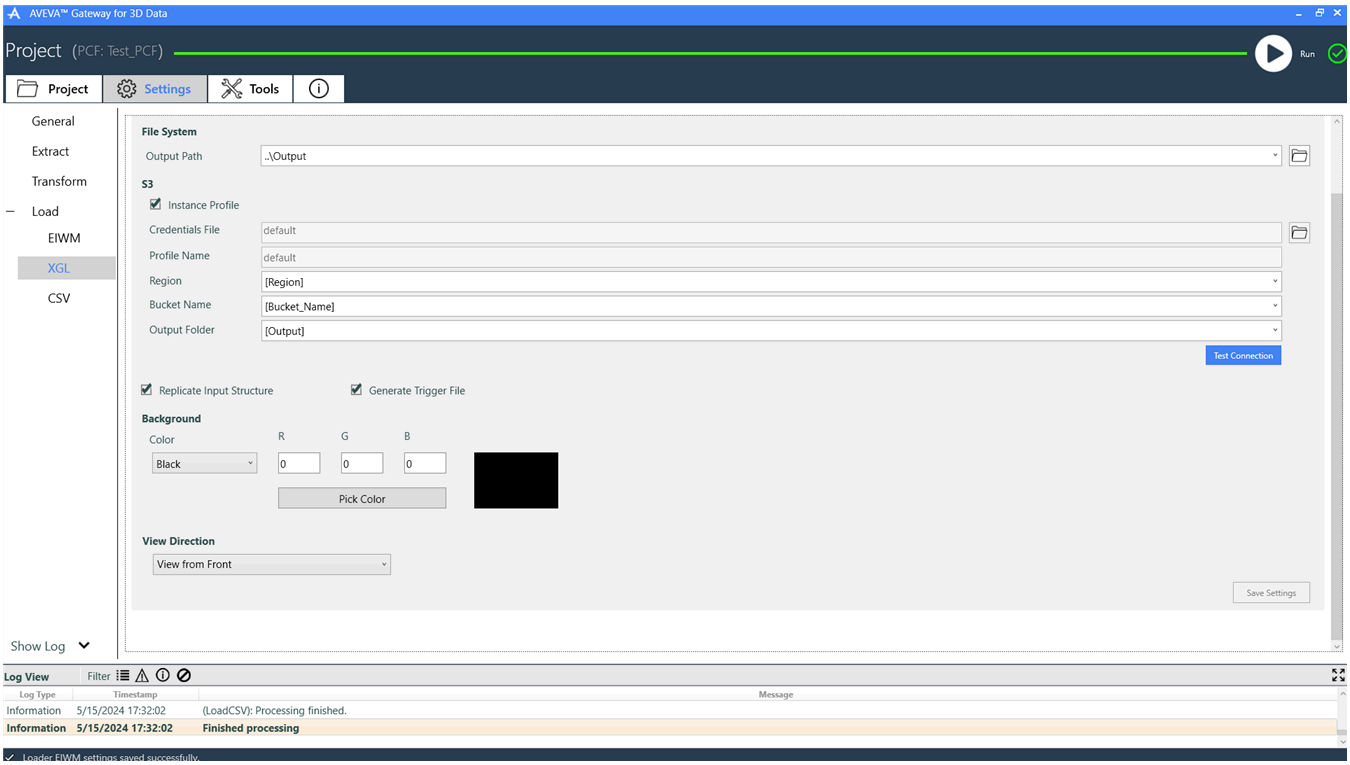
```
<file maxObjectCount="0" suffix="_part_"
suffixNumberFormat="DDD" contextSuffix="" />
```

You can modify the output file formatting as follows:

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

Load XGL

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



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

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

- **Graphics Output Type**: Select the output file type for storing the graphics, such as XGL (XML format) or ZGL (compressed XGL format). You can use **None** option if no 3D graphical data is needed, for example, if only the EIWM file is needed. Selecting **None** means that no graphical data is written, for example, if only the engineering data via EIWM XML is to be imported into AIM.
- **Output Destination**: Select whether the output will be written to a **File System** or an **AWS** or AVEVA Cloud Storage.
  - **File System**:
    - **Output** Path: Type the Output file path or click the **Browse for Folder** tab to enter the output folder path.
  - **S3:** To access an S3 bucket requires the relevant authentication details like credential file, region information, bucket name and output directory. For more information, see Accessing an AWS S3 Bucket.
  - **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.

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

Notes:

- You can set tracking of changes provided by XGL Load on data using annotations feature. For information about annotations, see Appendix C: Tracking Changes in Data.
- If during saving ID (object's or context's) to output EIWM or XGL/ZGL they contain any of illegal characters: "{" "|" "}", they are automatically converted to "_" to prevent issues with loading these files to AIM and warning message is added to the log.

  You can consider applying explicit transformation to change them to any other characters before reaching this stage of the processing.

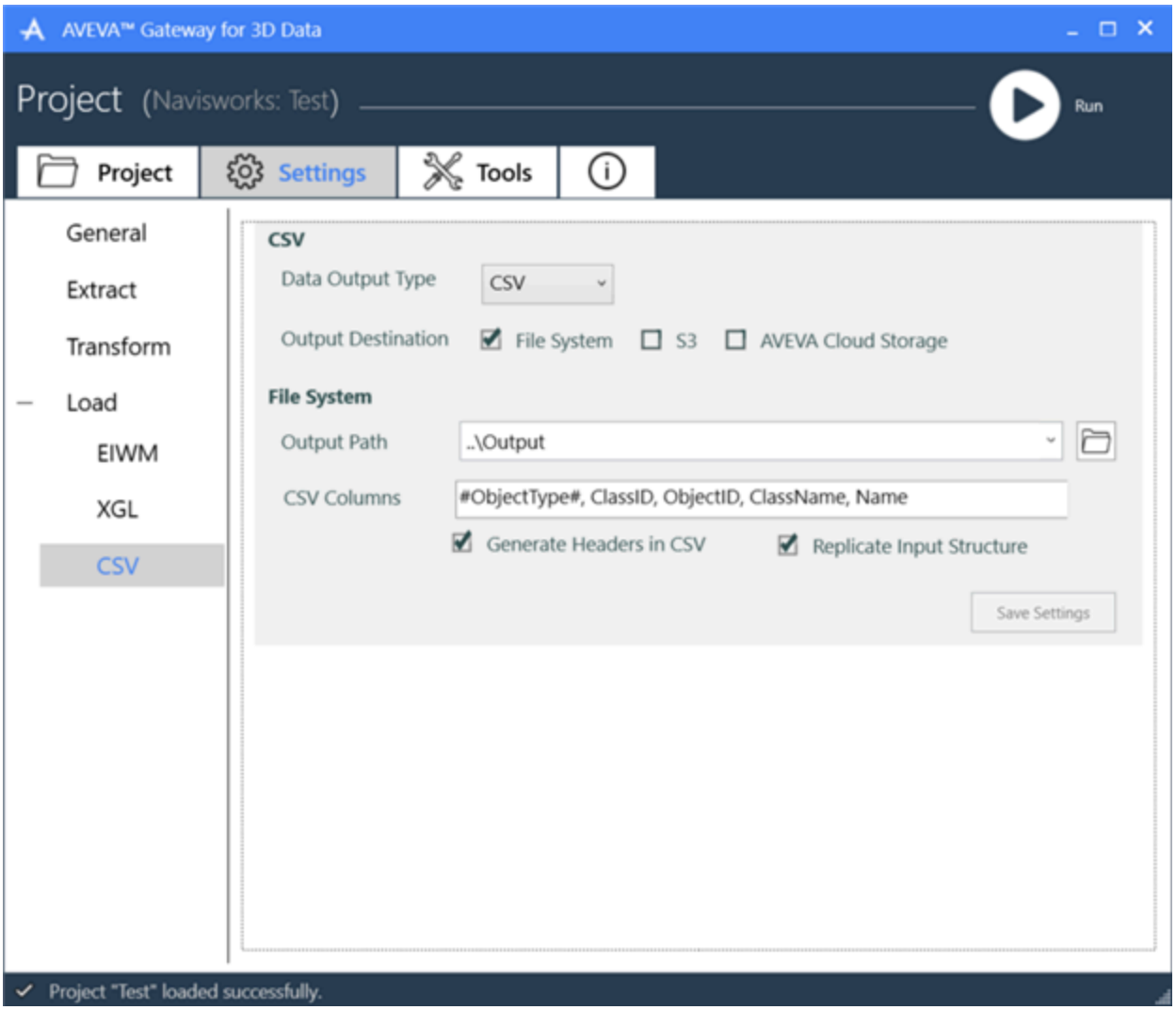
  Log messages are as follows:
  - (EIWM Loader): The text: '{ABC123}' contains some of the illegal characters: "{" "|" "}" and they are replaced with '_'. Consider applying explicit transformation to change them.
  - (XGL Loader): The text: '{ABC123}' contains some of the illegal characters: "{" "|" "}" and they are replaced with '_'. Consider applying explicit transformation to change them.

