AVEVA™ Gateway for 1D Data

Version 6.0.0

Publication date: Tuesday, November 4, 2025

Publication ID: 1567722

Contact information

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

https://sw.aveva.com/

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

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

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

Contents

Chapter 1

What is new in AVEVA Gateway for 1D Data?

These release notes explain the feature updates, new enhancements and known issues of AVEVA Gateway for 1D Data. They are usually released as part of a major product release.

This section describes the release rotes for the AVEVA Gateway for 1D Data.

Chapter 1 Gateway for 1D Data 6.0.0

A full release **71461** of *AVEVA Gateway for 1D Data 6.0.0,* which replaces the full release 71417 *AVEVA Gateway for 1D Data 5.1.4.*

The product installable by this release is as follows:

Product Code	**Product Name**	**Version Number**
V07FN459	*AVEVA Gateway for 1D Data 6.0.0*	6.0.0

Introduction

This Release Note about *AVEVA Gateway for 1D Data 6.0.0* describes about the enhancements, fault corrections, prerequisites, and so on.

Content

Prerequisite for this Release - Operating System

The Operating systems prerequisites for the Gateway are listed below:

Prerequisites	**Version No.**
Minimum Supported O/S Version (for each supported platform)	Microsoft Windows Server 2016 (64-bit) Microsoft Windows Server 2019 (64-bit) Microsoft Windows Server 2022 (64-bit)
Mandatory O/S Patches (for each supported platform)	

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

Prerequisite for this Release - Products

The prerequisites for this Gateway release are as follows:

- Windows Server 2016, 2019 or 2022 (64-bit)
- .NET Framework 4.8
- Microsoft Visual C++ Redistributable 2015-2019 (64-bit)
- Access Database Engine (ADE) 2016 (64-bit)

Notes:

- Installing ADE provides the ODBC driver needed by the Gateway to access data sources such as EXCEL files, MS ACCESS, CSV files, SQL server, Oracle and other ODBC-compliant databases.
- Neither the AVEVA Licensing System nor LaaS are required for this version.

Works (is compatible) with

The Gateway works (is compatible) with:

- *AVEVA Asset Information Management (AIM) 5.1.11*
- Microsoft SQL Server (64-bit) 2014, 2016 *
- Oracle database 12c Release 3 (12.1.0.2.0) *

Note: *Connections to these databases are only required if they are being used as inputs or for lookups.

Supersedes

This release supersedes the previous full release 71417 *AVEVA Gateway for 1D Data 5.1.4*.

To Install Product Release

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

Note: Before this AVEVA Gateway for 1D Data 6.0.0 is installed, please ensure that any previous version of AVEVA Gateway for 1D Data is uninstalled.

First released on:

This product release is first available as *AVEVA Gateway for 1D Data 6.0.0*, release number 71461.

List of Enhancements

This release includes the following functional enhancements:

Request Number	Story Number	Description
960119710	513326	1D Data - Add JSON Reporter
	1576462	CSV/TXT output should include information from file input
	1640457	Implement XML Extractor for 1D Data
	2251076	Create AIM bootstrap files
960269658	2352193	Alias creation for scanned documents

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

List of Fault Corrections

The list of fault corrections is as follows:

Incident number	Defect number	Description
960269640	2352124	Project configurations are being overwritten each time the project is opened
960231971	2377137	Bug in AIM-A does not support some characters
	2477274	Additional Engineering objects with ids "N/A" are getting exported to CSV files in case of PDF extraction
960304981	2509375	Multiple instances of same object ID are lost when using Search Criteria: 1D Data

Product Quality Statement

The development, maintenance and testing of AVEVA Solutions’ 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 test environment to verify that enhancements and/or fixes have not adversely affected the integrity of the product.

System Testing

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

Acceptance Testing

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

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

Known Issues

Changing names of output EIWM and CSV files in Base mapping.

Normally, the output filenames have the same value as the input source, but these can be changed in Base Mapping by setting values for the manifest attributes #TARGET_NAME_EIWM# and/or #TARGET_NAME_CSV#. We have detected an issue that when both are set, in which case the output CSV file has the same name as the output EIWM file.

Chapter 2

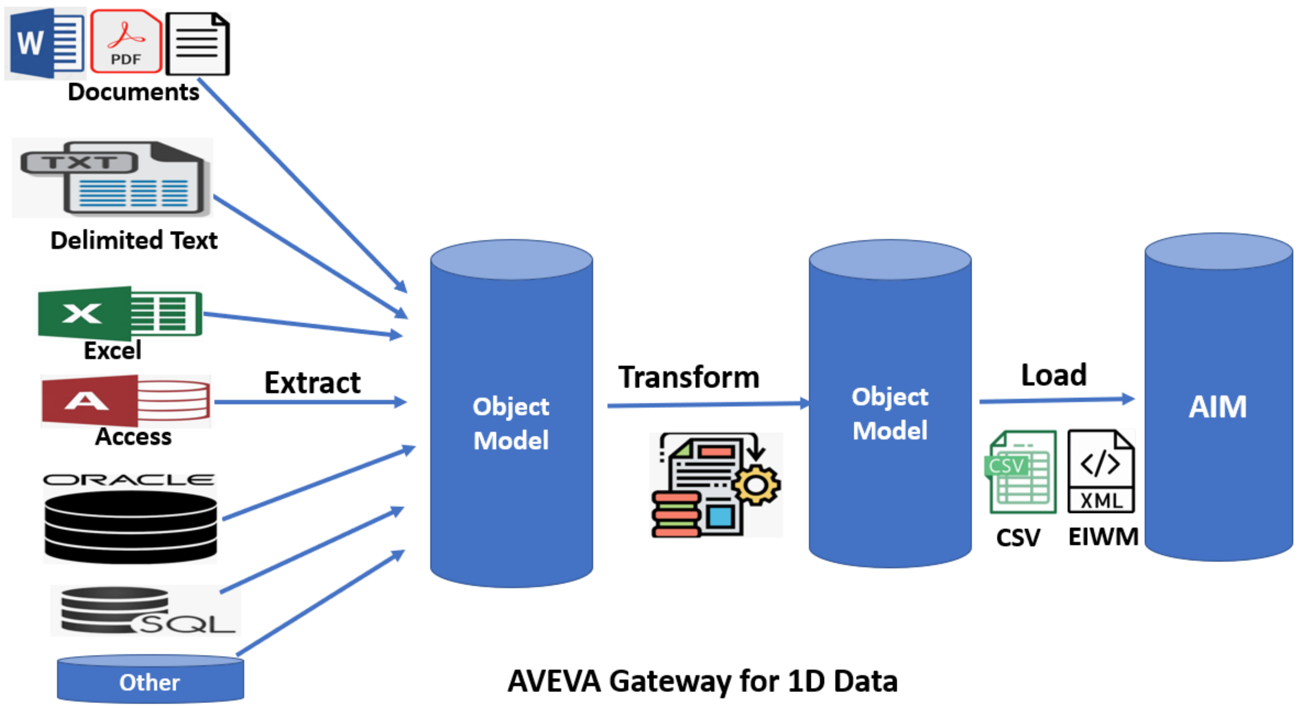
Gateway for 1D Data 6.0.0

The Gateway extracts data from various data sources that are in unstructured (Text, Word or PDF) or tabular (delimited text, Excel, Access) file formats and databases such as SQL Server and Oracle. The Gateway extracts data from specific data sources and transforms them into output EIWM format to be loaded into *AIM*. The output can also be stored on a Windows file system, in an Amazon Web Services (AWS) S3 bucket, or AVEVA Cloud Storage.

Characteristics of the Gateway

The following are the characteristics of the Gateway:

- **Common User Experience**: All new gateways built with common components ensure consistent configurations and operations.
- **Data Validation**: The Gateway validates data using either the whole or part of a value, conforming to a pattern, look up list and Report failures.
- **RegEx Utility**: The Gateway provides a utility to suggest RegEx patterns taken from a tabular Data Source (Excel, Access, SQL, Oracle and Delimiter Text) and provides an option to export into a `.csv` file.
- **Standalone**: As it is a standalone application you can deploy it independently either from its graphical user interface or from a command line interface.



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

Copyright and all other intellectual property rights in this manual and the associated software, and every part of it (including source code, object code, any data contained in it, the manual and any other documentation supplied with it) belongs to, or is validly licensed by, AVEVA Group Limited or its subsidiaries.

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

Trademark

Chapter 1 Get Started

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

Hardware Requirements

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

Component	Minimum	Recommended
Memory	8GB	32GB
Hard Disk Drive	80GB	
CPU	Intel® Xeon® processor, 3GHz, Dual core	Intel® Xeon® processor, 3GHz, Quad core

Usage of RAM

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

Software Requirements

Ensure the following software requirements are fulfilled, before installing the Gateway:

Application	Supported Version
Microsoft .NET Framework	4.8
Windows Server	2016 (64-bit), 2019 (64-bit), 2022 (64-bit)
Access Database Engine (ADE)	2010 (64-bit), 2016 (64-bit) **Note**: When you install ADE, this installs required ODBC drivers for datasources such as EXCEL files, MS ACCESS Database, CSV files, SQL server, and Oracle.
Visual C++ Redistributable Packages	Microsoft Visual C++ Redistributable 2015-2019 (64 bit)

Notes:

- The Gateway supports *AIM* version 5.1.11.
- You need to install ADE if you are using lookups in mapping.
- For Other datasources, ODBC driver for data sources is required.

- AVEVA Licensing System (ALS) and Licensing as a Service (LaaS) are no longer needed.

Access Database Engine

The Gateway uses the Microsoft ADE to read Excel/Access files.

You must install the Microsoft ADE to work on the Gateway. You can download the required version of the Microsoft ADE from:

https://www.microsoft.com/en-us/download

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 ADE (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 Access Database Engine (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 must 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, you can 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, you can restrict the IAMrole to access only S3 bucket. You can also 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, you can configure an SSL-encrypted connection between the Gateway and the database.
- You do not need to adjust your Firewall settings or User Account Control settings when you install or use the Gateway.

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.

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**.

Note: 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.

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 interactively install the Gateway from the Setup Wizard:

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

   Note: If you haven't uninstalled a previous version the installer for that version will be triggered so that you can proceed to uninstall it, note the previous version number in the Setup Wizard screen. Once completed, double-click the `1DDataGatewayInstaller.msi` file for the new version.

2. Select **Next**.
3. If you agree with the AVEVA End User License Agreement terms then select **I Agree**, and then select **Next**.

   Note: To print a copy of the **License Agreement**, select **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. Select **Next**.
7. Select **Install**. The installation process starts. Confirm that you want to make changes to your computer. Select **Finish** after the Setup wizard completes installing the Gateway.

Command Line Installation

To install the Gateway from the command line:

- Type `1DDataGatewayInstaller.msi /?` to see all options.

Example of Quiet installation with passing folder:

```
1DDataGatewayInstaller.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 select **Programs and Features**.
2. Select AVEVA Gateway for 1D 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 `1DDataGatewayInstaller`**.msi** file.
2. Select **Next**.
3. Select **Remove**.
4. To confirm that you want to remove the Gateway, select **Remove** again.
5. Follow the on-screen instructions.

Change and Repair 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 **1DDataGatewayInstaller.msi** file.
2. Select **Next**.
3. Select one of the following:
    a. Select **Change** if you want to modify any optional features installed with the application.
    b. Select **Repair** if you want to repair the errors in the most recent installation, repair any corrupt files, or fix any missing shortcuts or registry entries.
4. Follow the on-screen instructions.

Chapter 2 Run the Gateway from the Project Explorer

The **Project** menu enables you to create new projects and browse 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 **Gateway for 1D Data** in the **Search** box on the taskbar in your computer. Alternatively, you can click the **AVEVA Gateway For 1D Data** icon in the installed location or the shortcut icon on your desktop, if that option was selected when installing the Gateway.

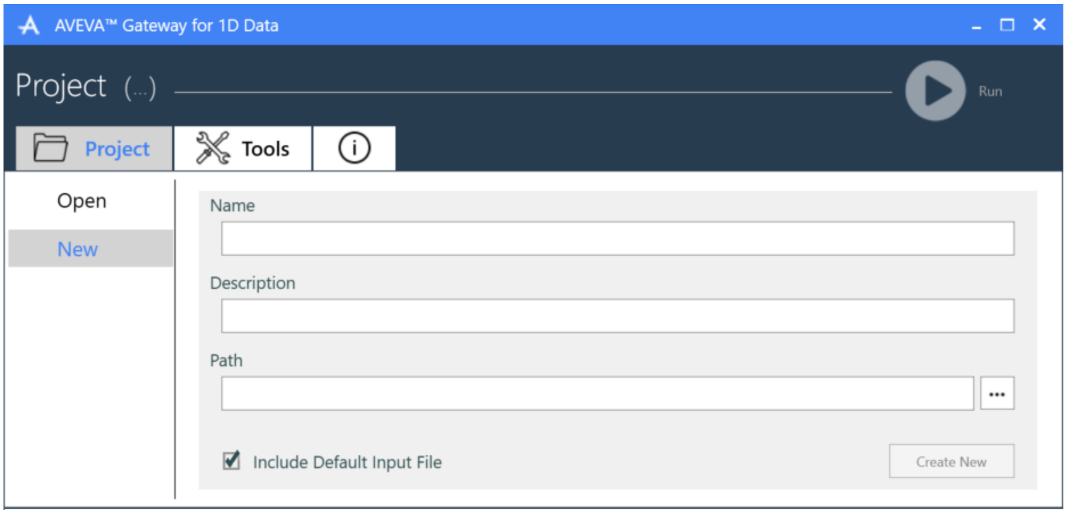
   The Gateway application opens.

2. The **AVEVA Gateway for 1D Data** page opens with the **Project**, **Tools** and **Information** menus in the main window.
3. The Tools tab gives access to the RegEx and Encrypt utilities, see Appendix C: RegEx Utility. and Appendix D: Encrypt Utility.
4. Click the **Information** ⓘ icon to get the information relating to the Gateway version, copyright information and product synopsis.

Create Projects

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

1. On the **Project** tab, select **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.

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.

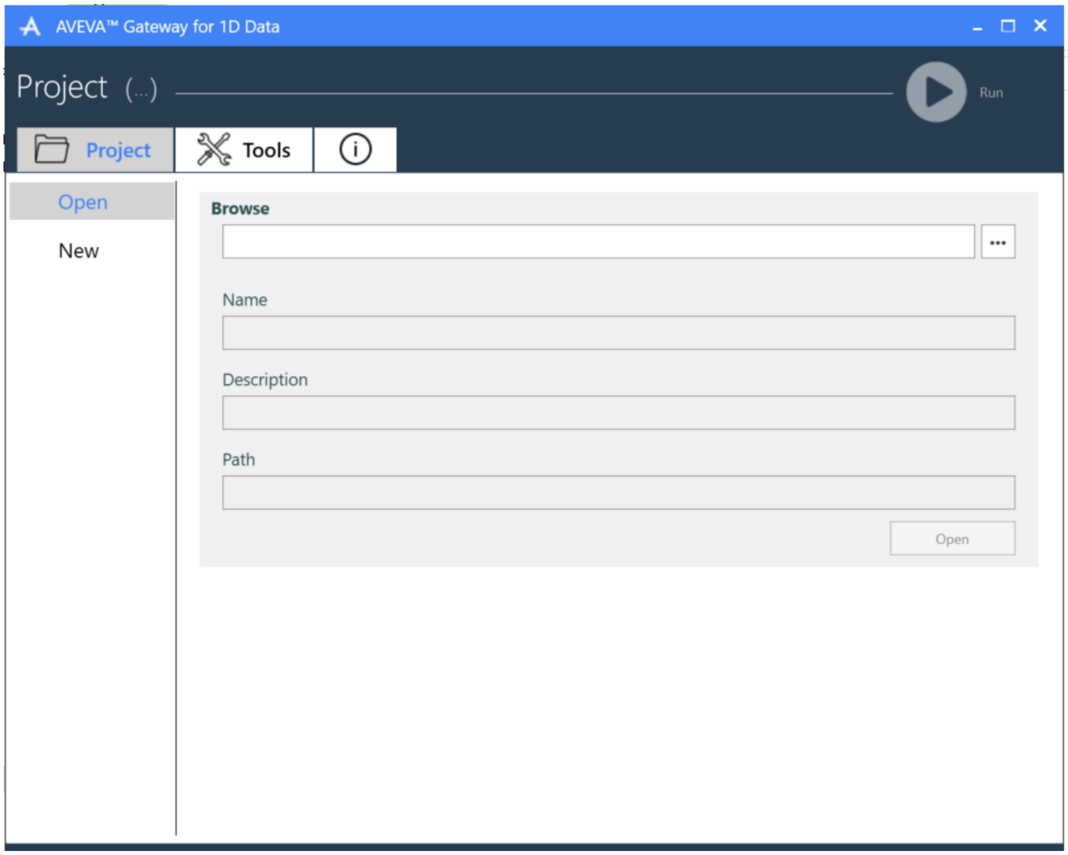
4. Select the **Include Default Input File** box if you want to create some example input files.
5. Click **Create New**.

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

A new Project configuration .xml file with the same Project Name is created. If the folder does not exist, a new project folder is also created. 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.

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 the **Browse** button 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 internally.
4. Click **Open**.

The **Settings** tab is enabled with the contents of `project xml` file.

Notes:

- The existing project configuration files are parsed for any schema validation issues. If any validation issue is detected, the error message is displayed in the Windows status bar on the relevant page affected by the error.
- Users familiar with the *Data Extractor Gateway* can recognize similar functionality with functions in the *Gateway for 1D Data*, but the overlap is not complete and the syntax of the configuration is also different. Therefore, there is no support for upgrading from Data Extractor configurations to *Gateway for 1D Data* project configurations.
- Upgrading from previous versions of the *Gateway for Tabular Data* are supported.

Define the Project Settings

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

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

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

General Settings

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

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

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

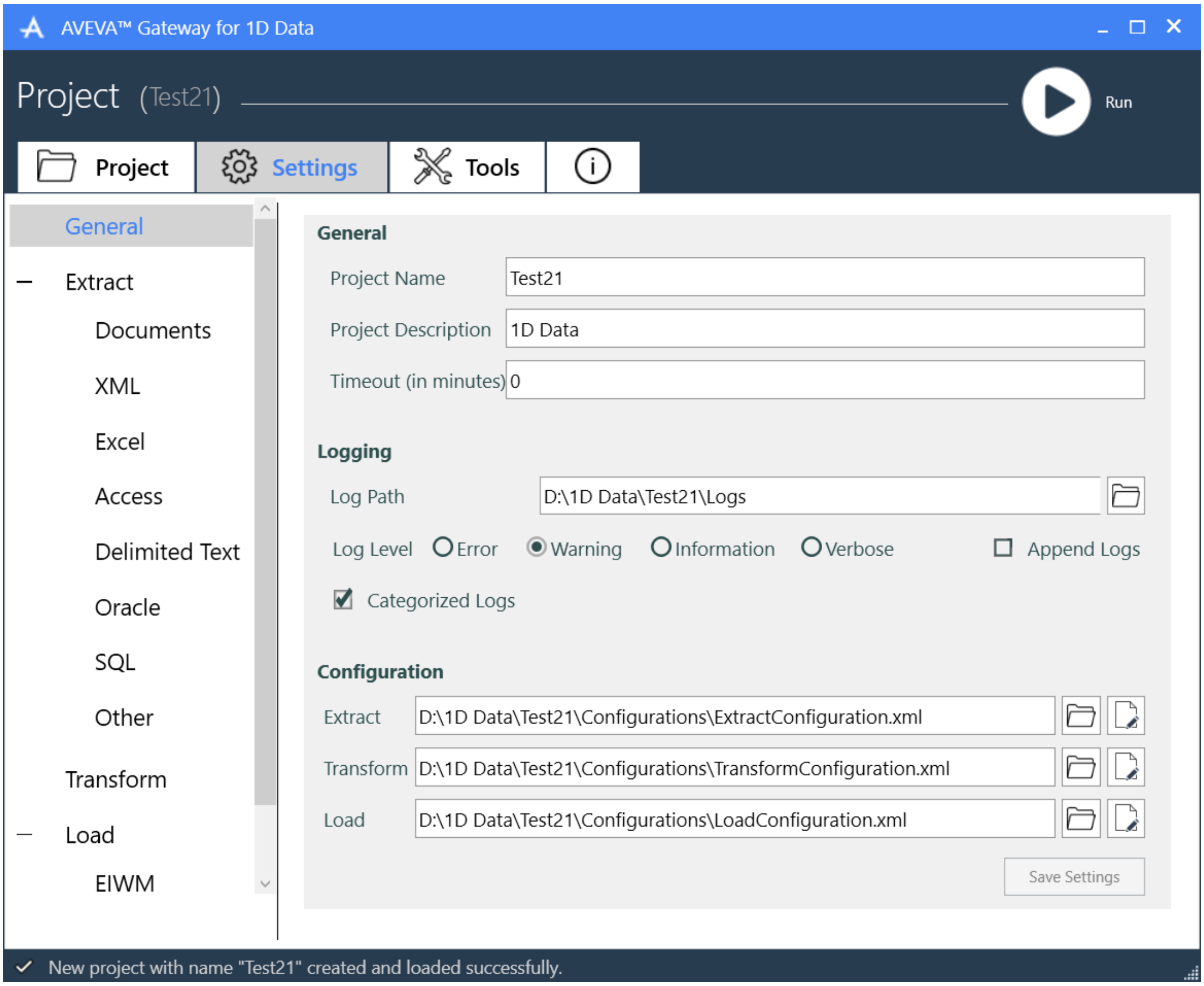
Using Relative Paths in the Configurations

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

Example of Global paths in the Project file:

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

Example of Relative paths in the GUI:



General

Project Name: Test21

Project Description: 1D Data

Timeout (in minutes): 0

Logging

Log Path: D:\1D Data\Test21\Logs

Log Level ○ Error ● Warning ○ Information ○ Verbose □ Append Logs

☑ Categorized Logs

Configuration

Extract: D:\1D Data\Test21\Configurations\ExtractConfiguration.xml

Transform: D:\1D Data\Test21\Configurations\TransformConfiguration.xml

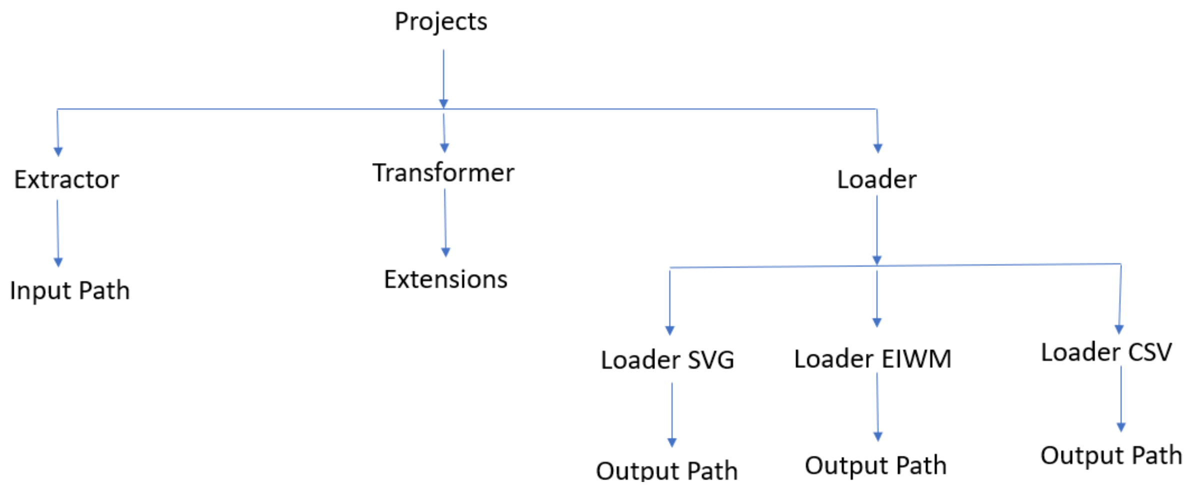
Load: D:\1D Data\Test21\Configurations\LoadConfiguration.xml

✓ New project with name "Test21" created and loaded successfully.

Example of Relative paths in the Project file:

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

Structure of the Project's Configuration and Mapping Files:



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

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

Relative Path is `.\Logs`

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

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

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

General Area

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

To complete the project related information:

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

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

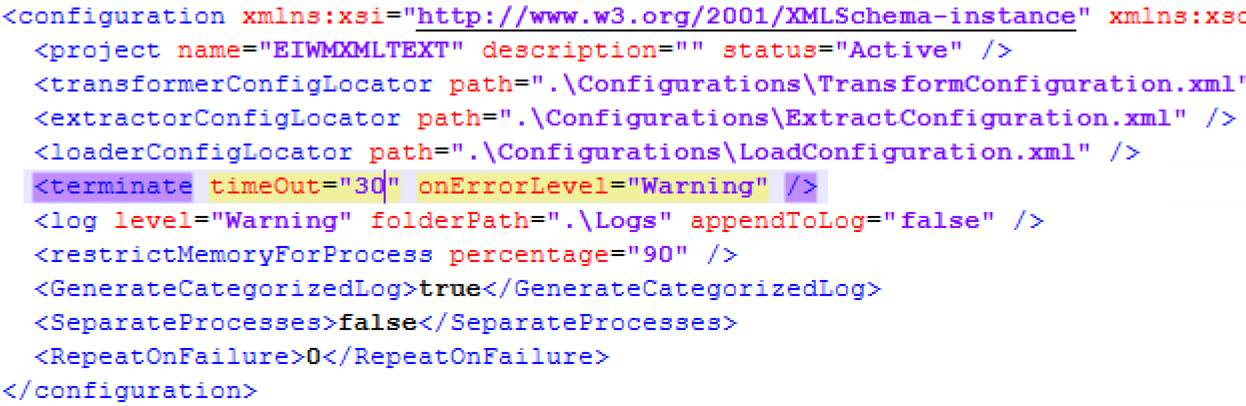
2. Type the time in the **Timeout (in minutes)** box to define the maximum execution time of the project.

   Notes:

   - The **Timeout** value must be a non-negative integer.
   - 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.
   - `OnErrorLevel` parameter is used to set the log level of the timeout log. If set to 0 then no timeout is applied.

     When a user sets the `timeout` (>0, in minutes), the Gateway cancels the execution if the processing time exceeds the set `timeout` value.

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