    Such messages may repeat in the log due to multiple occurrences in EIWM and XGL files.

Load CSV

Load CSV

This option enables you to configure the Load/CSV panel.



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

- **Output Destination**: Select the output destination type to store the output file type in the Destination folder. The destination type can be **Filesystem** or **S3** or Both.
  - **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. For more information about how to configure the column definitions, see Appendix B: CSV Reporting.
- **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.

  Review from here: https://dev.azure.com/AVEVA-VSTS/Interoperability/_workitems/edit/4072022/

Creation of Large CSV Files in Load CSV and Transform's CSV Reports

When the data being written to a CSV file exceeds **1,000,000 rows**, the Gateway automatically splits the output into multiple CSV files, such that for every additional 1,000,000 rows, a new CSV file is generated.

For example:

- The first file will be named: `test_AfterTransformation.csv`
- The second file (for the next 1,000,000 rows) will be: `test_AfterTransformation_1.csv`
- The third: `test_AfterTransformation_2.csv` and so on.

Execute Projects

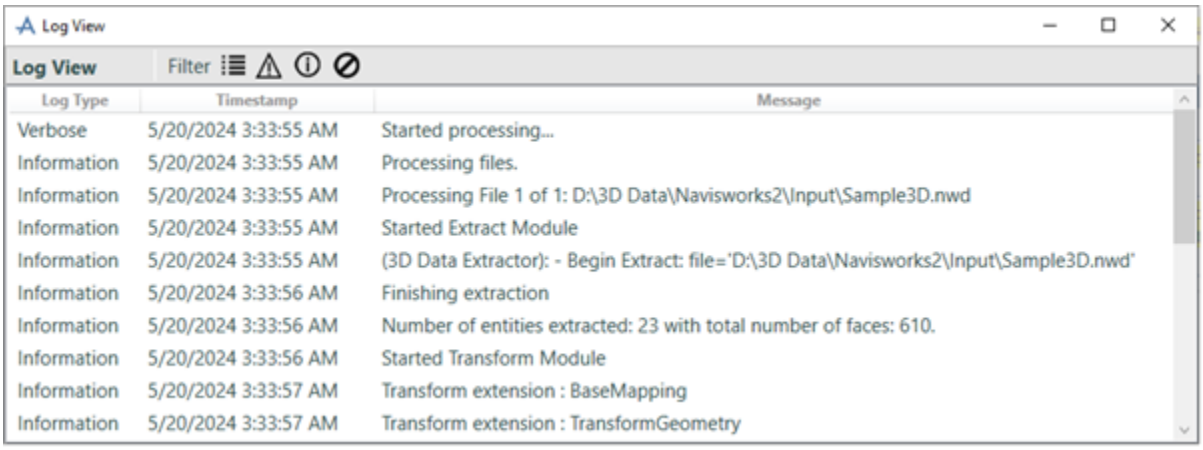
After you have selected the required project settings you can execute the project. While executing the project configuration, the Gateway searches for the input location and generates output files.

To execute the selected project:

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

Note: The input 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:



Notes:

- You can click the four icons on the **Log View** window to get the respective log messages about **Error**, **Warning**, **Information** and **Verbose** based on your selection in the **Logging** field.
- You can also click  to open this Log information in another pop-up window.
- By default, the messages relating to Information level is displayed if you do not select any of the four log level options.
- Timestamp is based on the defined system time set.

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

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

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

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

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

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

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

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

    Note: When loading to S3 bucket the object key is not required and the object key is automatically set to

the output name of EIWM (applicable only to the loader).

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

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

  In this case, the instance profile internally authenticates the call for the user (that is, without the use of any credential file). Instance profiles use an EC2 attached role to authenticate calls to other Amazon Resource Names (ARN).
- **When running the Gateway on their own server (that is, from outside AWS)**: You should rely on the credential file for authentication of their calls to the S3 bucket. A credential file is stored on your system and it contains AWS credentials for accessing AWS resources.
- The Gateway also supports the temporary credentials. The format for temporary credential file is as follows:

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

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

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

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

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

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

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

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

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

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

Access AVEVA Cloud Storage

Prerequisites:

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

Ensure that the following system environment variables are defined:

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

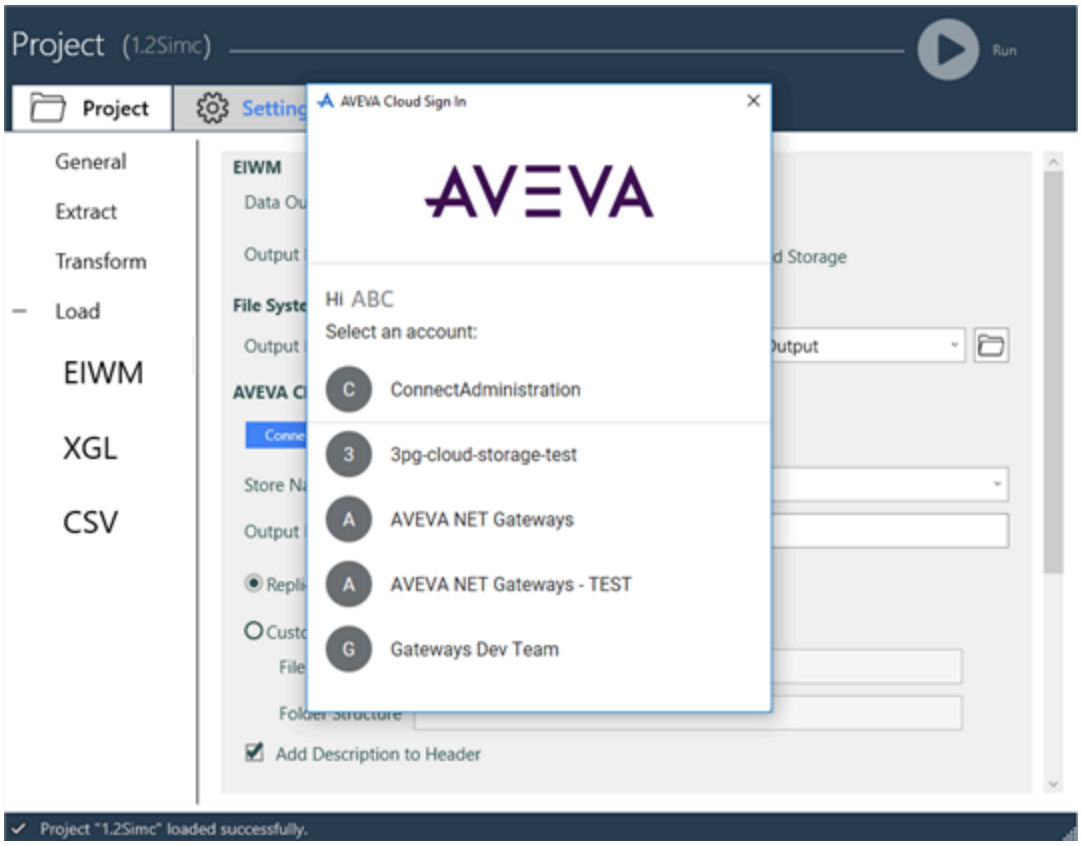
Note: The above-listed environment variables are mandatory for connecting to ACS.

ACS Gateway Configuration:

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

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

1. Select the relevant account that contains the Store (folder) that will be used. When you select this account, it authenticates and establishes a connection with **Connect** and a token is downloaded into **Userprofile/.aveva** folder with the client ID ending with the word "**Credential**", for example, **AVEVA.NET.Gateway.AC3D.ACS_credentials**.

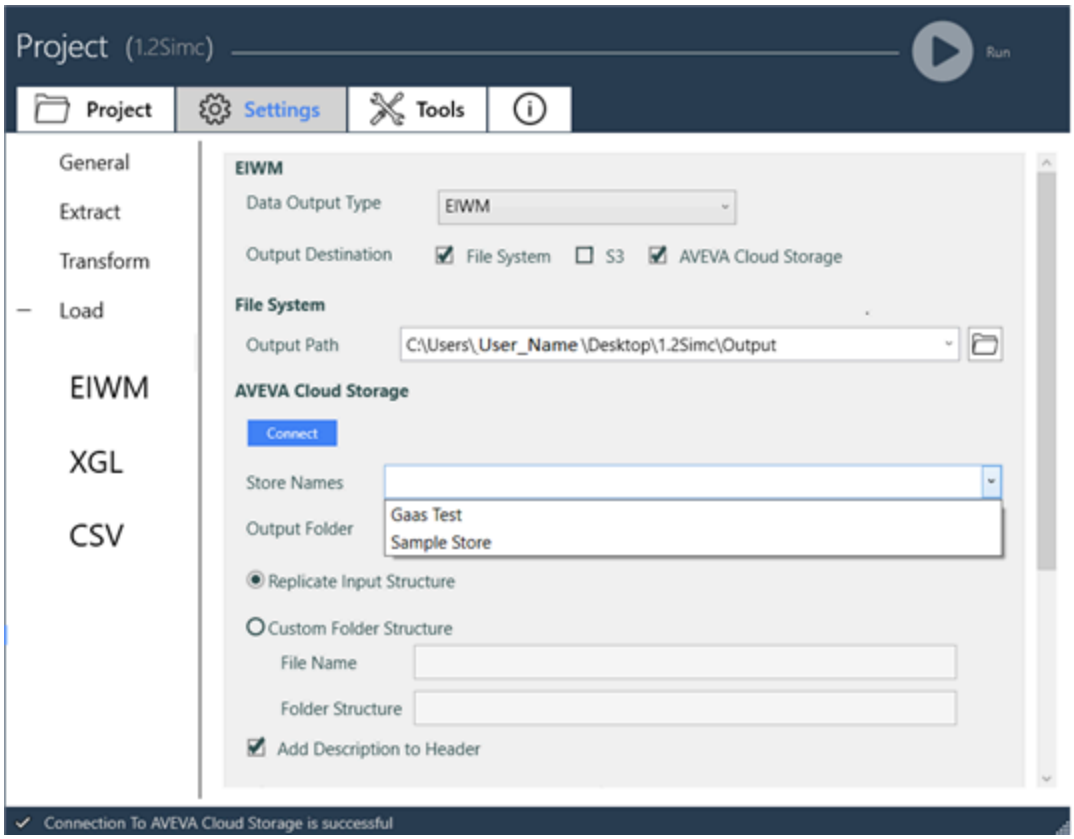


Notes:

- It is assumed that this one account will allow access to all of the ACS folders required by the Gateway.
- The refresh token normally has a lifetime of 6 months. When a token becomes expired, you will get the following error message:

  "Existing token may be expired. Please use **Connect** button from Loader pages in GUI mode to create credentials file."

2. 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.



3. Select the relevant **ACS** Store.
4. Save the changes.

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

```
<Outputs>
  <S3 isSelected="false">
    <Authentication instance="true">
      <CredentialFile path="default" profileName="default" />
    </Authentication>
    <Region />
    <BucketName />
    <OutputDirectory />
  </S3>
  <FileSystem isSelected="true">
    <OutputPath>C:\Output</OutputPath>
  </FileSystem>
<AvevaCloudStorage isSelected="false"><StoreName /><OutputDirectory /></AvevaCloudStorage></Outputs>
<OutputOptions FolderStructure="keepInputFolderStructureForOutput">
  <keepInputFolderStructureForOutput />
  <CustomFolderStructureForOutput>
    <FileName value="" />
    <FolderStructure value="" />
  </CustomFolderStructureForOutput>
</OutputOptions>
<dataType value="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: You can select multiple output destinations simultaneously such as **Filesystem** and **ACS** or **ACS** and **S3** or all the three output destinations together.

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 3D Data\Logs\ThreeDData_Log_<GUID>.txt`

The logs during project execution are logged separately to the log folder as defined by you either om 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 3D Data\AVEVA.NET.Gateways.Data3D.exe" <-cp> <-in> <-xglin> <-pcfin> <-op> <-lp> <-extcp> <-trnsfcp> <-ldrcp> <-context> <-ma> <-ma> <-upgradedProjectLocation>
```

where the parameters are listed as follows:

Command Line Parameter	Description
`-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. The -in parameter is only applicable for AC3D, NWD, IFC and STEP.
`-op <output folder path>`	(optional) Output folder path for all the the output EIWM/XGL/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.
`-ma "<manifest attribute name1>:<manifest attribute value1>"`	(optional) Allows the passing of a manifest attribute value from the command line.

Command Line Parameter	Description
	You can 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 PCF	
`-xglin <XGL input file/folder path/S3 file URL>`	(optional) Input file/folder path for the XGL file. You can use URL to S3 File or Directory. It works only when Extract configuration is set to handle S3 Bucket as Input Source. See Extract Settings for details.
`-pcfin <PCF input folder/file path>`	(optional) Input file/folder path for the PCF files.

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 3D Data\AVEVA.NET.Gateways.Data3D.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 3D Data\AVEVA.NET.Gateways.Data3D.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 3D Data\AVEVA.NET.Gateways.Data3D.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 3D Data\AVEVA.NET.Gateways.Data3D.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 3D Data\AVEVA.NET.Gateways.Data3D.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 3D Data\AVEVA.NET.Gateways.Data3D.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 3D Data\AVEVA.NET.Gateways.Data3D.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 input folders specific for PCF Gateway:

```
C:\Program Files\AVEVA\AVEVA NET Gateways\Gateway For 3D Data\
AVEVA.NET.Gateways.Data3D.exe" -cp "D:\Test\New_1\New_1.xml" -pcfin "D:\Test\New_2\
InputPCF" -xglin "D:\Test\New_2\InputXGL" Takes Input source files for PCF Gateway from
command line parameters: "D:\Test\New_2\InputPCF" and "D:\Test\New_2\InputXGL
```

With project configuration file and project context: `"C:\Program Files\AVEVA\AVEVA NET Gateways\Gateway For 3D Data\AVEVA.NET.Gateways.Data3D.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 3D Data\AVEVA.NET.Gateways.Data3D.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 3D Data\AVEVA.NET.Gateways.Data3D.exe" -cp "c:\GCTproject\GCTproject.xml" -upgradedprojectlocation "C:\UpgradedProject"`

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:

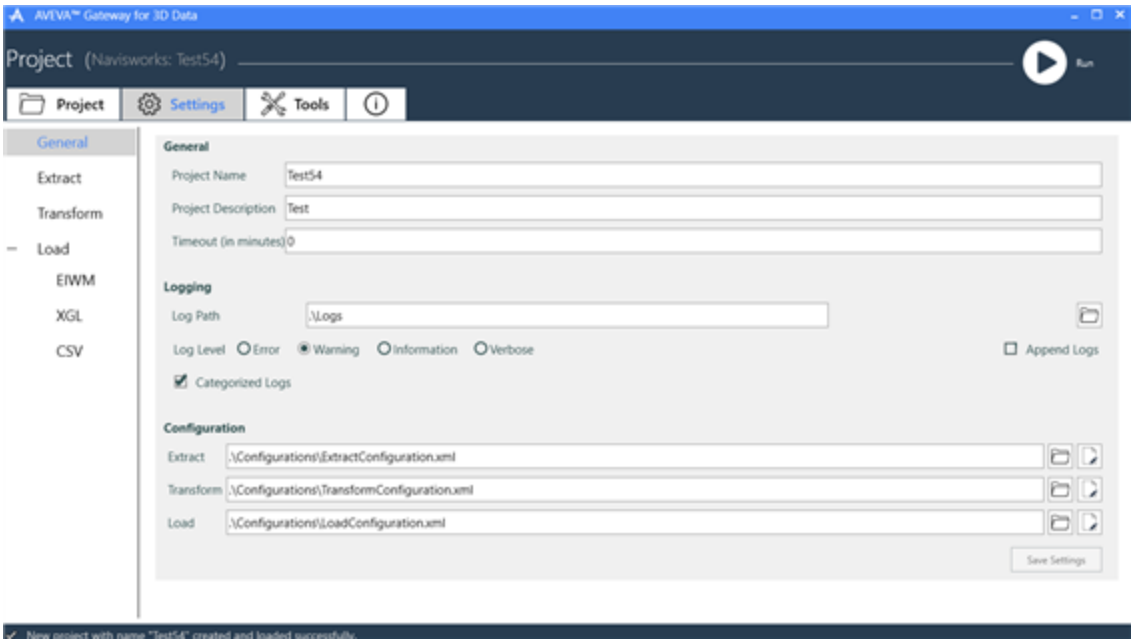
The following command line exit codes are defined for `AVEVA.NET.Gateways.Data3D.exe`.

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

Status Messages from the Gateway Processing

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



This option is controlled by the project setting in the selected section of the Gateway (this option is set to false by default).

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

When set to `true`:

- A JSON report file is written in the same location as the log file. The level of detail in the JSON report file depends on the project settings 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 3D 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

Field	Value Example	Description
		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:

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

The JSON report then contains one or more status messages:

Field	Value Example	Description
"ID"	2	Number of the message.
"DateTime"	"06-02-2020 19 06 43"	Timestamp of the message.
"SubComponent"	"AutoCAD 3D"	Source Gateway's component of the generated message.
"Severity"	"Warning"	Severity of the message: Debug/ Information/Warning/Error/ ErrorProcessingFailed/ErrorCritical.
"Code"	11644681	Unique ID of the message.
"CodeName"	"WAR_EXT_AUT_B1AF09"	Unique name of the message.
"Category"	"Extract"	Category of the message (see the following table.).