```
<configuration xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsc
  <project name="EIWMXMLTEXT" description="" status="Active" />
  <transformerConfigLocator path=".\Configurations\TransformConfiguration.xml"
  <extractorConfigLocator path=".\Configurations\ExtractConfiguration.xml" />
  <loaderConfigLocator path=".\Configurations\LoadConfiguration.xml" />
  <terminate timeOut="30" onErrorLevel="Warning" />
  <log level="Warning" folderPath=".\Logs" appendToLog="false" />
  <restrictMemoryForProcess percentage="90" />
  <GenerateCategorizedLog>true</GenerateCategorizedLog>
  <SeparateProcesses>false</SeparateProcesses>
  <RepeatOnFailure>0</RepeatOnFailure>
</configuration>
```

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

Logging Area

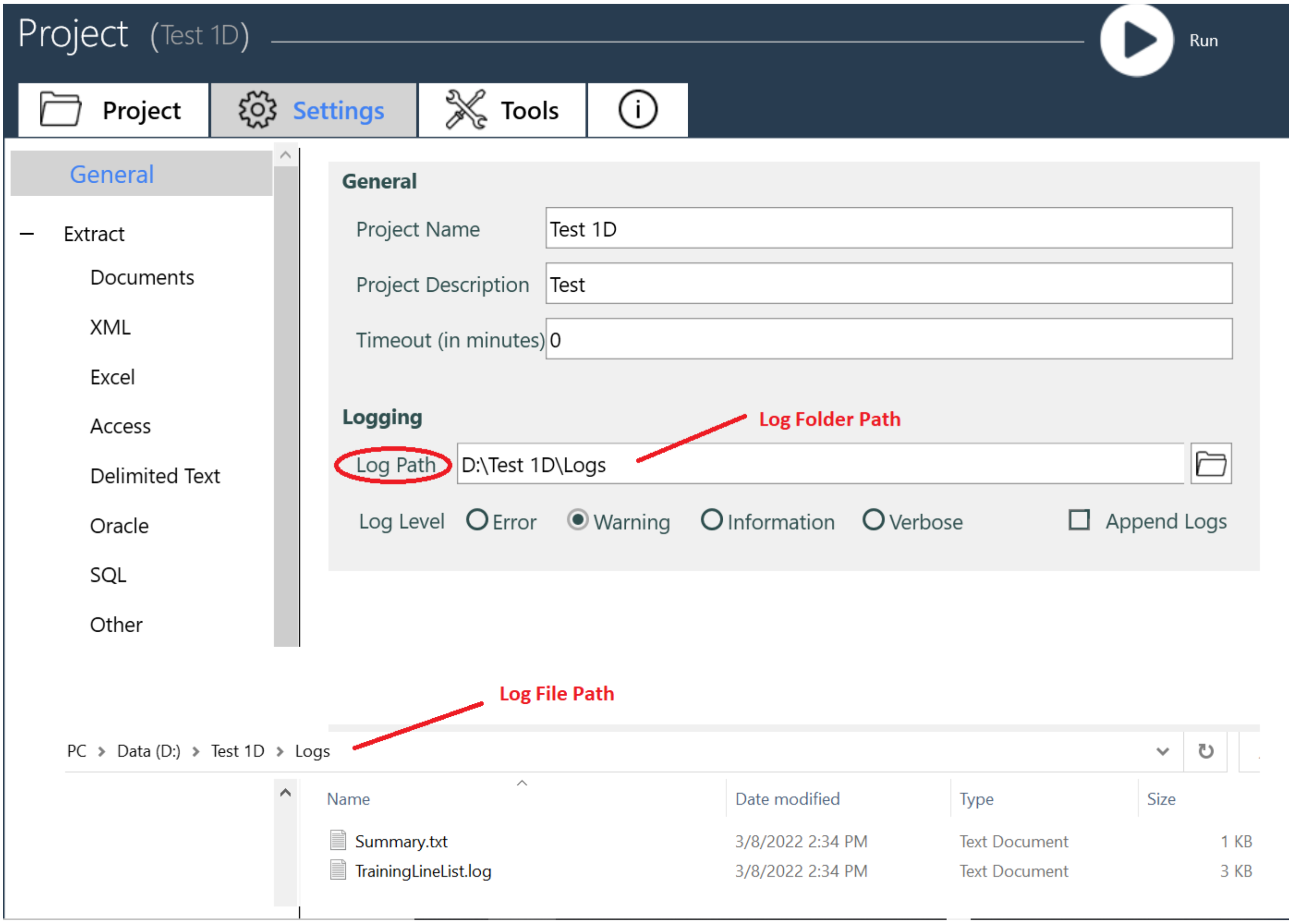
You can use the fields in the **Logging** area to specify different logging information such as warning and error messages. The Gateway produces a log file in the location defined by Log Path. The default location of Log Path is a Logs subfolder under the project file location.

To specify the Logging information:

1. Type the log file path in the **Log Path** box or click the **Browse Path** icon to specify the log file path.

   Note: The Log folder path is the folder where the `inputFile.log file` gets created. Log file path is the path of the `inputFile.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, the Gateway creates the folder automatically. The default name of the log file name is same as the input file name, for example, `<input filename>.log`.



General

Project Name: Test 1D

Project Description: Test

Timeout (in minutes): 0

Logging

Log Path: D:\Test 1D\Logs

Log Level ○ Error ● Warning ○ Information ○ Verbose □ Append Logs

PC > Data (D:) > Test 1D > Logs

Name	Date modified	Type	Size
Summary.txt	3/8/2022 2:34 PM	Text Document	1 KB
TrainingLineList.log	3/8/2022 2:34 PM	Text Document	3 KB

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, time stamp, 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

Log Level	Severity
	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 set to false.

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

   - If you select the **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, when the same input source is processed, the messages are appended to this existing log file.
   - If you deselect the **Append Logs** box, then each run of the Gateway project deletes the old log file if it exists, creates a new log file and adds 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 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.

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 and Load** fields, respectively.

   - **Extract**: Provide the configuration file details to extract the engineering and graphics data from the Tabular file. For new projects, a default configuration file for Extract is created, named as `ExtractConfiguration.xml`.
   - **Transform**: Provide the configuration file details to transform the extracted file. The configuration file for Transform is named as `TransformConfiguration.xml`.
   - **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: 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 📝. To browse to the path of configuration settings, click 📁.

2. Click **Save Settings** to save the **Configuration** field settings.

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

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

Repeat processing on failure

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

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

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

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

Extract Settings

The Gateway can extract data from a variety of input sources located in a local File System or from files stored in 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">`.

Notes:

- The values of the `source` attribute can be a case-insensitive representation of either "`FileSystem`" or "`S3`". The two sources from where the Gateway can extract data are as follows:
  - File-based Sources
  - Database Sources
- Users familiar with the *Data Extractor Gateway* can recognize similar functionality with functions in the *Gateway for 1D Data*, but the overlap is not complete and the syntax of the configuration is also different. Therefore, there is no support for upgrading from Data Extractor configurations to *Gateway for 1D Data* project configurations.
- Upgrading from previous versions of the *Gateway for Tabular Data* are supported.

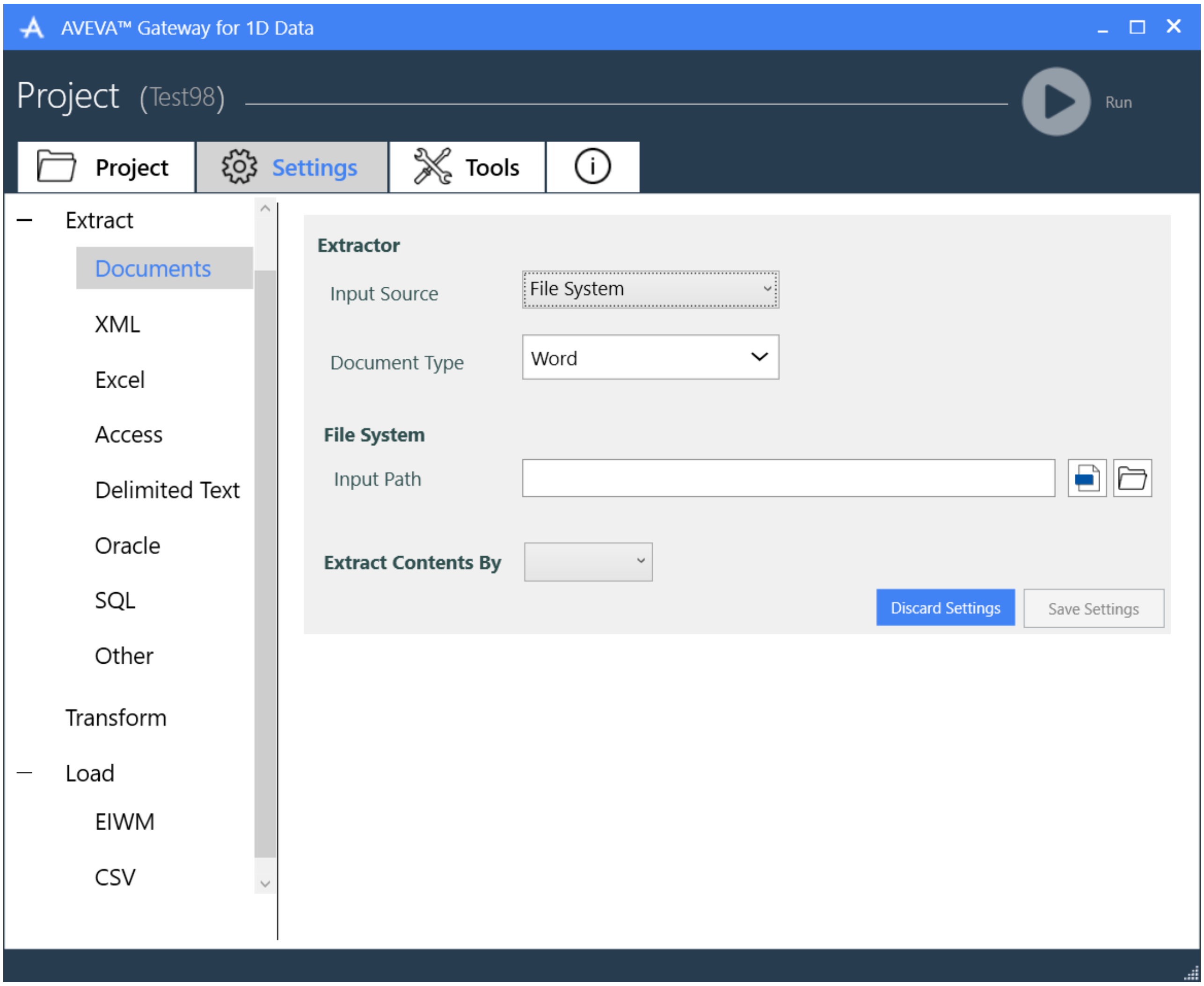
File-Based Sources

The Gateway extracts the data from file-based sources such as Word, PDF, Text, Excel, Access and Delimited Text.

Documents

The Gateway can extract different types of inputs files such as .pdf, .txt and .doc files.

This Extract component can read the data from **File System** or **S3 Bucket** sources.



Extractor

Input Source: File System

Document Type: Word

File System

Input Path:

Extract Contents By:

The details of the input storage to be read by the project are defined in the extract configuration.

To specify how to extract the data:

1. Click **Extract** to display the **Extract** page.
2. Select one or more file formats: **PDF** (.pdf), **Word** (.doc and .docx), and **Text** (.txt) from the **Document Type** drop-down box. You can select either single or multiple document types from the **Document Type** box.
3. Select the type of Input Source from the drop-down box, that is, **File System** or **S3**.

For File System:

1. Define the **Input Path** for the location of the input files, either typing directly or use the **Browse File Path** to select a single file or **Browse Folder Path** to select a folder of files.
2. **Select Extract contents By** option to specify how the text in the input file(s) will be stored in the Object Model for subsequent transformation, either as one word per object, one line per object, or treat the whole text as a single object:
   - Word
   - Line
   - Whole Text. By default, Whole Text is selected.

Note: If you select a folder of documents to process, then the Gateway generates output EIWM files, corresponding to each of the input files in the folder or in subfolders. These EIWM files will have the same names as the input file names.

Extracting Word/Line/Whole Text Positions During PDF File Extraction

When you extract the **PDF** file, you can also extract the X and Y positions of Word/line or whole text based on your selection. These positions are stored as X and Y extents attributes of each object in the Object Model:

- `#Xmin#`
- `#Ymin#`
- `#Xmax#`
- `#Ymax#`

These attributes will be represented in the output EIWM as characteristics of the text object, for example:

Use iFilter to Extract Embedded Excel Spreadsheet Contents from Word or PDF Files:

To extract the contents of an embedded Excel spreadsheet in a Word or PDF file, you need to install the relevant iFilter (for Word or PDF files) that is compatible with your server's operating system and set the attribute value `IFilter apply="true"` in the Extract configuration XML file. The default value is "`false`" as this allows for single words or lines of text to be searched and the X and Y values of the tag locations to be recorded, neither of which are available when using the iFilter mode. The Type attribute sets the specific document type to process using iFilter when the apply attribute is set to true. The document type can only be of Word and PDF type.