Field	Value Example	Description
`"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.
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.

Message Category	Message Description
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.

Appendix A: Mapping Configuration

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

```
<MappingTemplate>
<DataSourceLookups>
<CsvLookupDataSource>...</CsvLookupDataSource>
<MSAccessLookupDataSource>...</MSAccessLookupDataSource>
<MSExcelLookupDataSource>...</MSExcelLookupDataSource>
<MSSqlLookupDataSource>...</MSSqlLookupDataSource>
<OracleLookupDataSource>...</OracleLookupDataSource>
</DataSourceLookups>

<ManifestAttributes>
< Attribute>...</Attribute>
< Attribute>...</Attribute>
</ManifestAttributes>

<Datasets>
<Dataset>...</Dataset>
<Dataset>...</Dataset>
</Datasets>

<TemplateLookups>
<Template>...</Template>
<Template>...</Template>
</TemplateLookups>
```

```
<ObjectMappings regExTimeoutSecs="10">
<Object>...</Object>
<Object>...</Object>
<Object>...</Object>
</ObjectMappings>

</MappingTemplate>
```

Each configuration setting is detailed in the following sections.

LookupDataSource

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

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

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

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

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

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

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

Sheet1 is constructed as follows:

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

If no matching values are found in the data source, you can use the values specified in the `FailAction` element:

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

Notes:

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

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

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

The following data sources are supported:

```
<CsvLookupDataSource
id="CSVLookup"
file="C:\Mapping\Data Value Lookup\Mapping.csv"
separator=","
provider="Microsoft Access Text Driver (*.txt, *.csv)"
sourceColumn="Source"
targetColumn="Destination" />

<MSAccessLookupDataSource
id="MSAccessLookup"
file="C:\Mapping\Data Value Lookup\Mapping1.accdb"
query="SELECT [Source], [Destination] FROM [Table1]"
provider="Microsoft Access Driver (*.mdb, *.accdb)"
sourceColumn="Source"
targetColumn="Destination" />
<MSExcelLookupDataSource
id="ExcelLookup"
file="C:\Mapping\Data Value Lookup\Mapping1.xls"
```

```
provider="Microsoft Excel Driver (*.xls, *.xlsx, *.xlsm, *.xlsb)"
extendedProperties="Excel 12.0; HDR=YES"
query="SELECT [Source], [Destination] FROM [$Sheet2]"
sourceColumn="Source"
targetColumn="Destination" />
<MSSqlLookupDataSource
id="MSSqlLookup"
query="SELECT [Key], [Value] FROM [Table1]"
connectionString="Driver={SQL Server}; Server=serverxxx; Database=Databasexxx;"
sourceColumn="Key"
targetColumn="Value" />
```

Note: Providing UID and PWD is not a secure way of connecting to an SQL Server Database. Giving neither of the UID and PWD as shown above uses Windows Authentication for connecting to an SQL Server database. For backwards compatibility the Gateway also supports older methods, but it's recommended that users should use Windows Authentication for connecting to an SQL Server database.

Previous Methods:

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

Note: Providing explicit values for UID and PWD is not a secure way of connecting to an Oracle Database. Setting `UID=/`; `PWD=`; and `Integrated Security=true` as shown above uses Windows Authentication for connecting to an Oracle database. Please refer to "Connect to the Oracle Database Using Windows Authentication" in the Microsoft learning page.

For backwards compatibility the Gateway also supports older methods, but it's recommended that users should use Windows Authentication for connecting to an Oracle database.

Previous methods:

```
<OracleLookupDataSource
id="OracleLookup"
connectionString="WZz2JoEigxXpWGjsUwdTAxjSG7N8rI61d+aHz503TrfeCNNcLPpmOg=="
connectionStringEncrypted="true"
query="SELECT [Key], [Value] FROM [Table1]"
```

```
sourceColumn="Key"
targetColumn="Value" />
```

or

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

Notes:

- You can store the encrypted connection string details for `MSSqlLookupDataSource` and `OracleLookupDataSource` as the connecting string may contain sensitive information such as `user name` and `password`. The encrypted connection string can be created using the Encrypt tool present in the **Tools** tab as described in Appendix D: Encrypt Utility. If you encrypt the `connectionString` value, the `connectionStringEncrypted` attribute value must be set to true.
- The `extendedProperties` attribute is optional in the below-mentioned Lookup data sources. If not specified, these defaults to the following values:

Lookup Data Sources	Value
`MSExcelLookupDataSource`	`Excel 12.0;HDR=YES`

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

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

- For security reasons, it is advised that the permission of the DB (database) user account should be restricted to read-only.
- When connecting to databases such as SQL Server or Oracle, it is advised that users utilize Windows Authentication.
- In SQL Server this is activated by choosing Windows Authentication as the authentication mode under the user's properties in SQL Server Management Studio.
- In Oracle this is activated by following the procedure detailed in https://learn.microsoft.com/en-us/biztalk/adapters-and-accelerators/adapter-oracle-database/connect-to-the-oracle-database-using-windows-

authentication.

- In order to ensure the security of data which is in transit between the Gateway and an Oracle or SQL Server database, you can configure an SSL encrypted connection between the two.
- As the encrypted password is associated with the local machine where the Gateway is running, other machines cannot use the configuration file directly. You will have to encrypt the password before using the configuration created in another machine. You can obtain an encrypted password using the encryption utility.

Derive 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>
```

Handle Custom/Proxy Objects

AutoCAD 3D 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, you can 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, you can provide a default value in the base mapping if no attribute value is found in the data source. If no value is found in the data source for a mapped property/characteristic, then that will be excluded from the output file unless you define a default for it.

```
<Object>
……
<Attributes>
<Attribute>
<Name value="Pressure" />
<Value value="[max_pressure]" default="NA" />
</Attribute>
</Attributes>
</Object>
```

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

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

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

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

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

TemplateLookups

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

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

Note: In case 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.

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

the attribute value is in the format expected by the transformation sequence.

Example of Compose Delimiter:

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

```
<Transforms>
<Keep pattern="[A-Z]-\d{2}[A-Z]" />
<Compose delimiters="-" constructor="type: {1}, series: {2}" />
</Transforms>
```

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

Description:

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

Regular Expression Evaluation Timeout

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

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

```
<ObjectMappings regExTimeoutSecs="10">
```

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

Timestamp References

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

This placeholder is specified as `[DateAndTime]`.

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

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

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

AutoNumber References

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

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

Notes:

- The value of sequence start value should be an integer greater than or equal to "0". If you enter a negative value, then the sequence start value is defaulted to "1".
- Re-processing another version of the input 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`.

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

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

The Mapping entry in the above example generates an `ObjectID` from the internally generated GUID.

Note: Using `InternalId` is not recommended because the value is entirely independent of the input data and the value of an `InternalId` is not repeated. Hence re-processing the same input 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 "AC3D".
#CURRENT_USER#	User name of the person who is currently logged on to the Windows operating system during execution.
#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.
#DEFINED_SEGMENTS#	FALSE • BaseMapping extension is executed before "AVEVANETPresentationMapping" extension. • No Presentation Mapping. • Lack of 'segments' section in Presentation Mapping. • 'Segments' section exists without any 'Segment' node. TRUE • Segment's node exists (one or more).
#KEEP_FOLDER_TREE#	TRUE, if input folder hierarchy is preserved in output generation.
#TARGET_LOCATION#	Output folder path.
#SEGMENT_NAME#	Segment name of the graphics drawing.
#UNITS#	Source drawing units.

Note: Do not change the following attributes:

- #INPUT_FOLDER#
- #TYPE#
- #MANIFEST#
- #CURRENT_USER#
- #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>
```

User-Defined Manifest Attributes

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

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

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

```
<Object>
<IncludeAssociatedObjectAttributes 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#".

Review from here: https://dev.azure.com/AVEVA-VSTS/Interoperability/_workitems/edit/4110096/

You can also include user defined manifest attributes in conditions to be used for filtering by using the <Conditions> attribute "manifestMatch". Consider the below example:

```
<Object>
<Conditions>
<Attribute name="USER_DEFINED_ATTR" manifestMatch=" USER_ATTRIBUTE1" />
</Conditions>
<Attributes>
<Attribute>
<Name value="User Attribute" />
<Value value="[USER_ATTRIBUTE1]" />
</Attribute>
</Attributes>
</Object>
```

In this example, the manifestMatch parameter is used to select only objects that both have an attribute named "USER_DEFINED_ATTR" and this attribute value matches the value of the manifest object's attribute named "USER_ATTRIBUTE1". The <Attributes> node will then update or create an attribute named "User Attribute" with the value of the manifest attribute "USER_ATTRIBUTE1". Note that when the manifestMatch parameter is used, the Gateway automatically makes the manifest object's attributes available when assigning values, so that <IncludeAssociatedObjectAttributes> of the "is referenced in" association is not needed nor do you need to refer to manifest attributes with "associated:" prefix.

End review here

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

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

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

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

Note: You must not remove either the Manifest object or any of its attributes as these attributes such as #MODEL_NAME#, #TARGET_LOCATION# are needed for loader functions. You can control whether the Manifest object is included in the EIWM as the Document object via the createDocumentObject option. For more information, see Load Settings.

The following example shows how to remove the attributes from other objects without affecting the Manifest object.