```
    <DocumentFile Type="Word,PDF,Text">
      <Input source="S3">
        <S3>
          <Authentication instance="false">
            <CredentialFile path="C:\Users\user_name\.aws\credentials" profileName="3pg-
dev" />
          </Authentication>
          <Region>eu-west-1</Region>
          <BucketName>1ddatainput</BucketName>
          <ObjectKey />
        </S3>
      </Input>
      <ExtractBy>Word</ExtractBy>
      <IFilter apply="false" Type="PDF,Word" />
    </DocumentFile>
  </Datasources>
</configuration>
```

Limitations of iFilter

- If a PDF file contains tables, then it will not combine columns with space. This will lead to no spaces among the table content.
- If a PDF file contains multiple pages, then it will not provide any differentiation among pages.
- Extracted text from a PDF file will be represented as a single object so you can only extract contents by "Whole Text".
- The X and Y positions of the text cannot be determined so will not be available as X Min and Max and Y Min and Max values in the EIWM objects.

For S3

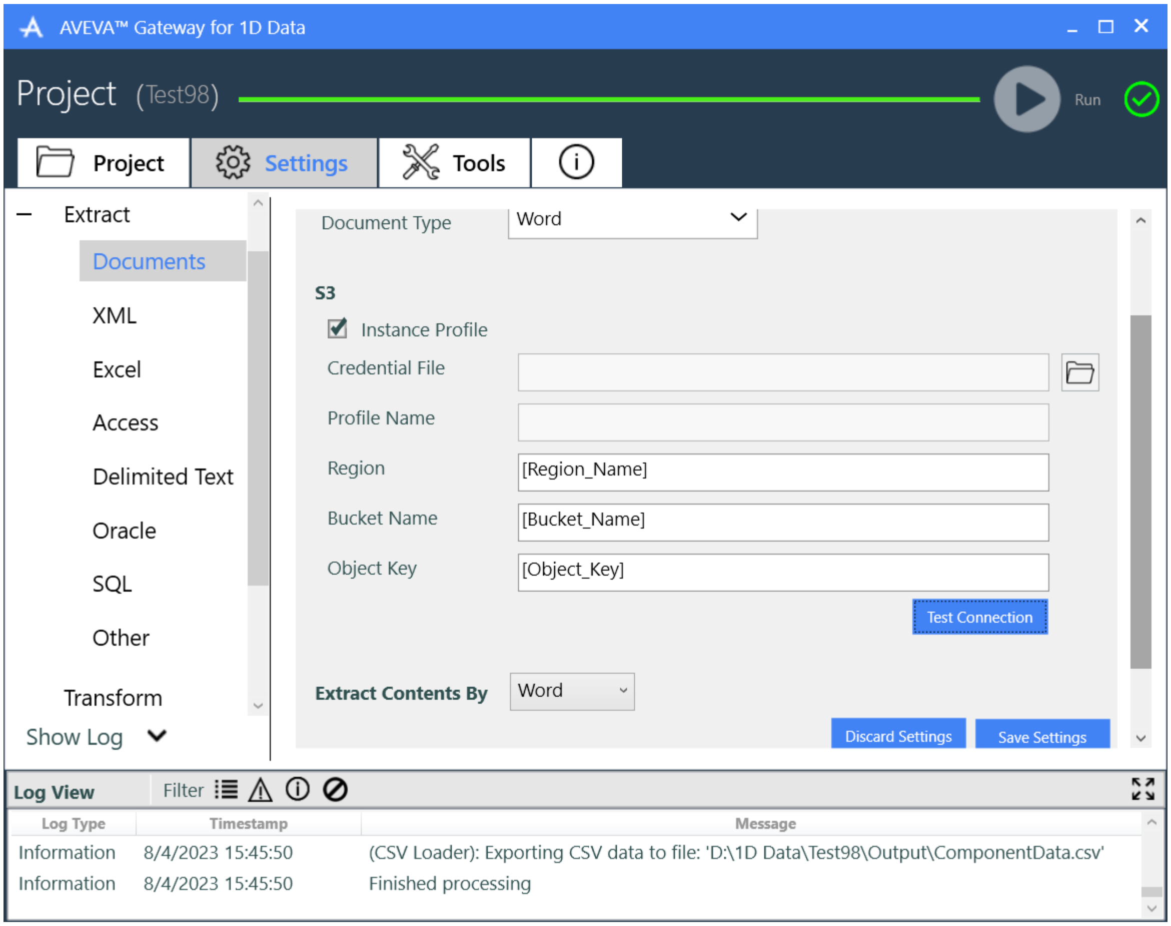
An S3 Bucket is a container for objects stored in Amazon Web Services (AWS) S3 Bucket container.

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

1. Follow the below procedures for Accessing an AWS S3 Bucket () to use the **S3 Credential** Details. If you want to test the connection to the S3 Bucket, click **Test Connection**. The result of the test is displayed in the status bar.
2. You can select either single document or multiple documents in the **Document Type** for an S3 Bucket Object Key:

   Single Document Selection: S3/Input folder or S3/Input/sampleword.doc file name provides the file that matches with the document type.

   Multiple Documents Selection: S3/Input folder is allowed but no object key with extension is allowed.



Document Type: Word

S3

☑ Instance Profile

Credential File:

Profile Name:

Region: [Region_Name]

Bucket Name: [Bucket_Name]

Object Key: [Object_Key]

Extract Contents By: Word

Log Type	Timestamp	Message
Information	8/4/2023 15:45:50	(CSV Loader): Exporting CSV data to file: 'D:\1D Data\Test98\Output\ComponentData.csv'
Information	8/4/2023 15:45:50	Finished processing

To configure the extractor for S3 Bucket as input source:

1. Define the elements in **S3 Credential Details** section.
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**.

Configuration

```
<configuration xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://
www.w3.org/2001/XMLSchema-instance" sourceProductName="AVEVA Gateway for 1D Data"
componentName="Extract" componentVersion="2.8.0.0" >
  <Datasources Type="Word,PDF,Text">
    <Input source="FileSystem">
      <S3>
        <Authentication instanceProfile="false">
          <CredentialFile path=" " profile=" " />
        </Authentication>
        <Region> </Region>
        <BucketName> </BucketName>
        <ObjectKey />
      </S3>
      <FileSystem>
        <InputPath>C:\Users\ <Gateway User> \Desktop \review \Input \</InputPath>
```

```
            </FileSystem>
          </Input>
          <ExtractBy>Line</ExtractBy>
        </Datasources>
      </configuration>
```

The Gateway extracts the configuration file containing a specific datasource from S3 repository.

XML

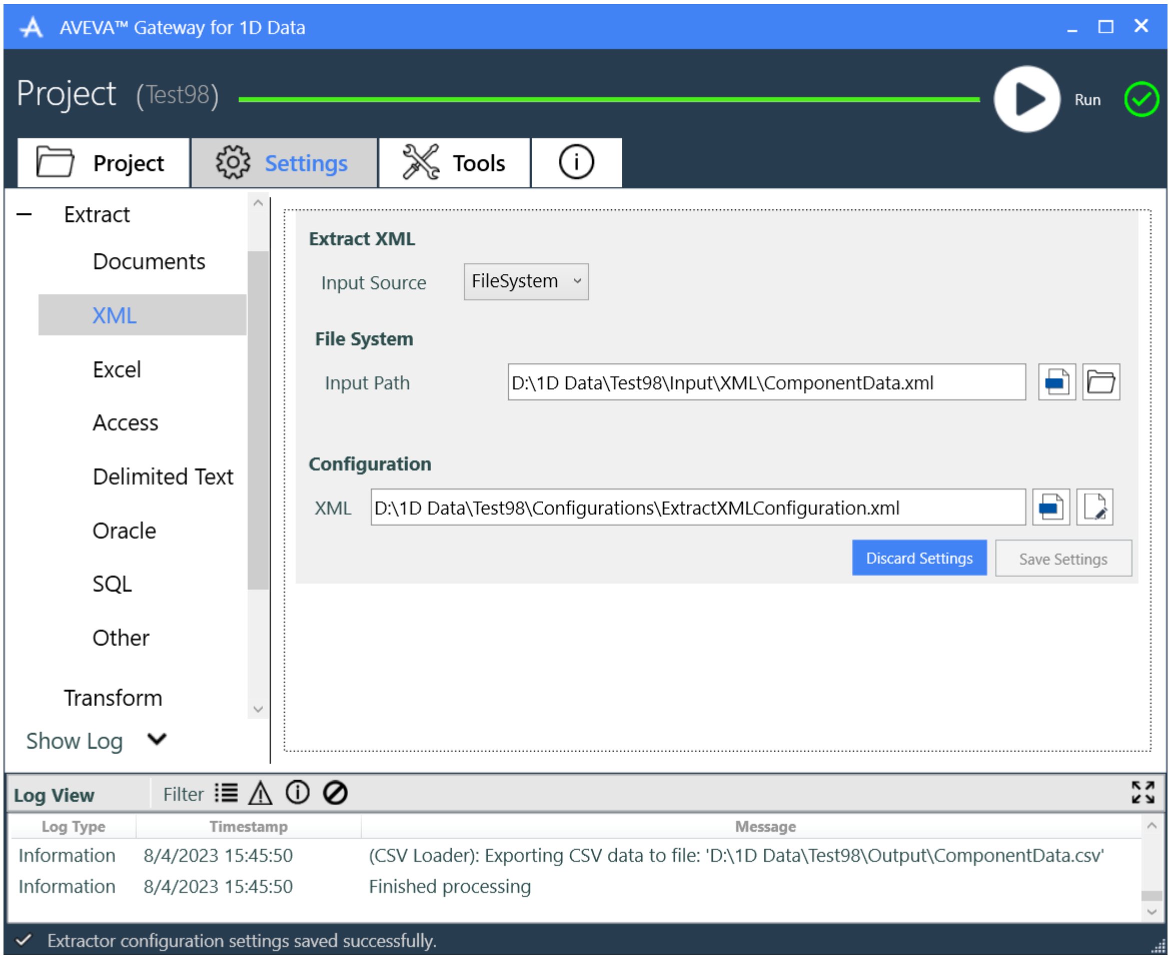
You can extract the XML-specific data (.xml) by defining specific settings for XML data:

To extract the specific settings for XML data:

1. Click **XML**.
2. Click the **Input Source** drop-down list.
3. Select the type of **Input Source** from the drop-down box, that is, **File System** or **S3**.

Configuring File System and S3

For FileSystem



Extract XML

Input Source: FileSystem

File System

Input Path: D:\1D Data\Test98\Input\XML\ComponentData.xml

Configuration

XML: D:\1D Data\Test98\Configurations\ExtractXMLConfiguration.xml

Log Type	Timestamp	Message
Information	8/4/2023 15:45:50	(CSV Loader): Exporting CSV data to file: 'D:\1D Data\Test98\Output\ComponentData.csv'
Information	8/4/2023 15:45:50	Finished processing

✓ Extractor configuration settings saved successfully.

1. Define the **Input Path** for the location of the relevant input files, either typing directly or use the **Browse File Path** to select the user input file in the correct format, that is, `.xml` or select **Browse Folder Path** to select a folder of files.
2. Browse and select the user input file in the correct format, that is, `.xml`.

   A separate XML configuration file `ExtractXMLConfiguration.xml` is created automatically in the configurations folder.
3. Edit the XML configuration file and make necessary configuration changes.
4. Save the configuration.

   The Extractor configuration is saved to an XML file with a data source section, for example:

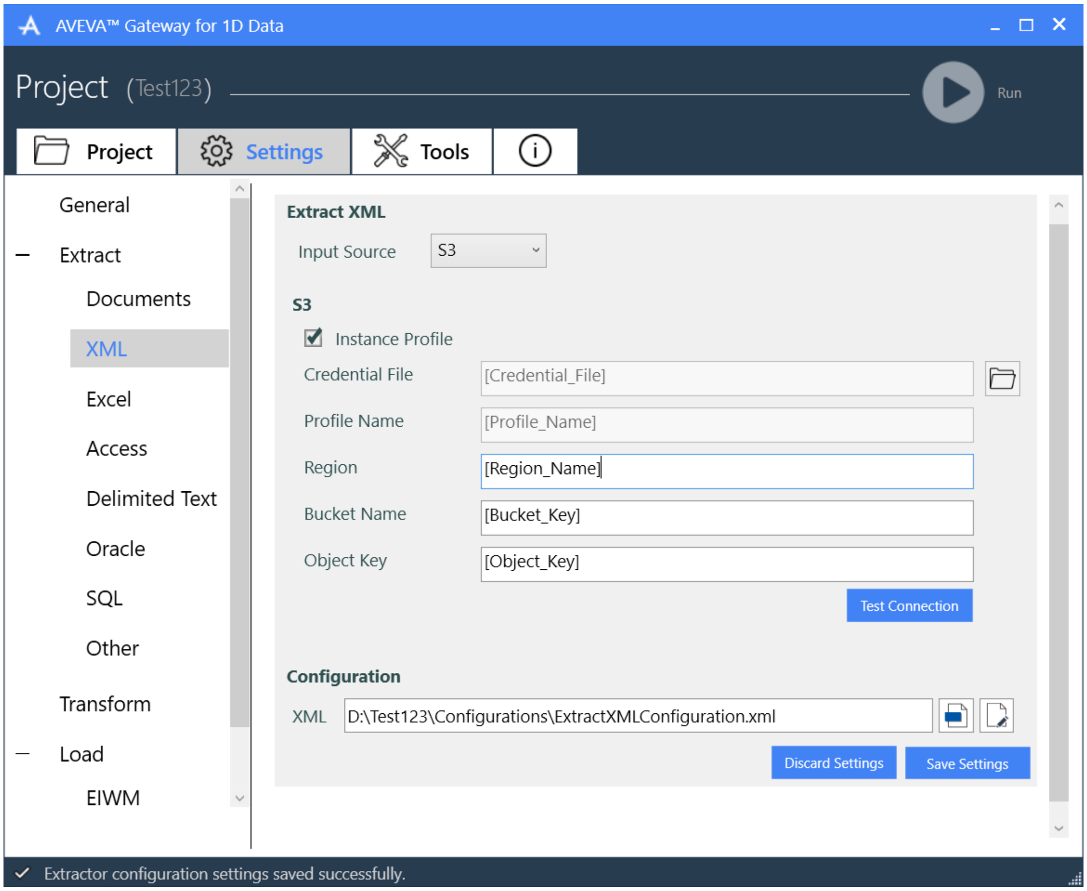
```
<Datasources>
    <XML>
      <Input source="FileSystem">
        <FileSystem>
          <InputPath>C:\Users\User_name\Desktop\1D Data
Projects\testing\Input\XML\ComponentData.xml</InputPath>
        </FileSystem>
      </Input>
      <ExtractXMLConfigurationPath>C:\Users\ User_name\Desktop\1D Data
Projects\testing\Configurations\ExtractXMLConfiguration.xml</
ExtractXMLConfigurationPath>
    </XML>
</Datasources>
```

5. Click **Run** to process all the files relevant to the extraction settings in the selected folder and its sub-folders, based on the extractor configuration that has been saved.

For S3

To configure the extractor for S3 bucket as input source:

1. Follow the below procedures for Accessing an AWS S3 Bucket () to use the S3 bucket Details.



Extract XML

Input Source: S3

S3

☑ Instance Profile

Credential File: [Credential_File]

Profile Name: [Profile_Name]

Region: [Region_Name]

Bucket Name: [Bucket_Key]

Object Key: [Object_Key]

Configuration

XML: D:\Test123\Configurations\ExtractXMLConfiguration.xml

✓ Extractor configuration settings saved successfully.

2. Define the elements in S3 section.
3. Click **Test Connection,** if you want to test the connection to the S3 bucket. The result of the test is displayed in the status bar.
4. Click Save Settings.

XML Configuration Example with S3 Input Source:

```
<Datasources>
   <XML>
    <Input source="S3">
      <S3>
        <Authentication instanceProfile="false">
          <CredentialFile path=" " profileName =" " />
        </Authentication>
        <Region> </Region>
        <BucketName> </BucketName>
        <ObjectKey />
      </S3>
      <FileSystem>
        <InputPath>C:\Users\<Gateway User>\Desktop\review\Input\</InputPath>
      </FileSystem>
    </Input>
    <ExtractXMLConfigurationPath> C:\Users\User_Name\Desktop\SubfolderTestProject\
Configurations\ExtractXMLConfiguration.xml </ExtractXMLConfigurationPath>
```

```
     </XML>
   </Datasources>
```

The Gateway extracts the configuration file containing a specific datasource from S3 repository.

ExtractXMLConfiguration.xml File:

The **ExtractXMLConfiguration** file reads the Xpaths provided in each of the `<objectsPath>` nodes, navigates to the particular node in the input XML based on the hierarchy provided in the Xpath and extracts the Internal nodes/Attributes available in that particular selected node.

If you want to extract all the internal nodes of selected node as attributes to the output EIWM file, the following mapping can be used:

```
<Attributes>
   <Attribute value="/*" />
</Attributes>
```

If you want to extract all the attributes of selected node as attributes to the output EIWM file, the following mapping can be used:

```
<Attributes>
   <Attribute value="/@*" />
</Attributes>
```

If you want to extract selected attributes/internal nodes of selected node as attributes to the output EIWM file, below mapping can be used:

```
<Attributes>
   <Attribute value="/@Attribute name" />
          <Attribute value="/Internal node name" />
</Attributes>
```

If you want to extract the internal nodes as separate objects with their own attributes, hierarchical mapping can be provided as mentioned below:

```
   <Object>
      <objectsPath value="/Plant/Equipments/Equipment" />
      <Attributes>
      <Attribute value="/@EquipmentID" />
        <Attribute value="/*" />
      </Attributes>
     <Object>
   <objectsPath value="/ComponentsPurchased/Component" />
        <Attributes>
         <Attribute value="/*" />
        </Attributes>
     </Object>
    </Object>
```

An additional association "is a part of" is added to the object created from Xpath "`/ComponentsPurchased/Component`" to the parent object created from Xpath "`/Plant/Equipments/Equipment`" in the output EIWM.Configuring XML Extraction

Configuring XML Extraction

A sample `ExtractXMLConfiguration.xml` is provided below, for processing the example XML file `ComponentData.xml`:

```
   <XmlSettings>
     <Objects>
       <Object>
```

```
      <objectsPath value="/Plant/Equipments/Equipment" />
      <Attributes>
      <Attribute value="/@EquipmentID" />
        <Attribute value="/*" />
      </Attributes>
     <Object>
   <objectsPath value="/ComponentsPurchased/Component" />
        <Attributes>
         <Attribute value="/*" />
        </Attributes>
     </Object>
    </Object>
   <Object>
      <objectsPath value="/Plant/Instruments/Instrument" />
      <Attributes>
        <Attribute value="/@*" />
      </Attributes>
    </Object>
  </Objects>
</XmlSettings>
```

The previous `ExtractXMLConfiguration` example creates attributes in the object model that can then be used in the Transformation base mapping, for example,

Sample Base Mapping Configuration

```
<MappingTemplate componentVersion="2.8.0.0" sourceProductName="AVEVA™ Gateway for 1D
Data" componentName="BaseMapping"
  <ObjectMappings regExTimeoutSecs="10">
    <Object>
      <Conditions>
        <Attribute name="#TYPE#" pattern="^manifest$" />
      </Conditions>
      <ObjectID value="[#MODEL_NAME#]" />
    </Object>
    <Object>
      <Conditions>
        <Attribute name="EquipmentID" pattern=".*" />
      </Conditions>
      <ObjectID value="[EquipmentID]" />
      <ClassID value="[Name]" />
    </Object>
    <Object>
      <Conditions>
        <Attribute name="ComponentID" pattern=".*" />
      </Conditions>
      <ObjectID value="[ComponentID]" />
      <ClassID value="[ComponentName]" />
    </Object>
    <Object>
      <Conditions>
        <Attribute name="InstrumentID" pattern=".*" />
     </Conditions>
      <ObjectID value="[InstrumentID]" />
      <ClassID value="[InstrumentName]" />
```

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

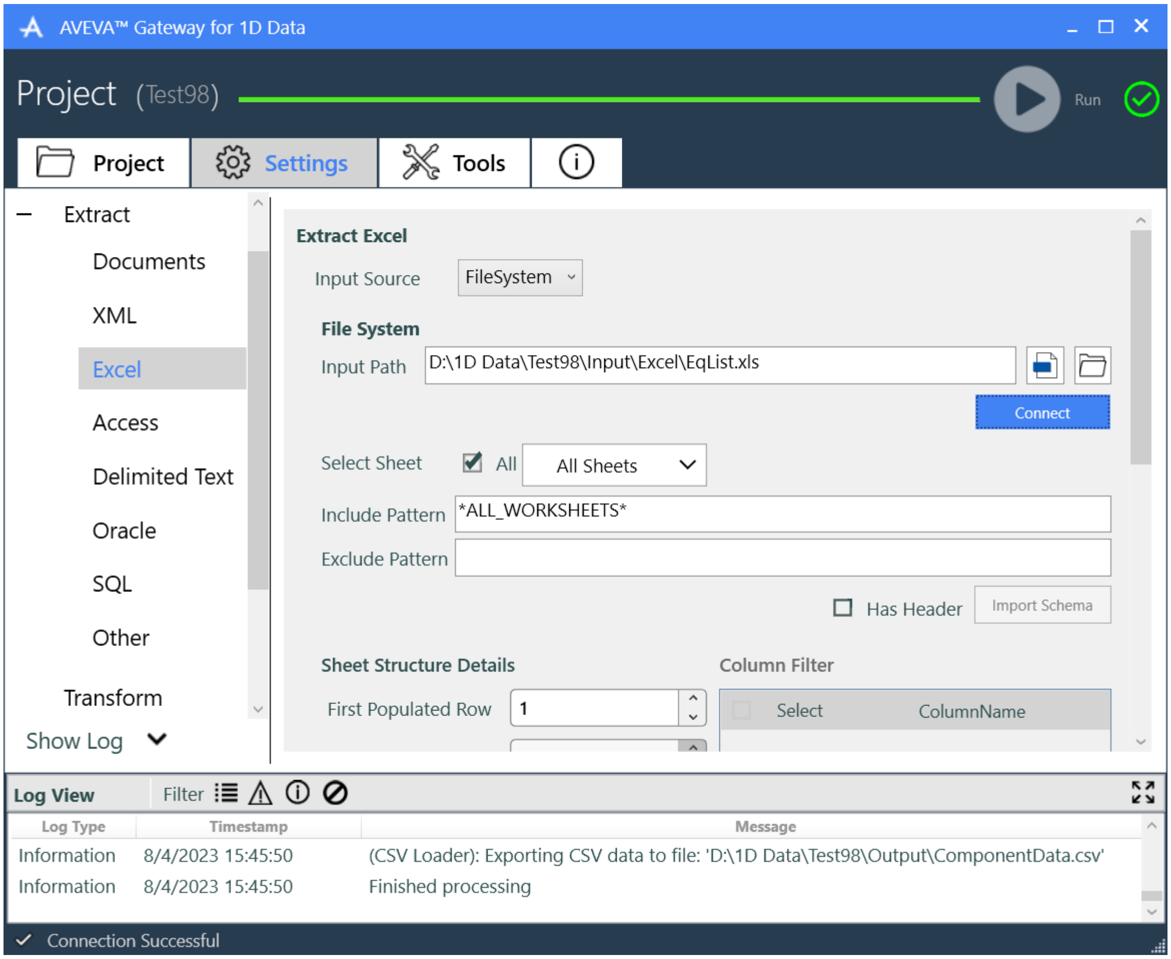
Excel

You can extract the Excel-specific data (`.xls` and `.xlsx`) from the Gateway by defining specific settings.

To extract the specific settings for Excel data:

1. Click Excel.
2. Click the **Input Source** drop-down list.

   You can select either **FileSystem** or **S3 Bucket** from the list.



Extract Excel

Input Source: FileSystem

File System

Input Path: D:\1D Data\Test98\Input\Excel\EqList.xls

Select Sheet: ☑ All — All Sheets

Include Pattern: *ALL_WORKSHEETS*

Exclude Pattern:

□ Has Header

Sheet Structure Details

First Populated Row: 1

```
Log Type      Timestamp           Message
Information   8/4/2023 15:45:50   (CSV Loader): Exporting CSV data to file: 'D:\1D Data\Test98\Output\ComponentData.csv'
Information   8/4/2023 15:45:50   Finished processing
```

✓ Connection Successful

For FileSystem Input Source

- **Input Path**: To process all relevant files in **File System**:
  a. Browse and select the user input file in the correct format, that is, `.xls` or `.xlsx`.
  b. Select an input file and connect to it, then select the sheet and import, enter Sheet Structure Details (see the Requirements for the Thermo Library Generation Tool section).
  c. Select the relevant columns to extract.

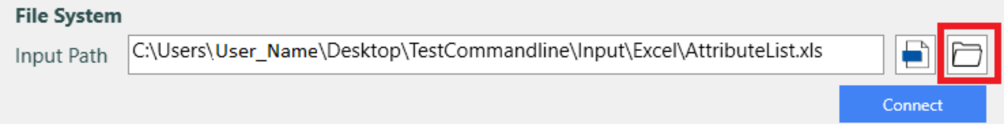
d. Generate the equivalent query and save the configuration.

The Extractor configuration is saved to an XML file with a data source section, for example:

```
<Datasources>
    <Excel hasHeader="false">
      <Input source="FileSystem">
        <FileSystem>

<InputPath>C:\Users\User_Name\Desktop\SubfolderTestProject\Input\Excel\AttributeList.xls
</InputPath>
        </FileSystem>
      </Input>
      <Sheet>Sheet1</Sheet>
      <Query />
      <Details firstPopulatedRow="1" headingsRow="0" firstDataRow="1"
lastDataRow="-1" />
    </Excel>
</Datasources>
```

If you want to apply the same configuration for more than one file in a folder, then you must first select one of the files, set the extract configuration as above and save the settings.

- Click the input folder icon and enter the path where the input files are present.



File System

Input Path C:\Users\User_Name\Desktop\TestCommandline\Input\Excel\AttributeList.xls

In this folder mode, you cannot modify the Extract setting for columns as this is driven by a specific file's data. However, you can still modify the query. Save the settings for the selected folder.

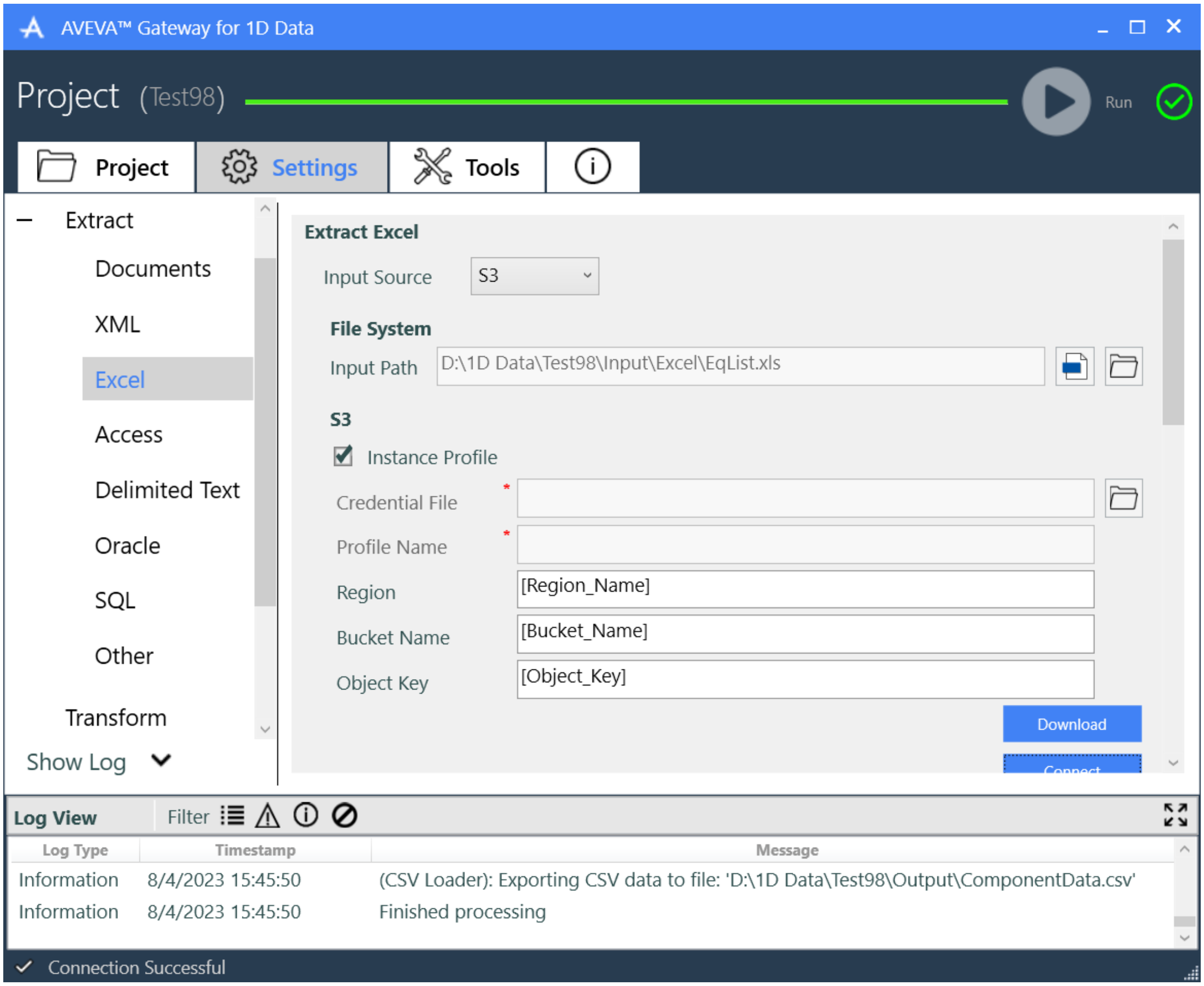
```
<Datasources>
    <Excel hasHeader="false">
      <Input source="FileSystem">
        <FileSystem>
          <InputPath>C:\Users\<user_name>\Desktop\SubfolderTestProject\Input\Excel </
InputPath>
        </FileSystem>
      </Input>
      <Sheet>Sheet1</Sheet>
      <Query />
      <Details firstPopulatedRow="1" headingsRow="0" firstDataRow="1"
lastDataRow="-1" />
    </Excel>
</Datasources>
```

If you enable the **Incremental Scan** while processing a folder, you must select the **Datetime Column** when configuring for a single file. You cannot edit the selected file. The **Last Scanned On** date can be updated but it will be applied to all of the files in the folder. See the Requirements for the Thermo Library Generation Tool section.

- Click **Run** to process all the files relevant to the extraction settings in the selected folder and its sub-folders, based on the extractor configuration that has been saved.

For S3 Input Source

- **S3 Details:** See Accessing an AWS S3 Bucket () for more information.



Extract Excel

Input Source: S3

File System

Input Path: D:\1D Data\Test98\Input\Excel\EqList.xls

S3

☑ Instance Profile

Credential File *

Profile Name *

Region: [Region_Name]

Bucket Name: [Bucket_Name]

Object Key: [Object_Key]

Log Type	Timestamp	Message
Information	8/4/2023 15:45:50	(CSV Loader): Exporting CSV data to file: 'D:\1D Data\Test98\Output\ComponentData.csv'
Information	8/4/2023 15:45:50	Finished processing

✓ Connection Successful

To process all the relevant files in an S3 bucket:

1. Fill all the mandatory fields with the required information to access the bucket, but you must not set a value for the Object Key.

   The first file in the bucket will be downloaded into a temporary location displayed in the **Input Path**.

2. Click **Connect** to start configuring the file format parameters such as **Sheet**, **Sheet Structure Details** and **Column Filter** (see the Requirements for the Thermo Library Generation Tool section).

3. Save this configuration as it will be applied to each of the files in the bucket after you click the **Run** button.

Note: Each download from an S3 bucket is into a new temporary folder to avoid processing previously downloaded files. Hence, its location is not saved in the configuration.

```
<Datasources>
    <Excel hasHeader="false">
      <Input source="S3">
        <S3>
          <Authentication instance="false">
            <CredentialFile path="C:\Users\User_name\.aws\credentials" profileName="3pg-
dev" />
```