```
<ObjectMappings regExTimeoutSecs="10">
<Object>
<Conditions>
```

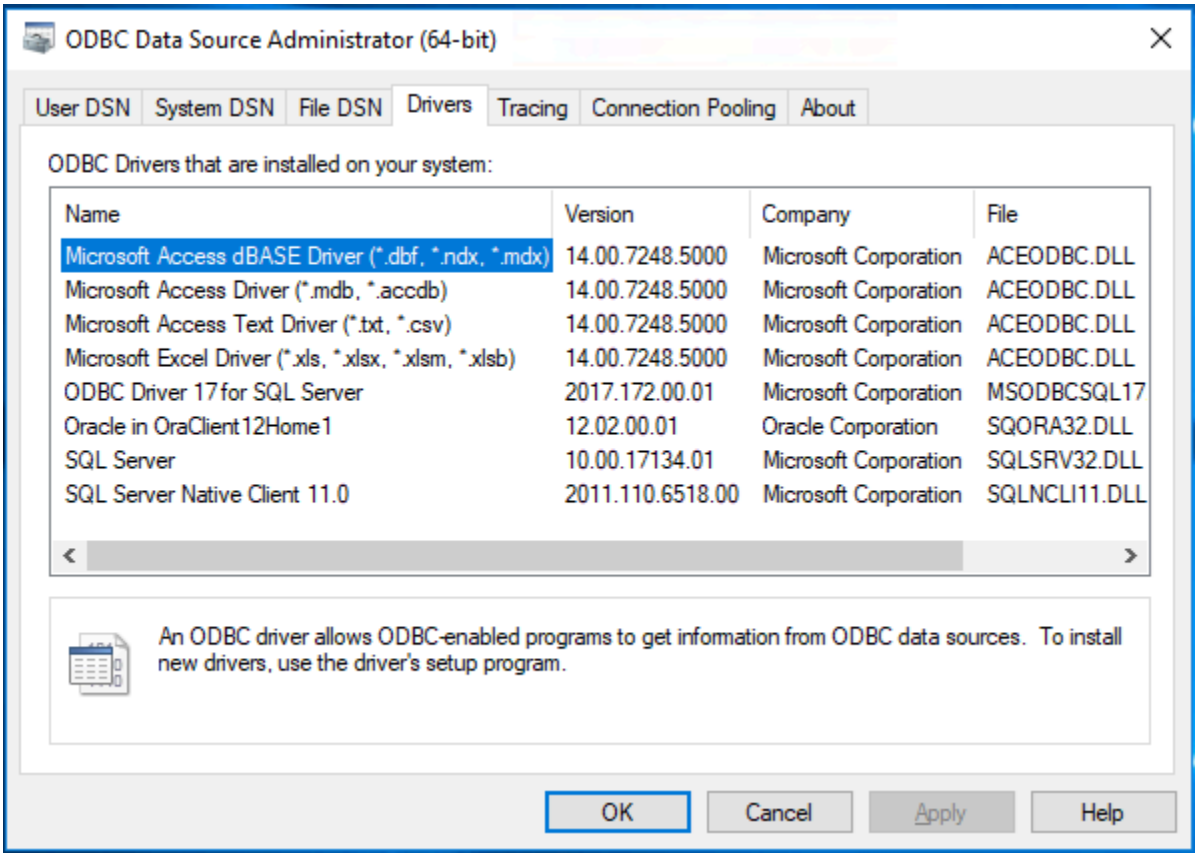
```
<Attribute name="#TYPE#" pattern="manifest$" />
</Conditions>
<TemplateID id ="default" />
</Object>
<Object>
<TemplateID id ="default" />
<ObjectID value="[TAG]"/>
<ClassID value ="[Classification]" />
<Attributes>
<Attribute name="*" >
<Remove />
</Attribute>
</Attributes>
</Object>
</ObjectMappings>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

Migrate Oledb to ODBC

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

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

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



Connection String Changes

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

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

The following are the connection string changes:

Excel:

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

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

This should be updated to support an ODBC driver:

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

Access:

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

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

This should be updated to support an ODBC driver:

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

CSV:

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

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

This should be updated to support an ODBC driver:

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

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

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

SQL Server:

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

```
connectionString="Provider=SQLOLEDB; Server=serverxxx; Database=Databasexxx; User
```

```
ID=userxxx; Password=pwdxxx;"
```

This should be updated to support an ODBC driver:

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

Note: Providing UID and PWD is not a secure way of connecting to an SQL Server Database. Giving neither of the UID and PWD, as shown below, uses Windows Authentication for connecting to an SQL Server database.

```
connectionString="Driver={SQL Server}; Server=serverxxx; Database=Databasexxx;"
```

'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" />
```

Notes:

- Providing explicit values for UID and PWD is not a secure way of connecting to an Oracle Database. Setting UID=/; PWD=; and Integrated Security=true as shown below uses Windows Authentication for connecting to an Oracle database.
- ```
  <OracleLookupDataSource id="OracleLookup" connectionString="Driver={Oracle in
  OraClient12Home1};
  DBQ=serverAddressxxx;
  UID=/; PWD=; Integrated Security=true;"
  query="select Source, Destination from DocumentLookup"
  sourceColumn="Source" targetColumn="Destination" />
  ```
- Alternative ODBC Drivers are available, but the connectionString details will be different. Examples of these are:
  - https://learn.microsoft.com/en-us/sql/connect/odbc/download-odbc-driver-for-sql-server (for SQL)
  - https://www.oracle.com/in/database/technologies/releasenote-odbc-ic.html (for Oracle)

Base Mapping Types

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

Object Mapping

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

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

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

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_XGL#`: Sets output name for XGL.
- `#TARGET_NAME_CSV#`: Sets output name for CSV.

Example Base Mapping:

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

Notes:

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

Attribute Mapping

Attribute Mapping determines how attributes from the source system are transformed to match the

requirement of the target system. For each attribute from the source system, each mapping entry is examined to determine whether the attribute matches.

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

Attribute mapping entries permit the following Values to be specified:

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

A sample Attribute mapping example is as follows:

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

Attribute Mapping entries can be broadly classified into the following:

Create:

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

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

This mapping entry adds a new attribute named `Identifier` to the source system. The attribute value is taken directly from the value of the `Tag` attribute from the source system.

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

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

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

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

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

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

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

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

Update:

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

```
<Object>
...
<Attributes>
```

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

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

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

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

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#"

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

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

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

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

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

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

Dataset Mapping

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

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

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

<ObjectMappings regExTimeoutSecs="10">
```

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

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

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

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

Note: The Dataset is assigned a `ClassID` of '`DATASET`' if not defined in the mapping.

Association Mapping

Association Mapping allows extracted associations to be customized in the output. It also permits new

associations to be created in the output by transforming attributes from the source system.

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

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

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

A sample Association mapping is as follows:

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

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

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

Multi-Value Attributes

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

separate association is created in the output for each value stored in the attribute.

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

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

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

`keepUnmappedAssociations` **Setting**

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

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

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

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

Notes:

- The k`eepUnmappedAssociations` 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.

- `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>
```

This syntax can also be used when an object to be removed has a graphical representation which also needs to be removed. In this case the association type is "GraphicalObjectAssociation":

```
<Object>
<Conditions>
<Attribute name="ObjectID" pattern="A002" />
</Conditions>
<Remove />
<Associations>
<Association relationship="GraphicalObjectAssociation">
<Remove includeTarget="true"/>
</Association>
</Associations>
</Object>
```

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.

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"`.

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

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

Create new associations via attributes containing the target identifier:

Relationships such as in hierarchies can be created via new associations between the related objects when a source object has an attribute containing the identifier of the target object. If the target object was already extracted or created by a previous transformation, then an association will be created between the source and target objects. If the target object doesn't exist, then a new object will be created and assigned an `ObjectID` corresponding to the source attribute value, and an association will be created between this new object and the source object.

Example base mapping configuration to create an association based on an attribute value:

```
<Object>
<Associations>
<Association>
<Type value="is a part of" />
<Conditions>
<Attribute name="IfcSpatialContainer GUID" />
</Conditions>
<TargetID value="[IfcSpatialContainer GUID]" />
</Association>
</Associations>
</Object>
```

ObjectID Mapping

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

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

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

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

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

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

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

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

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

For example, if the initial `ObjectID` value is `"3$pEhtFpv31QFndkpGikoC_this pump-01A is new"`, the final value after `<Transforms>` will be `"InsertBeforepump-01AInsertAfter"`.

Attributes from associated objects can also be used to modify the value of ObjectID, as shown below:

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

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

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

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

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

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

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

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

ClassID Mapping

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

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

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

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

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

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

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

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

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

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`.

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

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

ContextID Mapping

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

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

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

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

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

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

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

You can use Lookups and transforms to modify context value.