```
          </Authentication>
          <Region>eu-west-1</Region>
          <BucketName>1ddatainput</BucketName>
          <ObjectKey>Input/Excel/EqList.xls</ObjectKey>
        </S3>
        <FileSystem>
          <InputPath>C:\Users\User_name\Desktop\test\Input\Excel\AttributeList.xls</
InputPath>
        </FileSystem>
      </Input>
      <Sheet includePattern="*ALL_WORKSHEETS*" excludePattern="" />
      <Query />
      <Details firstPopulatedRow="1" headingsRow="0" firstDataRow="1" lastDataRow="-1" />
    </Excel>
 </Datasources>
```

- **Download**: Click **Download** to validate the AWS Credentials and download the specified file from the S3 bucket to the input location of the Project folder. **File System Details** will be filled with the downloaded location and file details.

Select Sheet

This section is common for both File System and S3 input file sources.

You can process either single or multiple worksheets using a single extract configuration using this feature.

Using Pattern mapping option, you can include/exclude patterns from the **Extract** section. The Extractor uses these patterns to select sheets from the available set of sheets in the excel file. This subset of sheets is then used to further extract data. For more information about processing multiple worksheets, see Requirements for the Thermo Library Generation Tool.

Sheet Structure Details

This section is common for both File System and S3 input file sources.

Define the following elements of the Source data structure:

- Has Header: Select this option if the Excel file contains a header.
- **Connect**: Click this button to open the Excel file. After this connection is established, the **Select Sheet** drop-down box is filled with the sheet name and Import Schema box is also populated. If that input is new and is not saved in the configuration file, then by default the first sheet is displayed. Otherwise, it populates the last selected sheet.
- **Select Sheet**: Select the relevant sheet. Only one sheet can be extracted at a time. Click Import Schema to fetch the list of rows in that sheet. These rows are displayed in the Column Filter box, which you can select to generate a query. The sheet structures are described in the following table:

Settings	**Description**
First Populated Row	First row in the worksheet having any type of content.
Heading Row	The row that contains the column headings. If omitted, it is assumed that there is no **Heading Row**. If heading row is set to "0", it is assumed that there is no heading row.

Settings	**Description**
First Data Row	The row that contains the first of the data rows. If omitted, it is assumed that the first data row is immediately after the **'HeadingRow'** or, if that is also omitted, the first row in the file.
Last Data Row	The last row that contains a valid data row. If the last data row is set to "-1", it is assumed that all rows need to be processed.

ODBC ignores the rows which are empty, if those are present before any data present in the Excel file. Row indices of the table that are extracted by ODBC from the Excel file may not match with the row indices of the Excel file. So the **First Populated Row** is provided to determine the correct Heading Row, **First Data Row** and **Last Data Row** indices.

These sheet structure settings help to extract the data correctly when:

- The first populated row in the Excel worksheet does not contain headings.
- Empty rows are between heading row and first data row.
- Text below the last data row is not considered as data.

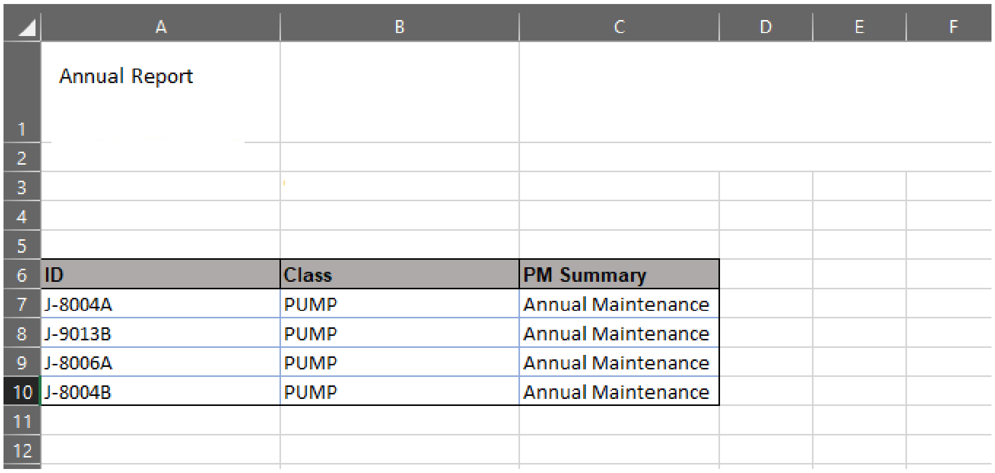
Note: You cannot process an Excel spreadsheet through the Gateway if its name contains one or more single quotation marks. For example, a sheet name `Pump's Detail` is invalid as per the ODBC standards. However, you can still process spreadsheets with other special characters in their name.

- **Column Filter**: Specify the column names and provides the filter to select particular column data for processing. Click **Select** to select all the columns.
- **Query Generator**: Type the query in this box.
  - **Generate Query**: Click this box to generate the query to fetch the data. Query is based on your selection criteria.
  - **Edit Query**: Click this box to edit the query.

Example of First Populated Row:

Scenario 1:

The XLS input file has a structure where general content is provided in the first row without any tabular data content.



	A	B	C	D	E	F
1	Annual Report					
2						
3						
4						
5						
6	ID	Class	PM Summary			
7	J-8004A	PUMP	Annual Maintenance			
8	J-9013B	PUMP	Annual Maintenance			
9	J-8006A	PUMP	Annual Maintenance			
10	J-8004B	PUMP	Annual Maintenance			
11						
12						

- When you click **Import Schema**, then the Heading rows (`ID`, `Class`, `PM Summary`) will not get extracted as expected. Because ODBC searches only the Data Row before the Heading Row. Hence, **Import Schema** option, which helps extract the schema of the sheet, will extract columns as F1, F2, F3 in the **Column Filter**. Hence, to process such files, you must select the required Heading Rows as F1, F2, F3 and so on and write the query manually, for example, `SELECT [F1],[F2] FROM [Sheet1$]` in the **Query Generator**.

  Also you must specify the sheet structure details as shown in the above scenario:

  - **First Populated Row** = 1

    Heading Row = 6

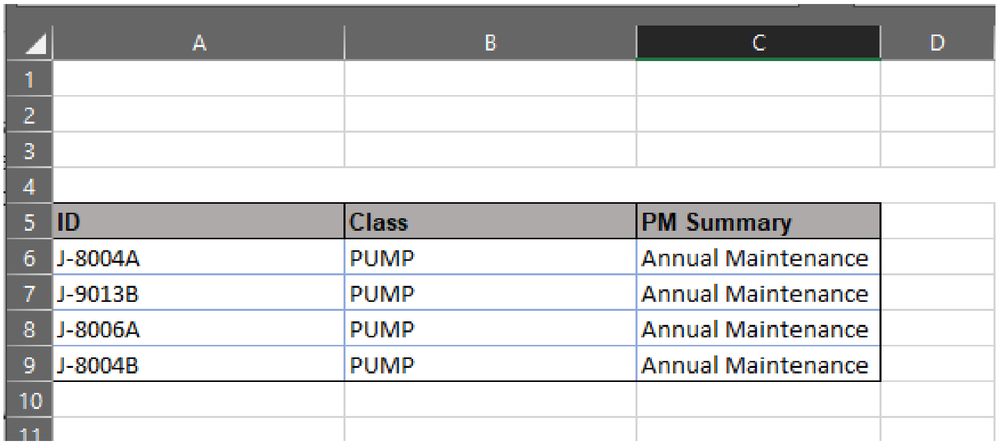
    First Data Row = 7

    Last Data Row = -1 (as it applies to any extent of data row.)

  When you execute the query, then the `Object ID` within the specified limit will be processed.

  Scenario 2:

  If the XLS input file has the following structure, having empty content in the first few rows:



	A	B	C	D
1				
2				
3				
4				
5	ID	Class	PM Summary	
6	J-8004A	PUMP	Annual Maintenance	
7	J-9013B	PUMP	Annual Maintenance	
8	J-8006A	PUMP	Annual Maintenance	
9	J-8004B	PUMP	Annual Maintenance	
10				
11				

- Then in this case **Import Schema** can extract the Heading Rows successfully and these rows will be displayed in the **Column Filter**. Because the rows before the Heading Row are empty, ODBC skips these rows while processing. ODBC returns 5th, 6th, 7th rows as 1st, 2nd, 3rd as actual first 4 rows are empty.

  Also you must specify the Sheet Structure Details as shown in the above scenario:

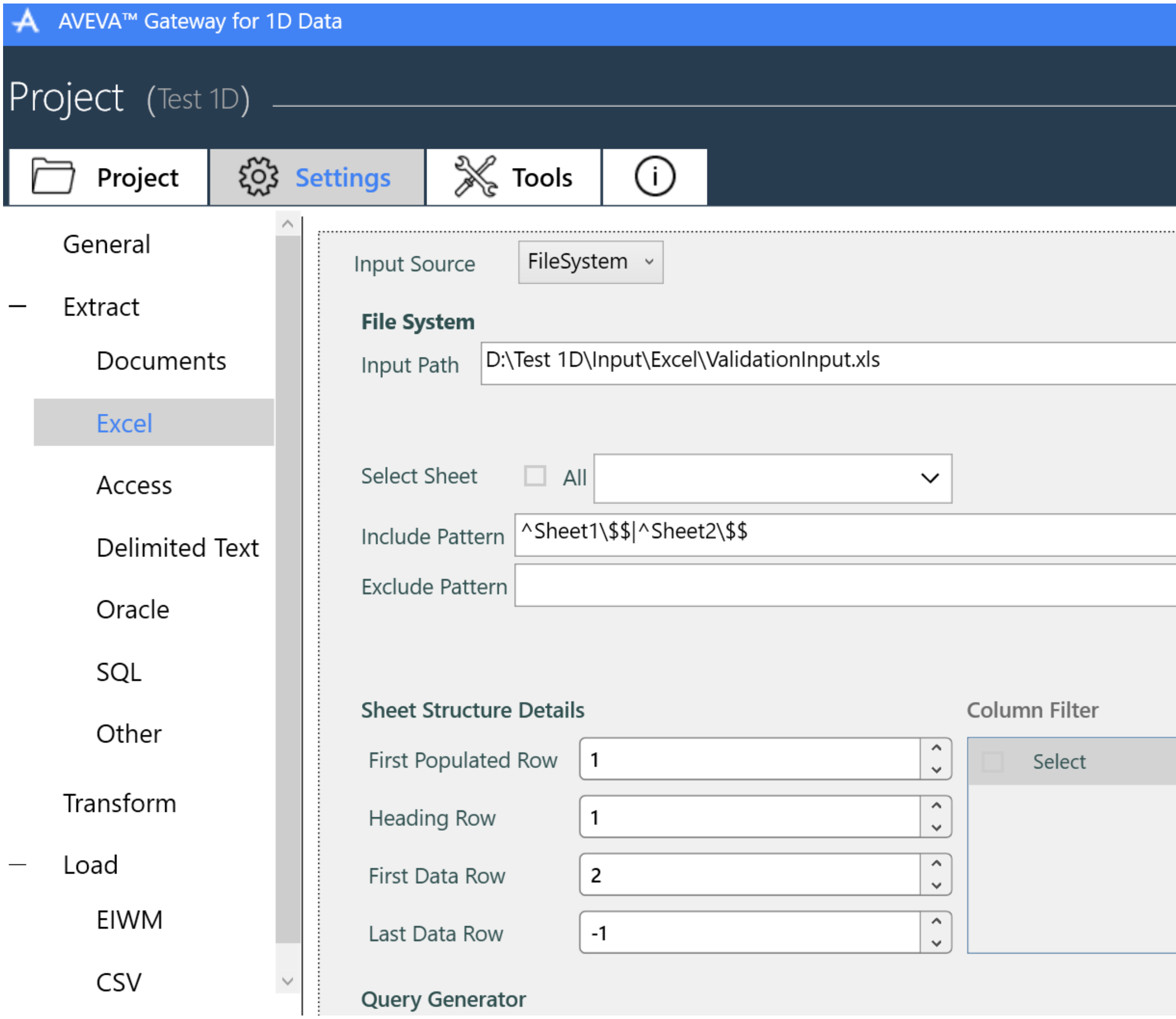
- **First Populated Row** = 5

  Heading Row = 5

  First Data Row = 6

  Last Data Row = -1 (as it applies to any extent of data row.)

When you execute the query, then the **Object ID** within the specified limit will be processed.



Input Source: FileSystem

File System

Input Path: D:\Test 1D\Input\Excel\ValidationInput.xls

Select Sheet: □ All

Include Pattern: ^Sheet1\$$|^Sheet2\$$

Exclude Pattern:

Sheet Structure Details

First Populated Row: 1

Heading Row: 1

First Data Row: 2

Last Data Row: -1

Note: The **Import Schema** is not dependent on Sheet Structure Details. You can populate the **Heading Row** as per the **First Data Row** and **Last Data Row**.

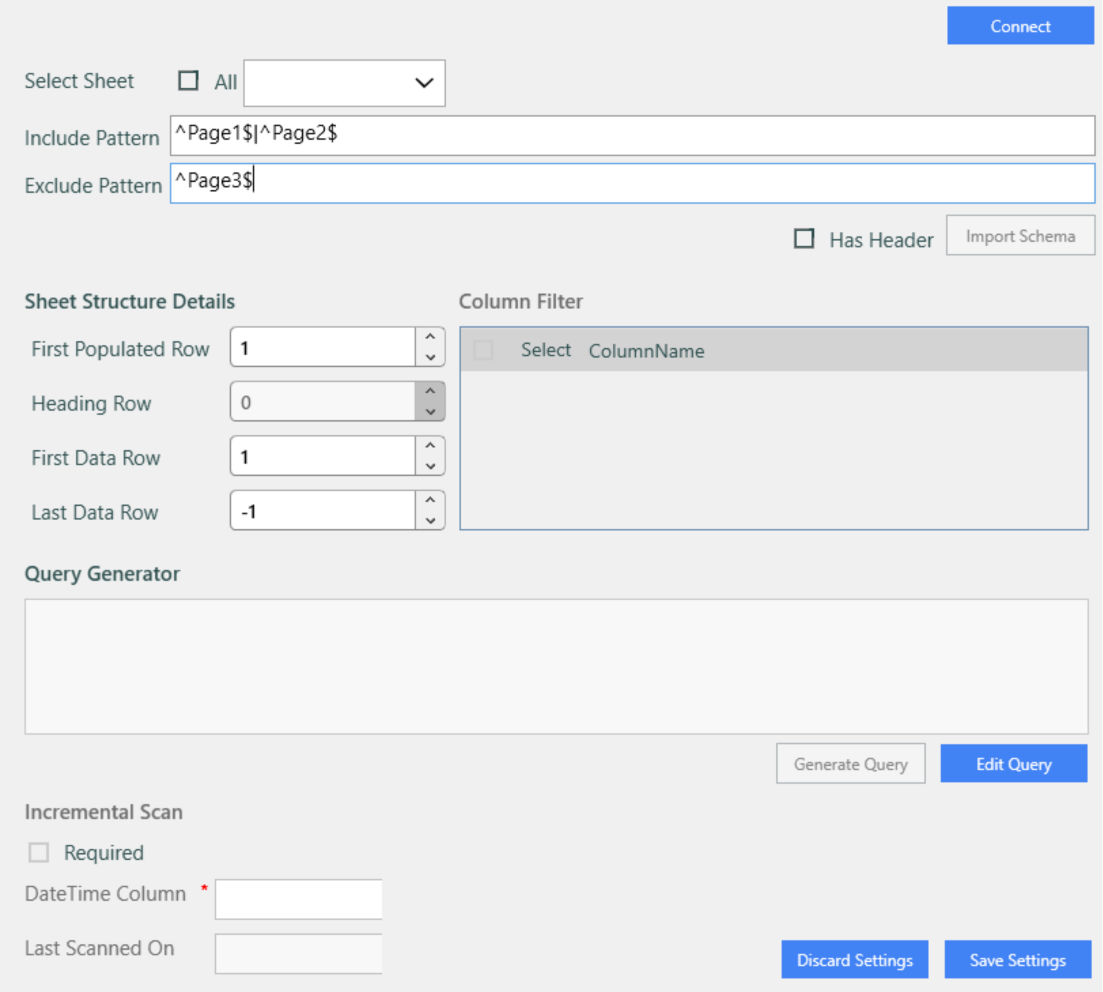
Processing Multiple Worksheets

This feature enables you to process multiple worksheets using a single extract configuration.

- Using the Pattern mapping option, you can include/exclude patterns from the Extract section. The Extractor uses these patterns to select sheets from the available set of sheets in the Excel file. This subset of sheets is then used to further extract data. You can have the following four scenarios to process the multiple worksheets:

- **Scenario 1: When you select a new Excel file (Default scenario)**

  When you select a new Excel file all the old data in the user interface screen stored previously is cleared. When you click Connect, the All check box is selected by default along with all the sheets in the All Sheets combo box. The Include Pattern is automatically generated and displays *ALL_WORKSHEETS* and the Exclude Pattern is empty by default. When you select more than one sheet then the Import Schema button is disabled along with the Column Filter and Query Generator sections. You can only perform Save Settings or Discard Settings and cannot generate a query.



Select Sheet □ All

Include Pattern ^Page1$|^Page2$

Exclude Pattern ^Page3$

□ Has Header

First Populated Row 1

Heading Row 0

First Data Row 1

Last Data Row -1

□ Required

- **Scenario 2: When you select all the sheets**

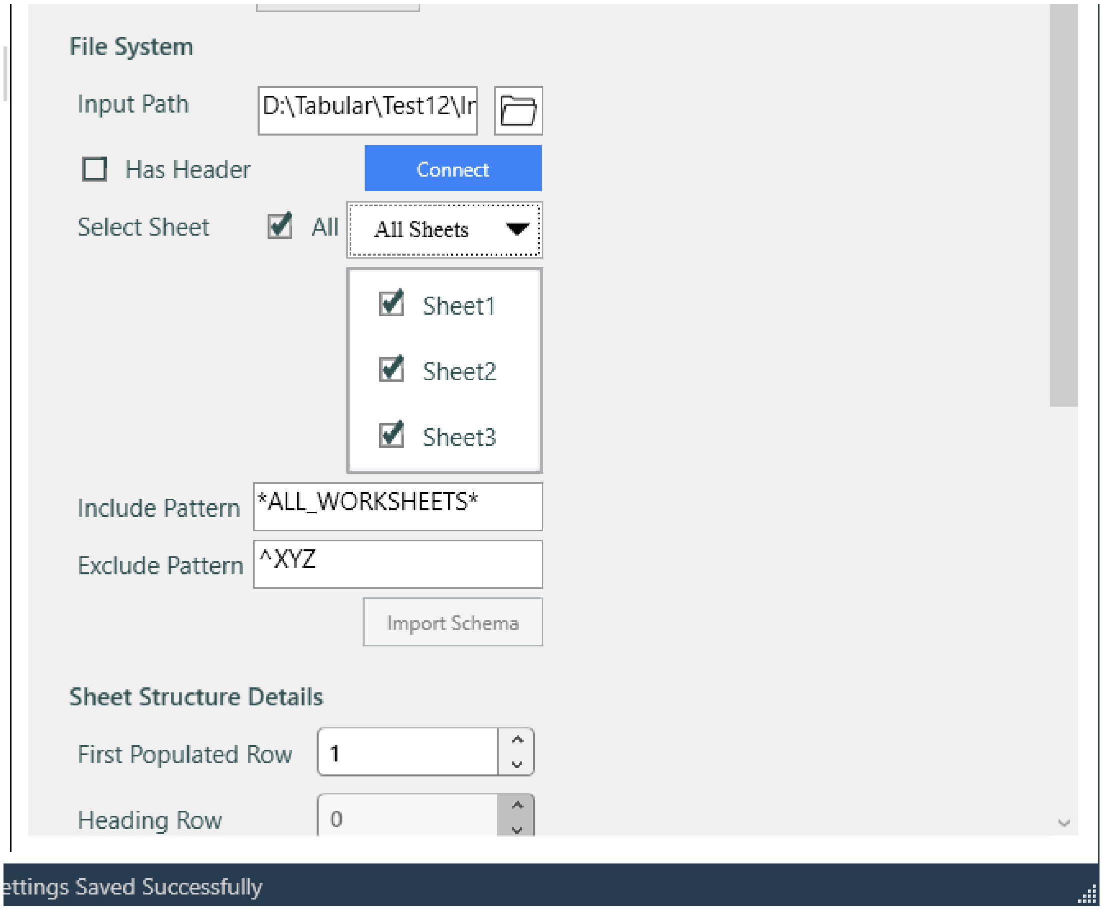
  When you select the **All** check box or individually select all the sheets present in the combo box, it will automatically select the **All** check box and set the Include Pattern to *ALL_WORKSHEETS* and the Exclude Pattern to empty.

- **Scenario 3: When you open a project containing "Include Pattern" and "Exclude Pattern" in the Configuration**

You can also open a project with previously defined Include Pattern and Exclude Pattern options that are saved in the configuration file. These values are used to populate the Sheets comb box on screen. The sheets will also be selected according to Include and Exclude Pattern.

- **Scenario 4: When you have set IncludePattern ="*ALL_WORKSHEETS*" and ExcludePattern="^xyz"**

  You have set `IncludePattern ="*ALL_WORKSHEETS*"` and `ExcludePattern="^xyz"` (as a random pattern). If no sheets match `ExcludePattern`, then it will display the sheets according to `IncludePattern`. The following image displays sheets as per the `IncludePattern`:



File System

Input Path: D:\Tabular\Test12\Ir

☐ Has Header

Select Sheet ☑ All — All Sheets

☑ Sheet1

☑ Sheet2

☑ Sheet3

Include Pattern: *ALL_WORKSHEETS*

Exclude Pattern: ^XYZ

Sheet Structure Details

First Populated Row: 1

Heading Row: 0

ettings Saved Successfully

Configuration Attributes in Excel Configuration:

- `includePattern *required*`: Describes the pattern used to match the sheets that will be included in the extraction process.
  - `valid inputs`: Any valid regular expression pattern, as well as *`ALL_WORKSHEETS*` which extracts every drawing present in the sheet.
- `excludePattern *optional*`: Describes the pattern used to match the sheets that will be excluded from the extraction process.
  - `valid inputs`: Any valid regular expression pattern.

If the selected Excel document `EquipmentList.xls` contains sheets - Sheet1, Sheet2, Sheet3, Test1, Test2, Test3, then the following are the scenarios of extraction:

- To Extract single sheet, you can use the following configuration:

```
<Excel  hasHeader="true">
   <Input source="FileSystem">
      <FileSystem>
         <InputPath>D:\Tabular\Input\EquipmentList.xls</InputPath>
     </FileSystem>
    </Input>
    <Sheet includePattern="^Test1$"/>
</Excel>
```

`^Test1$` is the exact pattern which will match the name of exact sheet.

- **To Extract all sheets,** you can use the following configuration:

```
<Excel  hasHeader="true">
   <Input source="FileSystem">
      <FileSystem>
         <InputPath>D:\Tabular\Input\EquipmentList.xls</InputPath>
     </FileSystem>
    </Input>
    <Sheet includePattern="*ALL_WORKSHEETS*"/>
</Excel>
```

`"*ALL_WORKSHEETS*"` indicates that the user wants to extract every sheet present in the Excel document.

- **Extract subset with include pattern**

  You can use the pattern that matches the subset of sheets that you want to extract. For example, if you want to extract all the sheets which start with the names `Test [Test1, Test2, Test3]`, then use an `includePattern="^Test"`:

```
<Excel  hasHeader="true">
   <Input source="FileSystem">
      <FileSystem>
         <InputPath>D:\Tabular\Input\EquipmentList.xls</InputPath>
     </FileSystem>
    </Input>
    <Sheet includePattern="^Test"/>
</Excel>
```

- **Extract subset with exact matching pattern**

  You can extract the sheets by providing exact matching pattern for `includePattern.` For example, if you want to extract data from sheets `[Test1, Test2, Sheet1]`, then use `includePattern="^Test1$|^Test2$|^Sheet1$":`

```
<Excel  hasHeader="true">
   <Input source="FileSystem">
      <FileSystem>
         <InputPath>D:\Tabular\Input\EquipmentList.xls</InputPath>
     </FileSystem>
    </Input>
    <Sheet includePattern="^Test1$|^Test2$|^Sheet1$"/>
</Excel>
```

This will match the exact sheet names with all the desired sheets in the Excel document.

- Extract subset with exclude pattern

You can use exclude pattern to extract subset when you want to discard some specific sheet(s) from the extraction process. Any sheet name that matches the pattern present in exclude pattern will be removed from the subset.

```
<Excel  hasHeader="true">
   <Input source="FileSystem">
      <FileSystem>
         <InputPath>D:\Tabular\Input\EquipmentList.xls</InputPath>
     </FileSystem>
    </Input>
    <Sheet includePattern="*ALL_WORKSHEETS*" excludePattern="^Sheet2$"/>
</Excel>
```

`includePattern="*ALL_WORKSHEETS*" excludePattern="^Sheet2$"`

This configuration describes a pattern that extracts everything from the Excel document excluding Sheet2 from the process.

Customized extraction (extracting columns selectively) is not supported from a single configuration element. To achieve this, you can use the following configuration:

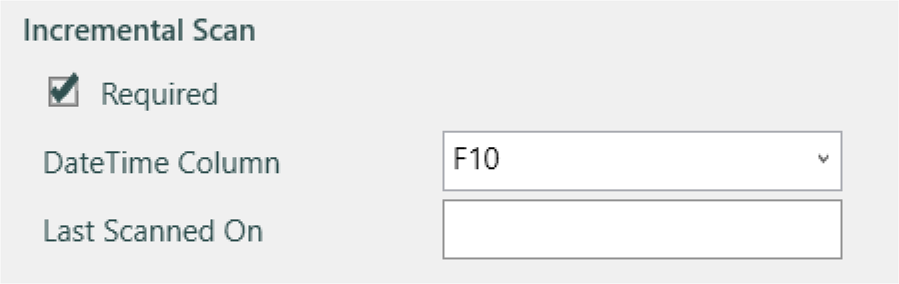
```
<Excel  hasHeader="true">
   <Input source="FileSystem">
      <FileSystem>
         <InputPath>D:\Tabular\Input\EquipmentList.xls</InputPath>
     </FileSystem>
    </Input>
    <Query>SELECT [Column1],[Column2] from [Sheet1$]</Query>
</Excel>

<Excel  hasHeader="true">
   <Input source="FileSystem">
      <FileSystem>
         <InputPath>D:\Tabular\Input\EquipmentList.xls</InputPath>
     </FileSystem>
    </Input>
    <Query>SELECT [Column1],[Column2] from [Sheet2$]</Query>
</Excel>

<Excel  hasHeader="true">
   <Input source="FileSystem">
      <FileSystem>
         <InputPath>D:\Tabular\Input\EquipmentList.xls</InputPath>
     </FileSystem>
    </Input>
    <Query>SELECT [Column1],[Column2] from [Sheet3$]</Query>
</Excel>
```

This configuration allows you to only extract selected columns from the source sheet.

- **Incremental Scan**

  Required: If you select this option, the Gateway keeps track of the last scan date by storing it in a configuration file in the same location where the project configuration file exists. When the Gateway is used for the first time, a text file is created. For each subsequent run, it reads the text file and uses the last scan date to select only those rows that have been updated since that date.

This function relies on one of the columns in the file to store the date-time when the row was created or updated.



Incremental Scan

☑ Required

DateTime Column: F10

Last Scanned On:

- **DateTime Column**: Use this option to build the queries for a specific date-time.
- **Last Scanned On**: Use this option to set the last scan date-time. By default, this field is automatically filled with the latest scan.

- **Discard Settings**: If you do not require those settings any longer, select this box to discard the settings.
- **Save Settings**: Click this to save the Extract Excel settings options.

Access

You can extract the Access-specific data (`.accdb`) from the Gateway by defining specific settings.

To extract the specific settings for Access data:

1. Click **Input Source** drop-down list.

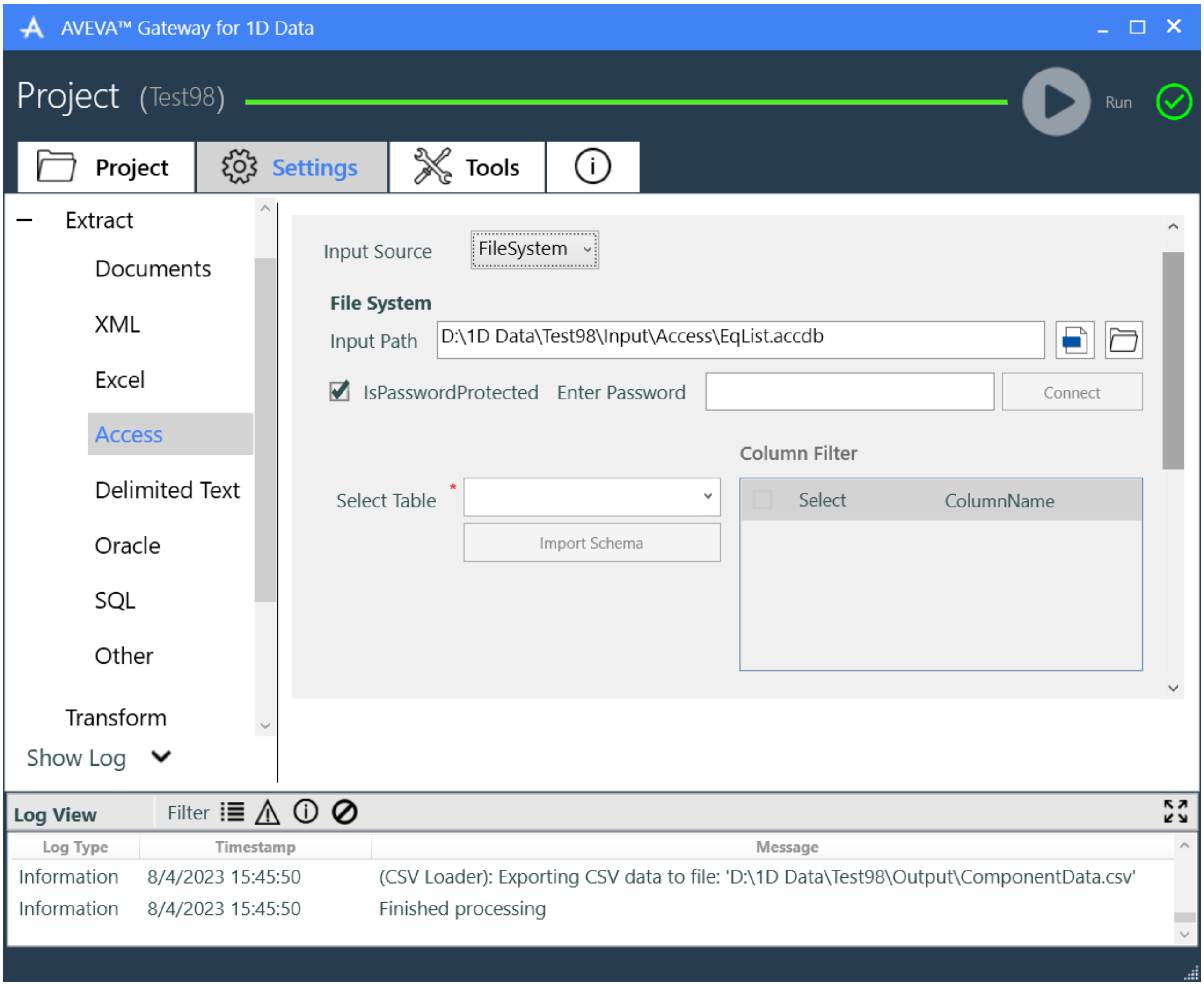
   You can choose either **FileSystem** or **S3 Bucket** from the list.

For FileSystem

- **Input Path**: Browse and select the correct format user input file. It must be of the Access format (.mdb or .accdb).

For S3 Bucket

- **S3 Credential Details**: See Accessing an AWS S3 Bucket () for more information.



Input Source: FileSystem

File System

Input Path: D:\1D Data\Test98\Input\Access\EqList.accdb

☑ IsPasswordProtected Enter Password

Select Table *

Log Type	Timestamp	Message
Information	8/4/2023 15:45:50	(CSV Loader): Exporting CSV data to file: 'D:\1D Data\Test98\Output\ComponentData.csv'
Information	8/4/2023 15:45:50	Finished processing

- **Download**: Click this button to validate the AWS Credentials and download the specified file from the S3 bucket to the input location of the Project folder. **File System Details** will be filled with the downloaded location and file details.

Source Structure: Define the following elements of the Source data structure:

- **IsPasswordProtected**: Select this box only if the file has got a password protection to it.
- **Enter Password**: Type the password for that particular file. When you save the settings, the password gets encrypted and stored in the configuration file.

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

- **Connect**: Click this to establish a connection between the Gateway and from the location where the tables will be extracted from the source file. After this connection, the **Select Table** drop-down box is filled with the sheet name and **Import Schema** box gets populated. If that input is a new one and is not saved in configuration file then by default the first table will be displayed. Otherwise, it populates the last selected table.

- **Select Table**: Select the table that gets populated in the **Select Table** box. Click **Import Schema** to fetch the list of rows in that table. These rows are displayed in **Column Filter** box**,** which you can select to generate a query.
- **Column Filter**: Specify the column names and provides the filter to select particular column data for processing. Click **Select** to select all the columns.
- **Query Generator**: Type the query in this box.
  - **Generate Query**: Click this box to generate the query to fetch the data. Query is based on your selection criteria.
    - **Edit Query**: Click this box to edit the query.
- **Incremental Scan Section**
  - **Required**: If you select this option, the Gateway keeps track of the last scan date by storing it in a configuration file of the extractor in the same location where **Project.xml** exists. When the Gateway is used for the first time, then a text file is created. Next time onwards, it reads the text file and passes this date to the Gateway.
  - **DateTime Column**: Use this option to build the queries for a specific date-time.
  - **Last Scanned On**: Use this option to store the last scan date-time of the most recent scan. The particular last scan data is appended to the existing query and accordingly you can generate the query. By default, this field is automatically filled with the latest scan.
- **Discard Settings**: Click this button if you do not require the settings.
- **Save Settings**: Click this button to save the Extract Access settings options.

Delimited Text:

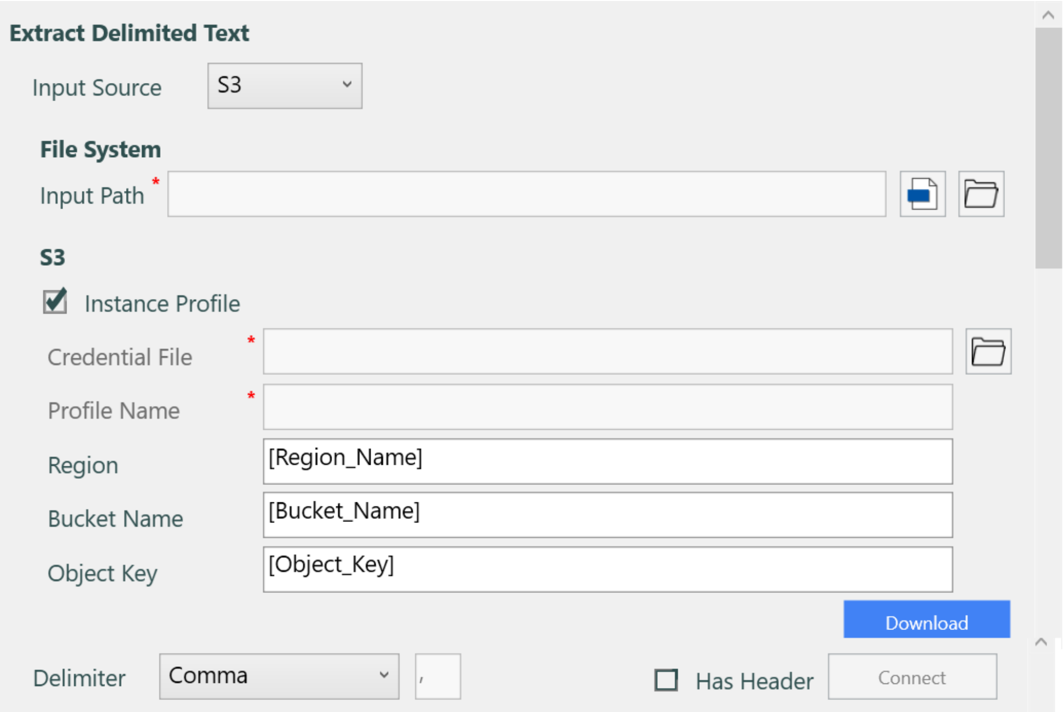
You can extract the Delimited text-specific data from the Gateway by defining specific settings.

Note: The Gateway supports the following Unicode subset ranges:

- Basic Latin (00–7F)
- Latin-1 Supplement (80–FF)
- Latin Extended-A (00–7F)
- Latin Extended-B (80–FF)

To extract the specific settings for Delimited Text data:

1. Click **Input Source** drop-down list.
2. You can choose either **FileSystem** or **S3 Bucket** from the list.



Extract Delimited Text

Input Source: S3

File System

Input Path *

S3

☑ Instance Profile

Credential File *

Profile Name *

Region: [Region_Name]

Bucket Name: [Bucket_Name]

Object Key: [Object_Key]

Delimiter: Comma ,

□ Has Header

For FileSystem:

- **Input Path**: Browse and select the user input file in the correct format, that is, it must be of the delimited text type, .csv or .txt file.
- **Data Source**: Provides the Data Source folder path. It is a non-editable field given just to indicate the content.
- **File Name**: Provides file name. It is a non-editable field to indicate the content.

For S3 Bucket:

- **S3 Credential Details**: See Accessing an AWS S3 Bucket () for more information.
- **Download**: Click this button to validate the AWS Credentials and to download the specified file from S3 bucket to Input Location of the Project folder. **File System Details** will be filled with the downloaded location and file details.

Source Structure: Define the following elements of the Source data structure:

- **Delimiter**: Select the delimiter from the drop-down box. It can be Comma or Tab or any Other delimiter. Select the delimiter from the drop-down box. It can be Comma or Tab or any Other delimiter.
- **Has Header**: Select **Has Header** option for the columns having a header.

  Note: If the columns do not have a header then do not select the **Has header** option. In this case, default names will be assigned to the attributes (columns) of F1, F2, F3, F4 and so on. These can be changed using Transform mapping configuration.

- **Column Filter**: Defines the column names and provides the filter to select particular column data for processing. Click **Select** to select all the columns.
- **Query Generator**: Type the query to fetch the data in the **Query Generator** box. Click this to generate the query based on your selection criteria.

  - **Generate Query**: Click this box to generate the query to fetch the data. Query is based on your selection criteria.
  - **Edit Query**: Click this box to edit the query.
- **Incremental Scan**:
  - **Required**: If you select this option, the Gateway keeps track of the last scan date by storing it in a configuration file of the extractor in the same location where **Project.xml** exists. When the Gateway is used for the first time, then a text file is created. Next time onwards, it reads the text file and passes this date to the Gateway.
  - **DateTime Column**: This option allows you to build the queries for a specific date-time.
  - **Last Scanned On**: This option allows you to store the last scan date-time of the most recent scan. The particular last scan data is appended to the existing query and accordingly you can generate the query. By default, this field is automatically filled with the latest scan.
- **Discard Settings**: Click this button if you do not require the settings.
- **Save Settings**: Click this to save the **Extract Delimited Text** settings options.

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.

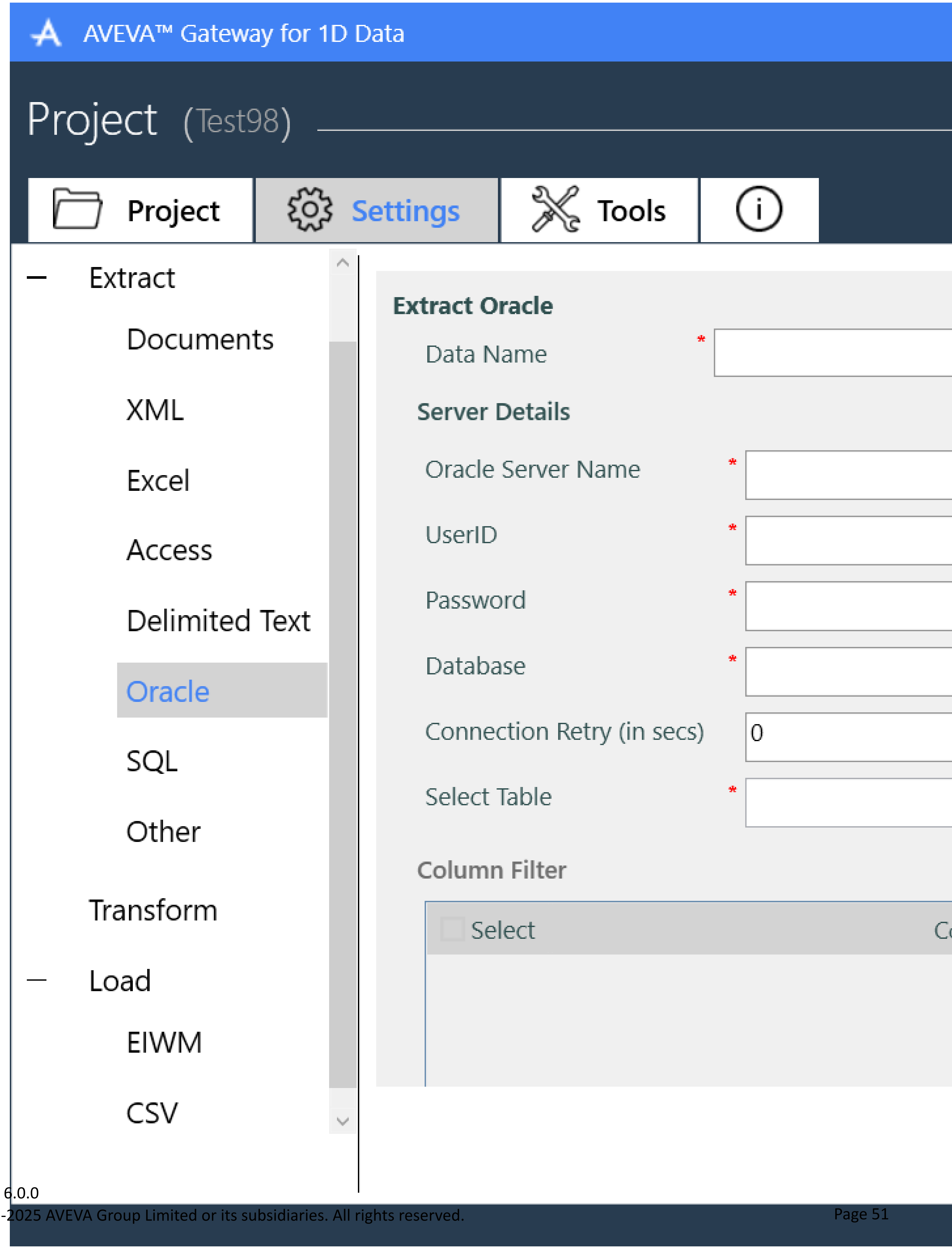
Database Sources

The Gateway extracts the files from data base sources such as Oracle and SQL Server.

Oracle/SQL:

You can extract the Oracle/SQL Server-specific data from the Gateway by defining specific settings in the Gateway.

Oracle Server:



Extract Oracle

Data Name *

Server Details

Oracle Server Name *

UserID *

Password *

Database *

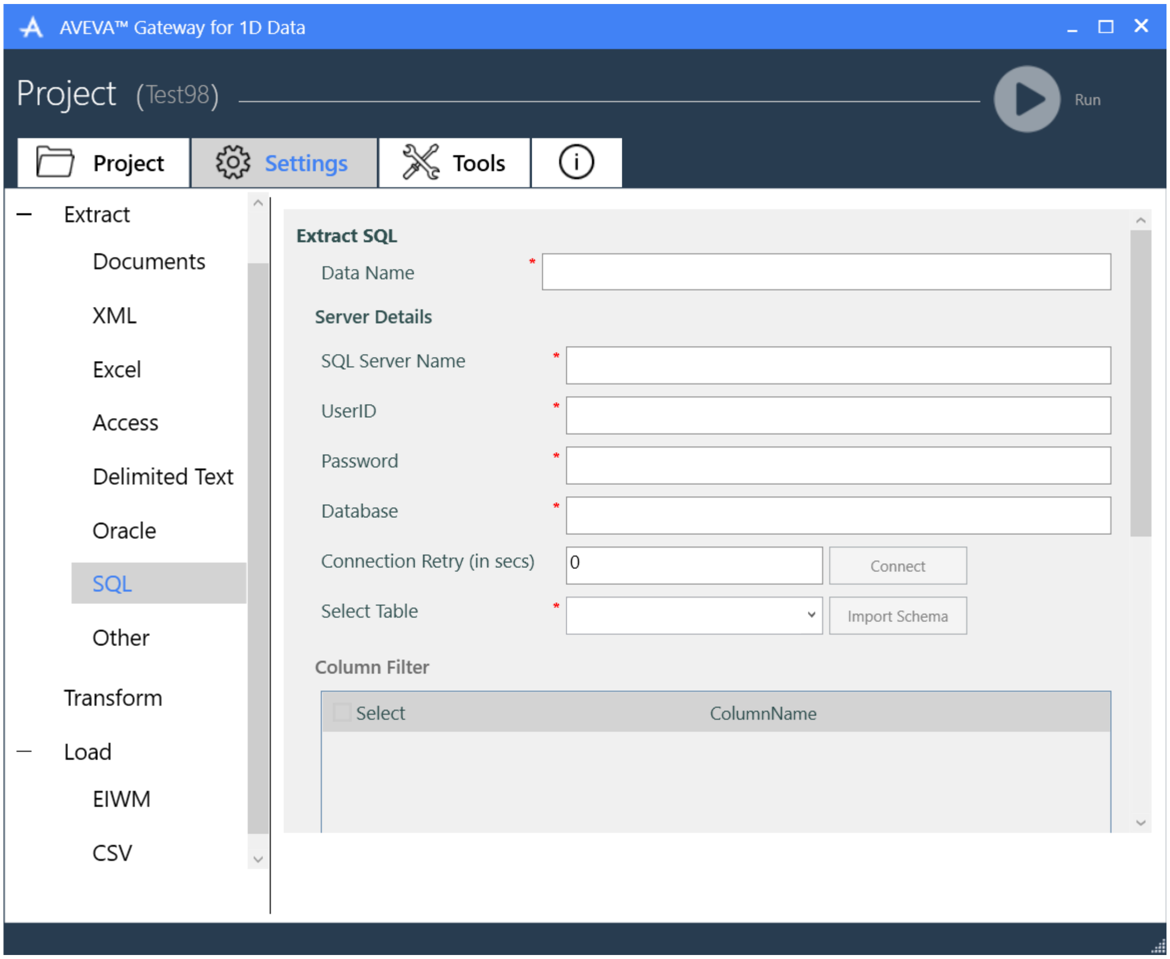
Connection Retry (in secs) 0

Select Table *

Column Filter

□ Select

SQL Server:



Extract SQL

Data Name *

Server Details

SQL Server Name *

UserID *

Password *

Database *

Connection Retry (in secs) 0

Select Table *

Column Filter

Select	ColumnName

To extract the specific settings for Oracle/SQL Server data you will need to install the relevant ODBC driver for that database:

- **Data Name**: Specify the extracted data name that is used to generate the name of the output EIWM file, CSV file and logs.
- **Server Details**: Defines the database server details.
  - **Oracle/SQL Server Name**: Type the database Server name.
  - **UserID**: Type the User name to connect to the database.
  - **Password**: Type the password to connect to the database server.

Notes:

- When you save the settings, the password will get encrypted and will be stored in the configuration file.
- Because the encrypted password is associated with the local machine where the Gateway is running, other machines cannot use the configuration file directly. You will have to encrypt the password before using the configuration created in another machine. You can obtain an encrypted password using the encryption utility.
- **Database**: Type the database name.

  - **Connection Retry Time**: Specify the timeout period for the Gateway to wait for a response from the database Server. When the Gateway requests a query from the database and does not receive a response, for example, if the database server or network is busy, the Gateway process waits for the timeout period and then sends another request to the database.
  - **Connect**: Select this option to connect to database server.
  - **Select Table**: Select the table that gets populated in the box. Click **Import Schema** to fetch the list of rows in that table. These rows are displayed in the **Column Filter** box, which you can select to generate a query.
- **Column Filter**: Specify the column names and provides the filter to select particular column data for processing. Click **Select** to select all the columns.
- **Query Generator**: Type the query to fetch the data in the box.
- **Edit Query**: Click this button to edit the query.

Note: Extraction Query (SQL Joins): You can extract and collate data from multiple tables within a database using a SQL Join syntax in the extraction's query. For example, if the data source has two tables, Orders and Customers with the following schema and data:

OrderID	CustomerID	OrderDate
10308	2	2014-09-18
10309	37	2014-09-19
10310	77	2014-09-20

CustomerID	CustomerName
1	First_Name1 Last_Name1
2	First_Name2 Last_Name2
3	First_Name3 Last_Name3

  - You can use the following SQL query to extract attributes from both the tables having a common CustomerID:

```
SELECT Orders.OrderID, Customers.CustomerName, Orders.OrderDate
FROM Orders
INNER JOIN Customers ON Orders.CustomerID=Customers.CustomerID;
```

  - The resultant set will be as follows:

OrderID	CustomerName	OrderDate
10308	Felicity Cheadle	2014-09-18

- **Incremental Scan Section**
  - **Required**: If you select this option, the Gateway keeps track of the last scan date by storing it in a configuration file of the extractor in the same location where **Project.xml** exists. When the Gateway is used for the first time, then a text file is created. Next time onwards, it reads the text file and passes this date to the Gateway.

- **DateTime Column**: Use this option to build the queries for a specific date-time.
- **Last Scanned On**: Use this option to store the last scan date-time of the most recent scan. The particular last scan data is appended to the existing query and accordingly you can generate the query. By default, this field is automatically filled with the latest scan.
- **Discard Settings**: If you do not require those settings any longer, select this box to discard the settings.
- **Save Settings**: Click this button to save the Oracle/SQL Server settings options.

Notes:

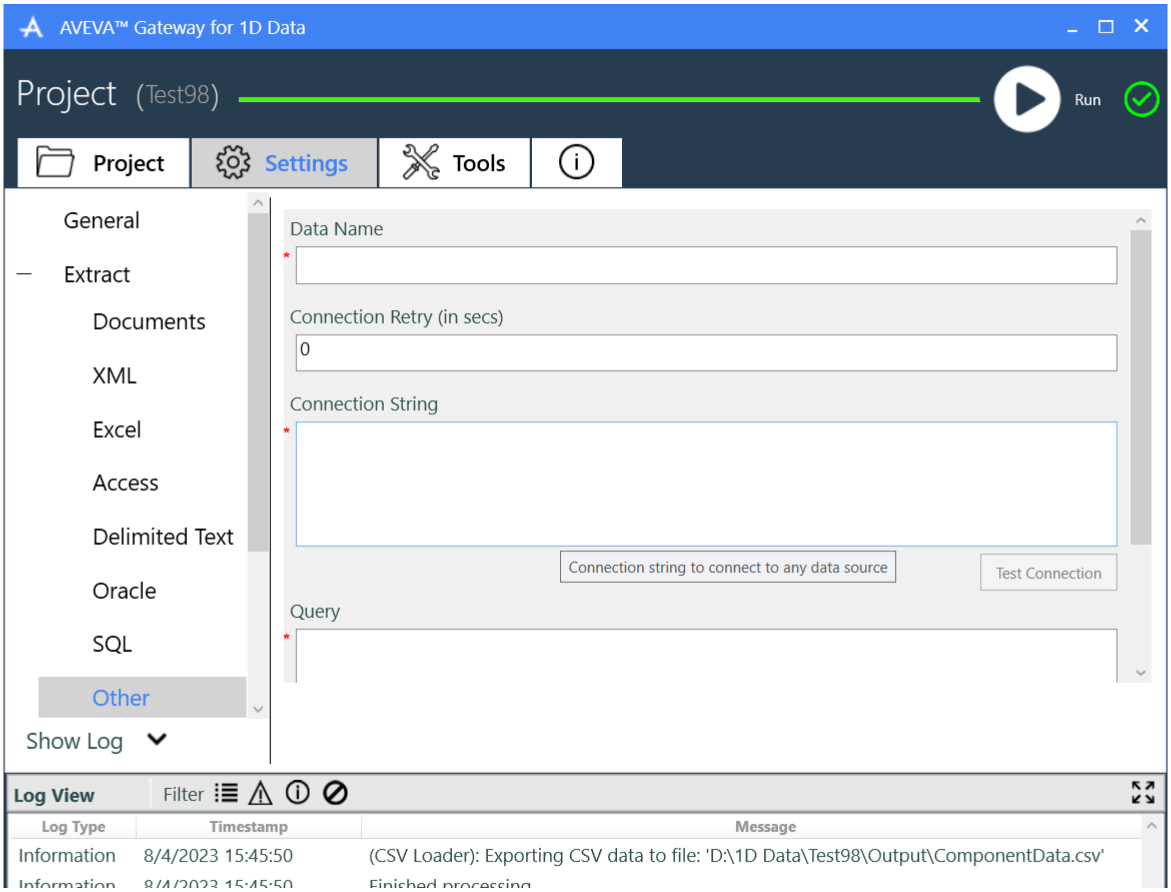
- For security reasons, it is recommended to restrict the database user account with the read-only access to the databases (or the selected tables/views) from which information needs to be extracted. It is required to avoid accidental addition, change or deletion of sensitive data.
- In order to ensure 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.

Other

If you have any other sources for which you have an ODBC driver installed, you can extract data from them by defining the relevant connection and query settings.

Note: As such connections have not been tested with the Gateway, you may not be able to resolve any issues related to these sources.

The following is a screenshot of the Other data sources under **Extract**.



Data Name *

Connection Retry (in secs)
0

Connection String *

Query *

```
Log Type      Timestamp            Message
Information   8/4/2023 15:45:50    (CSV Loader): Exporting CSV data to file: 'D:\1D Data\Test98\Output\ComponentData.csv'
Information   8/4/2023 15:45:50    Finished processing
```

- **Data Name**: Specify the extracted data name that is used to generate the name of the output EIWM file, CSV file and logs.
- **Connection Retry Time**: Specify the time that the Gateway waits to retry connecting to the data source.
- **Connection String: Type the code to connect to the data source supported by** ODBC**.**
- **Query**: Specify the SQL query that will be used to extract data from the data source.

Notes:

- If the other data source is pointing to a database system, it is recommended to restrict the database user account with the read-only access to the databases (or the selected tables/views) from which information needs to be extracted. It is required to avoid accidental addition, change or deletion of sensitive data.
- If the database supports SSL encryption then in order to ensure the security of data which is in transit between the Gateway and the database, you can configure an SSL-encrypted connection between the two.

Transform Settings

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

The default configuration file contains three transform extensions:

- `BaseMapping`: Modifies the engineering data of a model's objects present in the extracted data.
- `ObjectMergeMapping`: Merges the spatially related text strings extracted as Words into a single object based on their X and Y positions for a PDF document.
- `CSVReporting`: Exports the extracted input data from the extracted data in the form of a readable CSV file format. For more information, see Appendix A: Mapping Configuration ().

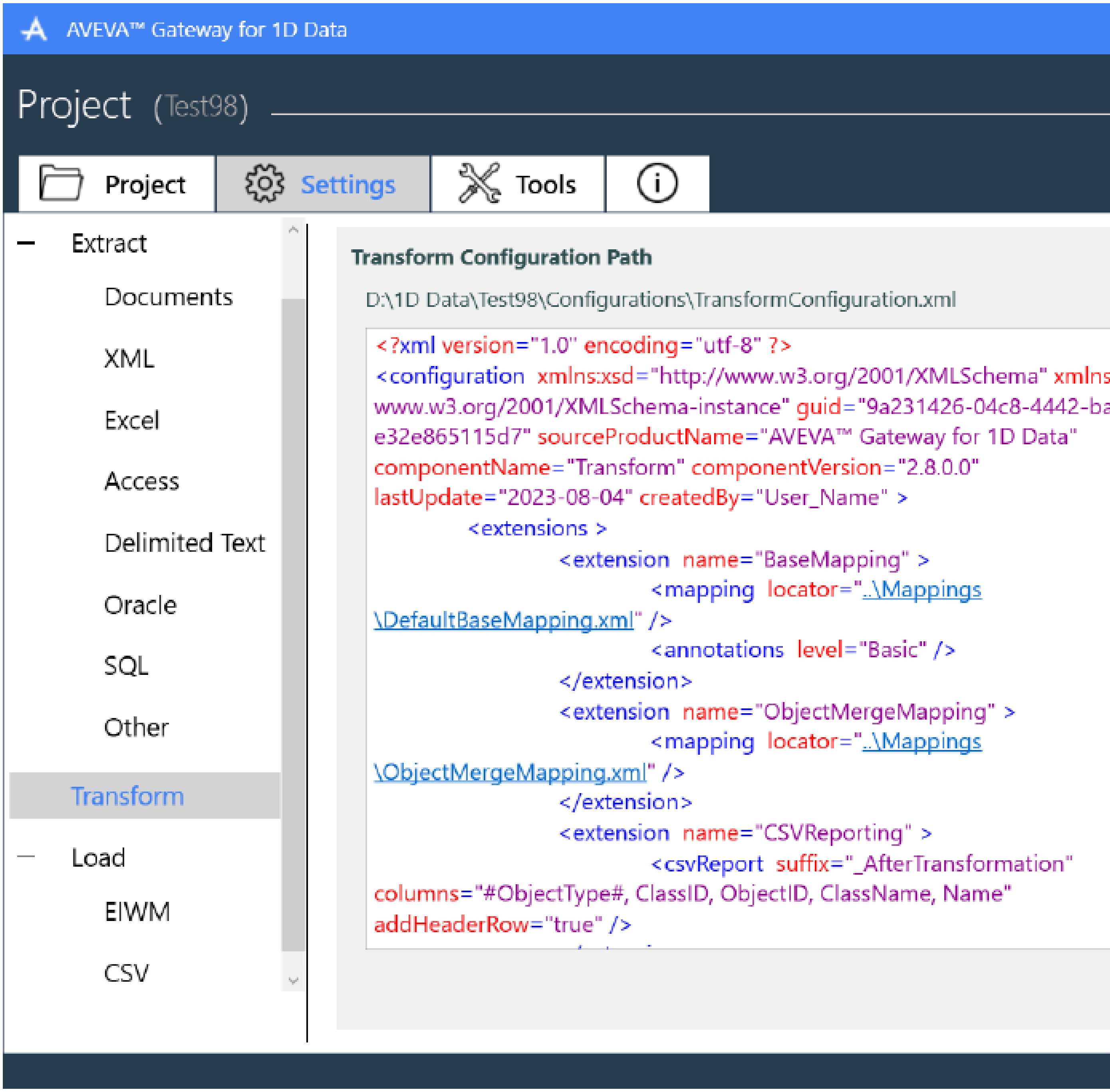
Notes:

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

The **Transform** tab 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.
2. Click **Edit** to open the settings in a **Notepad** where you can modify the default configuration settings relating to Transform.



Project (Test98)

Transform Configuration Path

D:\1D Data\Test98\Configurations\TransformConfiguration.xml

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns
www.w3.org/2001/XMLSchema-instance" guid="9a231426-04c8-4442-ba
e32e865115d7" sourceProductName="AVEVA™ Gateway for 1D Data"
componentName="Transform" componentVersion="2.8.0.0"
lastUpdate="2023-08-04" createdBy="User_Name" >
        <extensions >
                <extension name="BaseMapping" >
                        <mapping locator="..\Mappings
\DefaultBaseMapping.xml" />
                        <annotations level="Basic" />
                </extension>
                <extension name="ObjectMergeMapping" >
                        <mapping locator="..\Mappings
\ObjectMergeMapping.xml" />
                </extension>
                <extension name="CSVReporting" >
                        <csvReport suffix="_AfterTransformation"
columns="#ObjectType#, ClassID, ObjectID, ClassName, Name"
addHeaderRow="true" />
```

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 data.

Notes:

- Any update in the configuration document must be saved before running the process.
- A CSV Report that is defined in the Transformation configuration file can have its output path defined by the `outputFolder` parameter. This parameter is documented as optional but is mandatory as it is a necessary part of the CSV file definition. Currently, these only folder definitions in a File System are supported, not an S3 Bucket.
- 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 extension configuration. Available `annotations level` values are `'None'` and `'Basic'`. For example:

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

  For more information, see Appendix B: CSV Reporting ().
- For more information about handling system attributes, see Appendix G: Handling of System Attributes ().