```
<Object>
```

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

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.

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

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

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

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

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`.

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

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

Search Criteria Mapping

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

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

```
<SearchCriteria attribute = "Content">
<Search>
<Patterns>
```

```
<Pattern value = "[A-Z]-\d{3}"/>
<Pattern value = "[A-Z]-\d{3}[A-Z]"/>
</Patterns>
<ClassID value = "Document" />
<ContextID value = "Test"/>
<TemplateID id = "default" />
<Attributes keepUnmappedAttributes = "true">
<Attribute>
<Name value = "Description" />
<Value value = "[Content]" />
</Attribute>
</Attributes>
<Associations>
<Association>
<Type value = "Modified association" />
<TargetID value = "abc123" />
<TargetClassID value = "Document" />
</Association>
</Associations>
</Search>
</SearchCriteria>
```

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

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

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

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

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

If two Patterns are defined:

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

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

If the order of the Patterns is reversed:

```
<Search>
<Patterns>
```

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

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

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

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

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

Note: 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.

Expand Attribute Mapping

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

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

For example, suppose an input attribute called Content has the value Material, Steel, you can create a new attribute called Material with a Value of Steel using the following mapping. Note that as the separator is not defined, the default value of comma is assumed.

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

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

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

```
</Attribute>
</ExpandAttributes>
```

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

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

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

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

Either value or index attribute can be used with `Name`, `Value` and `Units` elements, but not both for the same element.

3D Model Hierarchies

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

In AutoCAD 3D 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 hierarchy is reflected in the object explorer tree structure.

Presentation Mapping

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

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

Section with Materials Mapping

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

Example:

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

Section with Colours Mapping

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

Example:

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

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

- RGB, for example, 128, 0, 128.
- Known colours (HTML colour standard), for example, Red.
- HTML hexadecimal: `#E3D3D3`.
- Wildcard "*" in `fromColour` attribute to map all colours not affected by previous colour mapping items.
- AutoCAD Color Index (ACI) mapping colours can be specified in `fromColour` attribute in format `ACI<colourIndex>`, for example,

  `fromColour="ACI170"`.

Section with Segmentation of Data:

This section is used to segment the model to produce a subset or subsets of graphical objects into XGL files. You can segment a model in four ways:

- **According to any attribute value:**

  This segments the model according to any attribute value. For example, `fromValue="*" fromAttribute="Pressure"'` generates a model that contained only the objects with a `Pressure` attribute of any value.
- **According to extents of the geometric model** (min and max points and additional `includeParts` attributes):

  This segments the model by its extents represented by two points max and min. The `includeParts="true"` (or false) attribute determines whether objects that cross this border are included. By default, the attribute value is false.
- **According to any association via three types of attributes:**
  - `fromAssociation`: The value provides the associated object's name.
  - `fromTargetAttribute`: The value provides the associated object's attribute name.
  - `fromTargetValue`: The value provides the associated object's attribute value.
- **When an object is not handled by any segment node:**

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

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

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

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

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

Example:

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

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

Example of segmented output with no separator and suffixNumberFormat:

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

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

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

You can name the segment by using attributes:

- `to`:

  For example, `to="[#MODEL_NAME#]"` - Sets the segment to value of attribute `'#MODEL_NAME#'.`
- `toSuffix`:

  For example, `toSuffix="_seg1"` - Adds suffix to current name. For example, if input model is 'house' then segment name will be 'house_seg1'.

Process 2D Entities

By default, the Gateway processes the 2D entities in a 3D model as objects.

If you do not want to extract any specific type of 2D graphics, you can use the following base mapping to remove it from the output model. You can use the Remove feature, as shown in the following example:

```
<Object>
<Conditions>
<Attribute name="autocad type" pattern="Polyline" />
</Conditions>
<Remove />
</Object>
```

The full domain of `"autocad type"` 2D entities contains the following:

- Circle
- Arch
- Hatch
- MText
- Line
- Polyline
- Text

Transform Geometry Configuration

The **transformCoordinates** and **matchPoints** functions can be used to transform the input 3D model from one coordinate system to another.

transformCoordinates Configuration:

The transformCoordinates parameters allow you to translate, rotate and scale the geometry in the 3D model:

- **Translation**: Moves the model from one location to another without changing its size, shape, or orientation.
- **Rotation**: Rotates all the points in the model about its origin separately in X, Y, and Z dimensions.
- **Scaling**: Changes the size of the model by multiplying it with a scale factor, same for each axis.

Each action in **transformCoordinates** is applied in the configured order. This can impact the accuracy of the overall transformation. For example, if both a rotation and a large offset are required, then it is usually more accurate to apply the translation and then the rotation. Note also that if the rotation was done first then the translation would need to be defined relative to the new rotated coordinate system.

Example of transformCoordinates:

```
<transformGeometry xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" sourceProductName="AVEVA Gateway for 3D Data"
componentName="TransformGeometry" componentVersion="2.12.0.0"
<transformCoordinates leaveOriginalModel="false" objectsTypes="3D">
<action type="scale" factor="0.05” />
<action type="translate" x="10.0” y=”10.0” z =”0.0"/>
<action type="rotate" x="45.0” y=”10.0” z=”0.0"/>
```

```
</transformCoordinates>
</transformGeometry>
```

Note: Transformations are applied in the order they are configured.

matchPoints Configuration

The matchPoints node is an alternative method for transforming the coordinates that simply relies on defining the coordinates of three points in common between the source model (that is the input IFC file) and a target model that may have been previously imported into AIM but has a different coordinate system to the source model. The Gateway then automatically applies the coordinate transformation that will then align the source model with the target model. The **matchPoints** example configuration should define the coordinates of the same three points in the source and target models. Ideally these should be well-separated and in different planes to improve the accuracy of the transformation.

```
<matchPoints leaveOriginalModel="false" objectsTypes="3D">
<sourceCoords>
<point1 x="0.0” y=”0.0” z=”0.0"/>
<point2 x="0.0” y=”100.0” z=”0.0"/>
<point3 x="0.0” y=”0.0” z=”100.0"/>
</sourceCoords>
<targetCoords>
<point1 x="10.0” y=”0.0” z=”0.0"/>
<point2 x="0.0” y=”110.0” z=”0.0"/>
<point3 x="10.0” y=”0.0” z=”100.0"/>
</targetCoords>
</matchPoints>
```

The effect of these three points would be to translate the source model by 10 units in the X direction and rotates it by 90 degrees in the XY plane.

The sequence of the points must be the same so that point 1 in the source system is equivalent to point 1 in the target system. An error is raised if the relative shapes of the triangles defined by the two sets of points are different.

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

The Gateway generates the CSV report during transformation and customizes the type of the information written

to the CSV file. You can also generate multiple reports by using different suffixes and output paths for CSV files.

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

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

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

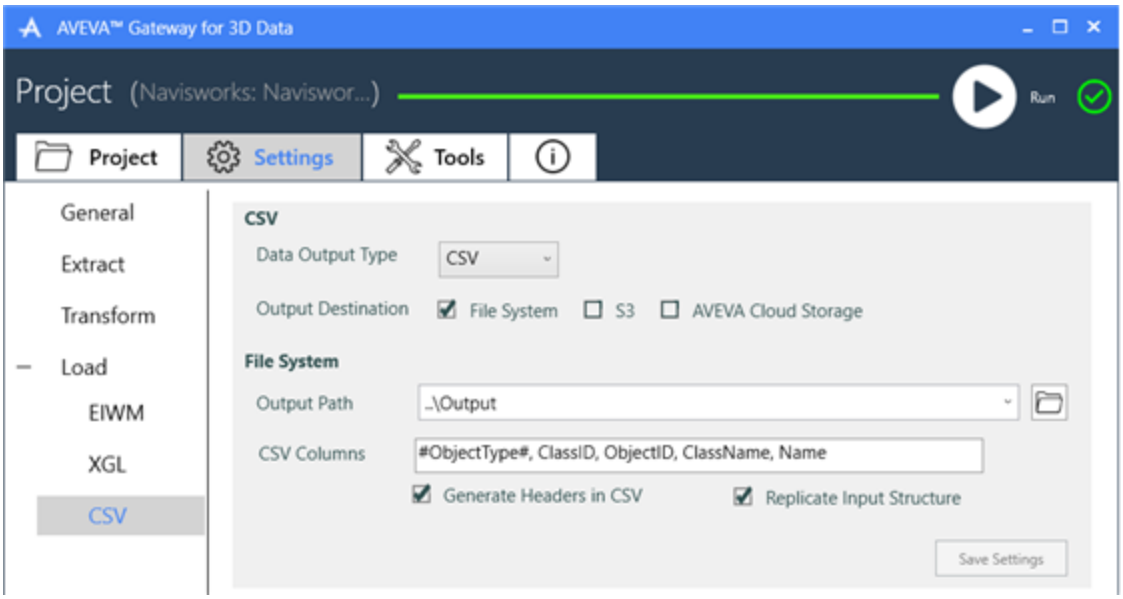
```
<extension name="CSVReporting">
<csvReport
suffix="_AfterTransformation"
addHeaderRow="true"
columns="#InternalId#, ClassName">
<column value="#Tracking:
DateTime
ModuleName
EventType
ChangedMember
PrevAttributeName
AttributeName
PrevAttributeValue
AttributeValue
PrevAssociationType
AssociationType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

Note: The optional '**annotations**' feature described in Annotation provides additional information which also can be reported. In the above example, after keyword `#Tracking:` there are listed tracking attributes that store data about creation, modification and deletion operations and can be inspected via CSV reporting as any other information.

Reporting after transformation configured from GUI

This CSV report is generated when **CSV** value is selected in **Data Output Type** combo list in **Load>CSV** tab. 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 tab:



Settings

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

- `columns = "<keywords, names...>"`: (**Mandatory**) Sets number of columns, each separated by a comma.
  - `#Attribute: <name>` - (**Optional**) Sets the name of the attribute to select the attribute values that will be shown in the CSV column for all extracted data entities. Name of the attribute can also be written without `#Attribute`. `#Attribute` is added to maintain consistency with other keywords. Besides standard objects' attributes, you can also collect different data using keywords:
    - `#InternalID#` - Specifies the Internal Globally Unique Identifier (GUID) of the object.
    - `#SourceID#` - If available, the source ID of the object specified in the source data, for example, the STEP line number in IFC files.
    - `#ObjectType#` - Specifies the internal type of the object, for example: `EngineeringObject`, `GraphicalObject2D`.

      These attributes cannot be changed but may be helpful in tracking the history of the objects or filtering the CSV report by Object Type.
  - `#Association`: `<type>` - Checks if specified association type is available in objects. CSV cells indicate **AVAILABLE** status if the corresponding object contains selected association type.
  - `#Associations` - Checks if associations are available in objects. All types of associations are present in the extracted data and each CSV column represents one type of association. CSV cells indicate AVAILABLE status if the corresponding object contains association type defined in the CSV column.
  - `#Association:<type>` and `#TargetAttribute:<name>` - Used together to first set the name of the association and then the name of the attribute whose value should be shown in the CSV column.

    Note: An association may point to many referenced objects and in this case values of attributes of all these objects are added to the cell in the CSV column, each separated by a comma.
  - `#Unit` - Used only with one of the three keywords: `#Attribute: <name>`, `#Attributes` or `#TargetAttribute:<name>`. If attribute contains unit, it is added to the value after putting a single space.

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

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

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

- `suffix = "<file name suffix>`: (**Optional**) Defines a suffix to add to the CSV file name. It is used only in Transform configuration.
- `outputFolder = "<output folder path>"`: (**Optional**) Stores the CSV files in the location indicated by this setting. Otherwise, files are passed to the output folder set in Load settings. It is used only in Transform configuration.
- `addHeaderRow = "<true/false>"`(in **Generate Headers in CSV** checkbox): (Optional) Sets to "**True**" to write the names of the attributes or associations to the first row of the CSV file. This setting is enabled by default.

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

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

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

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

Object Parameters Available for the CSV Report

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

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

When a relationship is used, the property names from each relationship get concatenated, for example, HasAssociations:Materials:Name.

Note: Objects containing an attribute "#EXCLUDE_FROM_CSV#" will not be included in the output .csv file. For more information, see Attribute Mapping.

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>
```

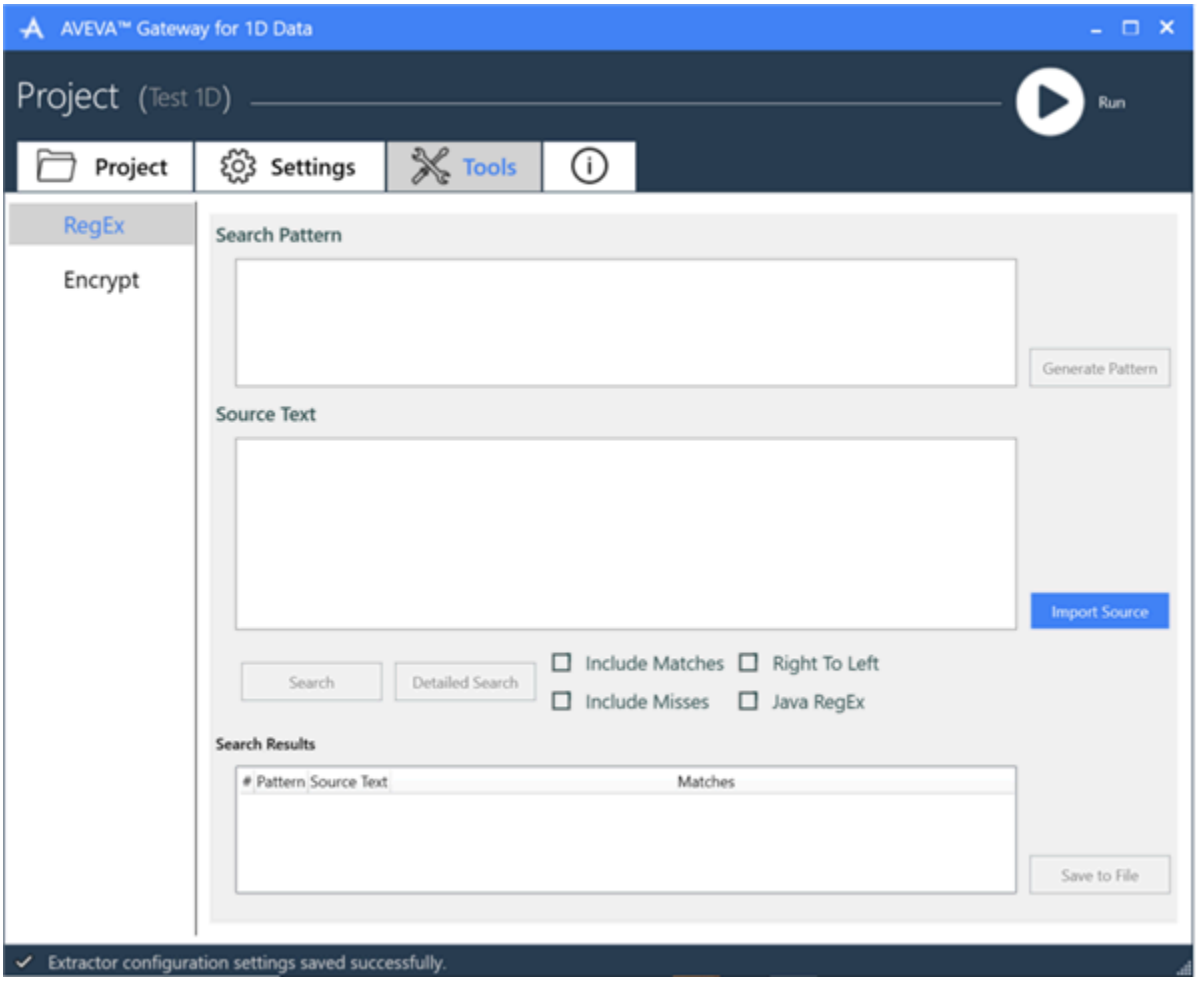
Example of using ObjectIdTracking level in BaseMapping Extension Node:

```
<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.

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.

RegEx Utility Options	User Action
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.
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.

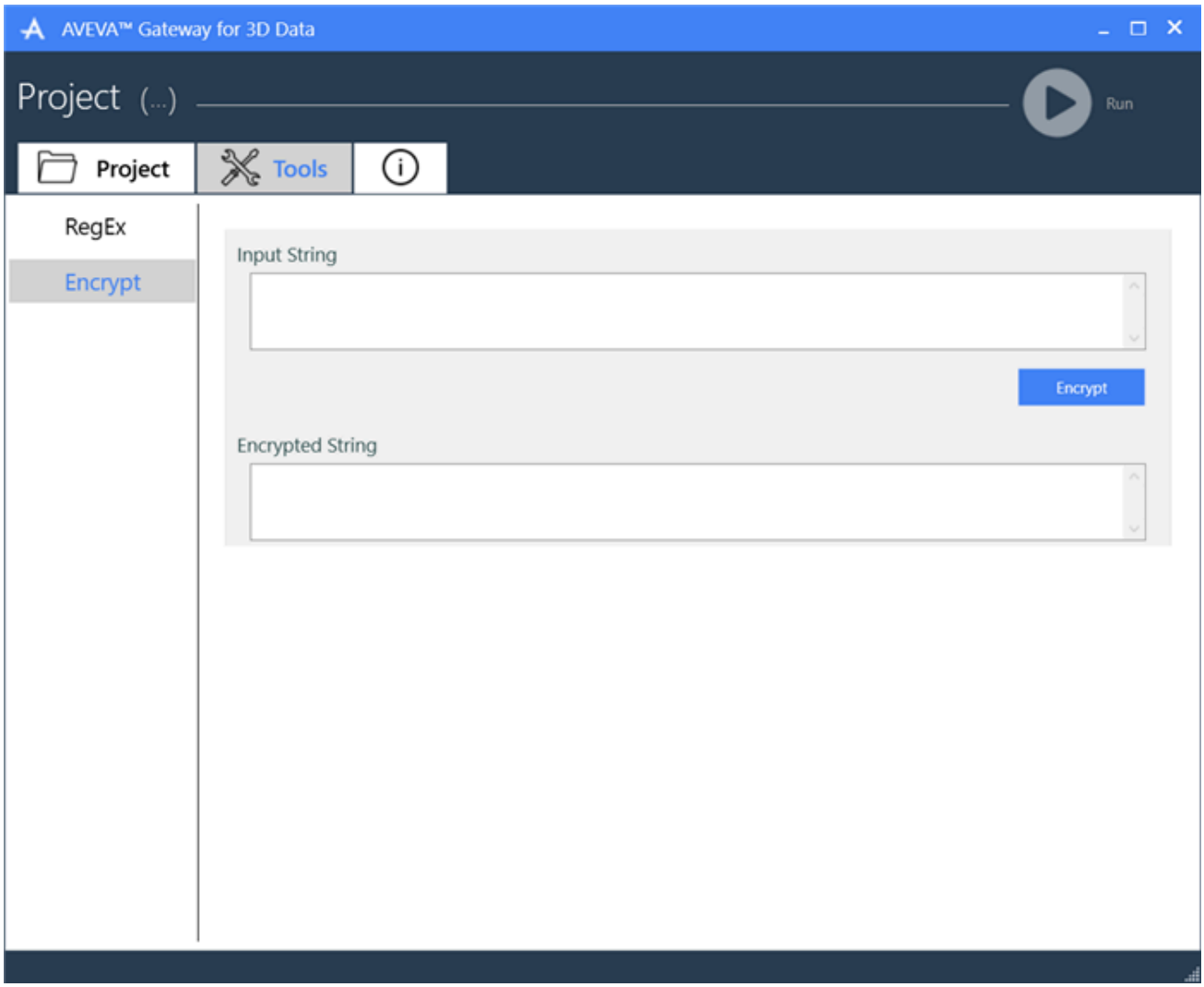
Appendix E: Encrypt Utility

For the lookup data sources that have connection strings, you can store the connection string encrypted in the configuration file, as the connection string may have sensitive information such as a **user name** and **password**.

To encrypt the connection strings:

1. Specify the connection string.
2. Select the option `Tools > Encrypt` and enter the connection string in the Input String box.
3. Click **Encrypt**.

   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.



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

Appendix F: Reserved Attributes Used in the Gateway

The following reserved attributes are used by the Gateway to apply and implement mapping. They should therefore not be present in the extracted source data, unless they are being used for this purpose.

- `ObjectId`: It is used to map Id to the EIWM object. During the transformation phase, the Gateway creates "`ObjectId`" attribute or overwrites it, if already present.
- `ObjectName`: It is used to map Name to the EIWM object. During the transformation phase, the Gateway creates "`ObjectName`" attribute or overwrites it, if already present.
- `ClassId`: It is used to map `Class Id` to the EIWM object. During the transformation phase, the Gateway creates "`ClassId`" attribute or overwrites it, if already present.
- `ContextId`: It is used to map Context to the EIWM object. During transformation phase, the Gateway creates

"ContextId" attribute or overwrites it, if already present.

- Revision: It is used to map revision to the EIWM object. During the transformation phase, the Gateway creates "Revision" attribute or overwrites it, if already present.
- TemplateId: It is used to map Template Id in which the EIWM object is present. During the transformation phase, the Gateway creates "TemplateId" attribute or overwrites it, if already present.
- IncidentalClassification: It is used to map IncidentalClassification association to EIWM object. During the transformation phase, the Gateway creates "IncidentalClassification" association or overwrites it, if already present.

If any of these attributes are present in the source data, then appropriate mapping should be configured to ensure that they do not have unintended effects. For example, if the source data has a "Revision" attribute, then mapping can be used to rename this attribute, for example, "Source_Revision". Otherwise, the EIWM loader interprets this to be the Revision value for the relevant object.

Reporting for Reserved Attributes in Extracted Data

You can generate a CSV report of the extracted data to check if there are any reserved attributes present in the extracted data.

This CSV report should be generated before any other transformation processing. You can insert the following transformation extension node before all other extensions, in the transformation configuration file.

```
<extension name="CSVReporting" >
<csvReport suffix="_BeforeTransformation" columns="ObjectId, ObjectName, ClassId,
ContextId, Revision, TemplateId, IncidentalClassification" addHeaderRow="true" />
</extension>
```

This will generate a .csv file containing information about engineering objects that may contain these attributes. If there are any objects that have a value associated to any of the above attributes, it might cause unexpected results.

Appendix G: Handle System Attributes

The Gateway creates "System Attributes" that are used to store metadata about the input source and control the behavior of the EIWM Loader. Transformation prevents modification of many of these system attributes. However, transformation can also use these values in filters or update other attributes with their values.

During processing, the Gateway uses these system attributes, as listed in the following table:

Attribute	Properties			Usage
	Occurs in Object Type	Base Mapping	Presentation Mapping	EIWM Loader
#TYPE#	manifest	R	R	-
#MANIFEST#	manifest	R	R	EXPORT
#INPUT_LOCATION#	manifest	R	R	EXPORT

Attribute	Properties			Usage
#INPUT_FOLDER#	manifest	R	R	EXPORT
#GRAPHICS_FORMAT#	manifest	R	R	-
#UNITS#	manifest	R	R	EXPORT
#DEFINED_SEGMENTS#	manifest	R	R/W	-
#KEEP_FOLDER_TREE#	manifest	R	R	-
#TARGET_LOCATION#	manifest	R	R	EXPORT
#FILE_TIME_STAMP#	manifest	R	R	-
#UNITS-BORE#	manifest	R	R	EXPORT
#UNITS-CO-ORDS#	manifest	R	R	EXPORT
#UNITS-BOLT-LENGTH#	manifest	R	R	EXPORT
#UNITS-BOLT-DIA#	manifest	R	R	EXPORT
#UNITS-WEIGHT#	manifest	R	R	EXPORT
InfoLocator	manifest	R	R	EXPORT
#material_ambient_colour_RGB	Engineering Object	R	R/W	EXPORT
#SEGMENT_NAME#	Manifest and Engineering Object	R	R/W	-
keepUnmappedAssociations	Engineering Object	R/W	R	USED FOR FILTERING
keepUnmappedAttributes	Engineering Object	R/W	R	**USED FOR FILTERING**

Notes:

R – The attribute is read only. It is not possible to modify its value.

R/W – The attribute can be modified in transformer configuration.

Example – Copying of System Attribute to Normal Attribute:

In this example, the attribute #GRAPHICS_FORMAT# is not exportable to EIWM. To include the attribute's value in EIWM, you can create or re-use a non-system attribute and set its value to the value of the system attribute.

```
<Attribute>
<Conditions>
<Attribute name="#GRAPHICS_FORMAT#" pattern="^.*$"/>
</Conditions>
<Name value="GRAPHICS_FORMAT" />
<Value value="[#GRAPHICS_FORMAT#]"/>
</Attribute>
```

The example above will add the engineering attribute named GRAPHICS_FORMAT.

Appendix H: Known Limitations

The following are some of the limitations of Gateway for 3D Data:

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.

Missing Layer Attributes in Navisworks files Derived from MicroStation 3D DGN Files

When NWD files are derived from an import of MicroStation 3D models, the 'layer' attributes of objects are not extracted by the Gateway. Navisworks files derived from other 3D sources such as AutoCAD and IFC are unaffected by this limitation.

AVEVA Group Limited
High Cross
Madingley Road
Cambridge
CB3 0HB
UK

Tel +44 (0)1223 556655

www.aveva.com

To find your local AVEVA office, visit
www.aveva.com/en/about/about-aveva/locations/