Load Settings

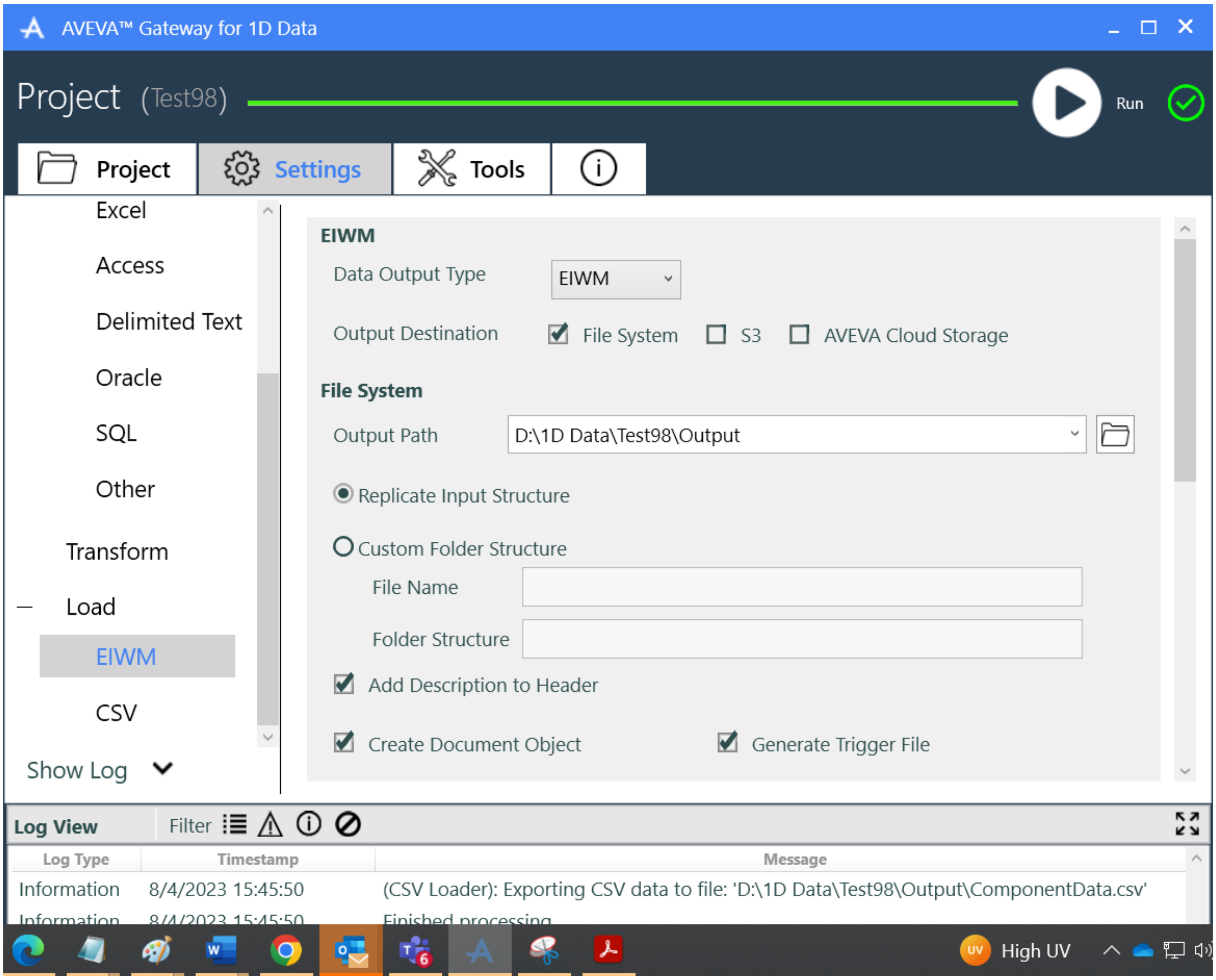
The Gateway uses the **Load** component to load the selected and transformed objects into AIM via EIWM files. It can also export the transformed data into CSV files.

The following code shows an example of Load configuration:

```
<configuration xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/
2001/XMLSchema-instance" sourceProductName="AVEVA.Gateways.2DData" componentName="Load"
componentVersion="2.10.0.0" >
  <components>
    <component name="LoadEIWM"
locator="D:\Tabular\Test1\Configurations\LoadEIWMConfiguration.xml" />
    <component name="LoadCSV"
locator="D:\Tabular\Test1\Configurations\LoadCSVConfiguration.xml" />
  </components>
</configuration>
```

Load EIWM

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



EIWM

Data Output Type: EIWM

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

File System

Output Path: D:\1D Data\Test98\Output

● Replicate Input Structure

○ Custom Folder Structure

File Name:

Folder Structure:

☑ Add Description to Header

☑ Create Document Object ☑ Generate Trigger File

```
Log Type     Timestamp           Message
Information  8/4/2023 15:45:50   (CSV Loader): Exporting CSV data to file: 'D:\1D Data\Test98\Output\ComponentData.csv'
Information  8/4/2023 15:45:50   Finished processing
```

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

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

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

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

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

  - **File System Details**:
    - **Output Path**: Type the output file path or click the **Browse for Folder** tab to enter the output folder path.

  - **S3 Bucket Details**: Writing the output files to an AWS S3 bucket is an option when using the Gateway with AWS resources. To access the S3 bucket, it requires the relevant authentication details like credential file, region information, bucket name and output directory. For more information, see Accessing an AWS S3 Bucket ().

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

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

- **Replicate Input Structure**: Select this option to generate an output folder structure from the base input path. Any sub-folders below the input locator will be replicated in the output destination, and the files generated from each input file in that structure will be in the corresponding output sub-folders. You can configure this using the `FolderStructure` attribute of the `OutputDestination` element, for example, `<OutputDestination Destination="FileSystem" FolderStructure="keepInputFolderStructureForOutput ">`

- **Custom Folder Structure**: Select this option to enable the **File Name** and **Folder Structure** fields 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]" />
            <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 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 either to allow more efficient imports or when large EIWM files (approximately > 60 MB) fail to import via the AVEVA NET Import Controller.

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

Expected Values:

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

Expected Entries in a Folder Structure	**Description**
Engineering attributes	Attributes extracted from the data source along with the attributes created during the base mapping can be used as elements in the folder structure, for example, `[Tag]`, `[Custom Attribute]` and so on. Also the reserved attributes such as `[TemplateID]`, `[ObjectID]`, `[ClassID]`, `[ContextID]`, `[ObjectName]`, and `[Revision]` can be used. **Note**: Nested context must be resolved as separate sub folders in the hierarchy.
EIWM reserved attributes	EIWM reserved attributes can be used as elements in the folder structure:

Expected Entries in a Folder Structure	Description
	[eiwm:id], [eiwm:classid], [eiwm:name], [eiwm:revision] or [eiwm:context]
Manifest attributes	For more information about manifest attributes, refer to Manifest References ().
Reserved placeholders	Reserved placeholders can be used as elements in the folder structure: • [Date]: Provides the value of the current date. Default format for date is dd-MM-yyyy. This attribute provides a format string, for example, [Date{d.MMM.yy}]. • [Time]: Provides the value of the current time. Default format for date is hh-mm-ss. This attribute provides a format string, for example, [Time{H.mm}]. • [DateAndTime]: Provides the value of the current date and time. Default format for date is dd-MM-yyyy hh-mm-ss. This attribute provides a format string, for example, [Date{d.MMM.yy H:mm}]. **Note**: The formats can be from any of the formats as specified under MSDN DateTime Formats.
Static values	The static values are simple text values and left as is and can be defined without any square brackets. Example: <FileName value="output_[eiwm:id].xml" />. Here "output_" is a static value.

Notes:

- Every "\" in the folder hierarchy is treated as a delimiter and is resolved as a subfolder.
- Configured attributes that resolve to an empty value are skipped during the path generation.
- The values that are not resolved are skipped and a warning is logged.
- Characters such as ‘/’, ‘:’, ‘*’, ‘?’, ‘“’, ‘<’, ‘>’ and ‘|’ are not allowed and are skipped if found in the resolved attribute values or in the static values. A warning is logged.
- Attributes that need to be resolved must be written within square brackets, for example, [name], [id] or [DateAndTime].

- **Create Document Object: A Document Object is the metadata object present in the EIWM that describes the input data source. It holds the information about the data source for extraction like filename, date and time of extraction and so on. The default value of** createDocumentObject **option is** true**, and is set in the** <configuration> **node by the parameter** <createDocumentObject apply="true" />'**.**

  When this option is true, the document object is created in the EIWM output file and all other objects have associations "is referenced in" to this document object.

  When this option is false, the document object is not created in the EIWM output file and all other objects have no "is referenced in" associations.

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. You can 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.

- **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. The 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>
    ```

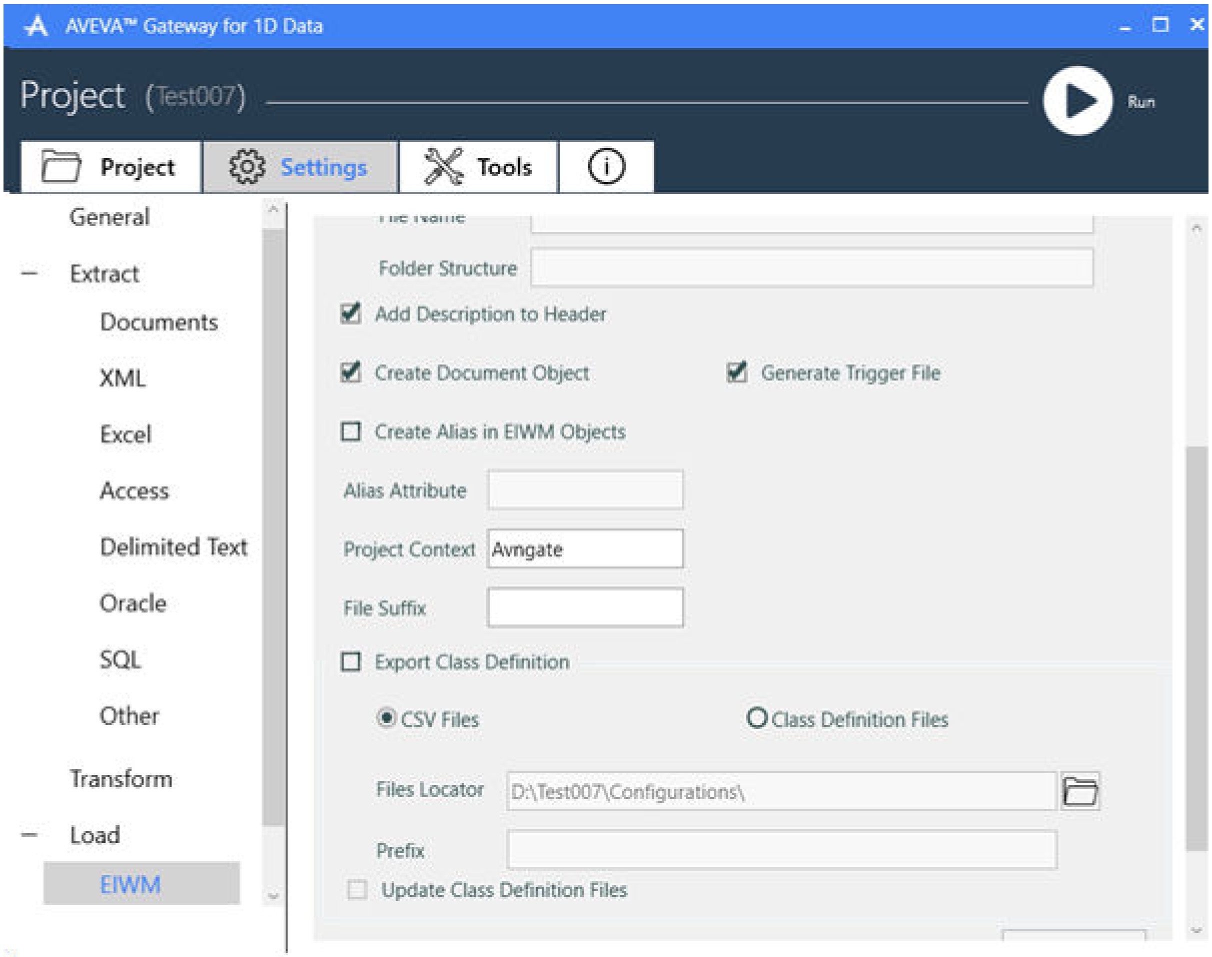
    The output object should look similar to the following, where no alias is created, as both the `ObjectID` and `Alias` attribute are mapped to the same attribute "`GlobalID`".

    ```
    <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 NET 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.



File Name

Folder Structure

☑ Add Description to Header

☑ Create Document Object ☑ Generate Trigger File

☐ Create Alias in EIWM Objects

Alias Attribute

Project Context: Avngate

File Suffix

☐ Export Class Definition

● CSV Files ○ Class Definition Files

Files Locator: D:\Test007\Configurations\

Prefix

☐ Update Class Definition Files

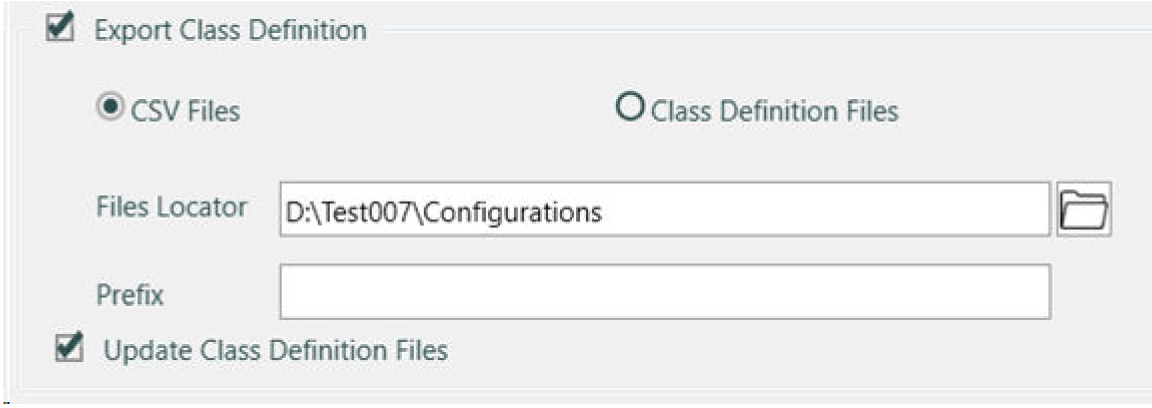
- **Export Class Definition**: Select this option to generate AIM bootstrap files that can be imported into AIM by the Import Controller to generate or update the AIM schema definition for: classes, characteristics, properties, associations and permissible associations. You can configure this by selecting 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.

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** button, 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.



☑ Export Class Definition

● CSV Files ○ Class Definition Files

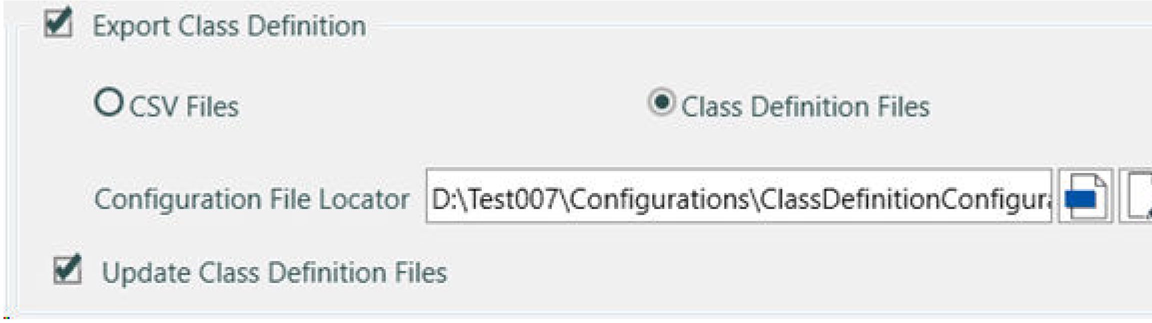
Files Locator D:\Test007\Configurations

Prefix

☑ Update Class Definition Files

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

Configuration File Locator: Selects the location for the Class Definition Configuration xml file. This configuration file contains the details of the names and parameters of the 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. 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.



☑ Export Class Definition

○ CSV Files ● Class Definition Files

Configuration File Locator D:\Test007\Configurations\ClassDefinitionConfigur

☑ Update Class Definition Files

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

In the above image, there are five different nodes for: Class, CharacteristicClass, PropertyClass, AssociationClass and PermissibleAssociation. Each node is used to generate the respective output bootstrap files.

Notes:

- In each node, the **[path]** attribute value is mandatory to generate the respective output bootstrap files. It should be kept as empty value, that is, path = “”, if that output bootstrap file generation in not needed. Otherwise, it should point to the already existing bootstrap file for that particular node class, or a new file can be created by giving the full path along with the xml filename.
- For the Association node, there is only one option to use in the **ClassDefinitionConfiguration.xml** file, that is, "type" of the association which is the association itself. The same attribute "`type`" is used to generate the "`ID`", "`SourceRole`" and "`TargetRole`".
- Update Class Definition Files controls whether new classes identified in the input data are appended to the existing output bootstrap files. If cleared, any existing output bootstrap files of the same name are overwritten.

The following code shows an example of Load EIWM configuration:

```
<configuration xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://
www.w3.org/2001/XMLSchema-instance" sourceProductName="AVEVA™ Gateway for 1D Data"
componentName="LoadEIWM" componentVersion="2.8.0.0" >
  <Outputs>
    <S3 isSelected="false">
      <Authentication instance="true">
        <CredentialFile path="default" profileName="default" />
      </Authentication>
      <Region />
      <BucketName />
      <OutputDirectory />
    </S3>
    <FileSystem isSelected="true">
      <OutputPath>D:\Test007\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"
contextSuffix="" />
  <annotations level="Basic" />
  <createClassDefinitionFiles isSelected="true">
    <updateClassDefinitionFiles apply="true" />
    <csvClassDefinitionFiles>
      <classDefinitionFilesLocator path="D:\Test007\Configurations" />
      <prefix />
    </csvClassDefinitionFiles>
    <xmlClassDefinitionFiles>
      <classDefinitionFileConfigLocator
path="D:\Test007\Configurations\ClassDefinitionConfiguration.xml" />
    </xmlClassDefinitionFiles>
  </createClassDefinitionFiles>
</configuration>
```

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

The following code shows an example of Load configuration:

```
<configuration xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://
   www.w3.org/2001/XMLSchema-instance"
   sourceProductName="AVEVA.Gateways.2DData" componentName="Load"
   componentVersion="2.10.0.0" >
     <components>
       <component name="LoadEIWM"
   locator="D:\Tabular\Test1\Configurations\LoadEIWMConfiguration.xml" />
       <component name="LoadCSV"
   locator="D:\Tabular\Test1\Configurations\LoadCSVConfiguration.xml" />
     </components>
   </configuration>
```

The following code shows an example of Load EIWM configuration:

```
<configuration xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://
   www.w3.org/2001/XMLSchema-instance"
   sourceProductName="AVEVA.Gateways.2DData" 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 enables you to set the format and the naming convention of the output XML. You can modify this setting 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:

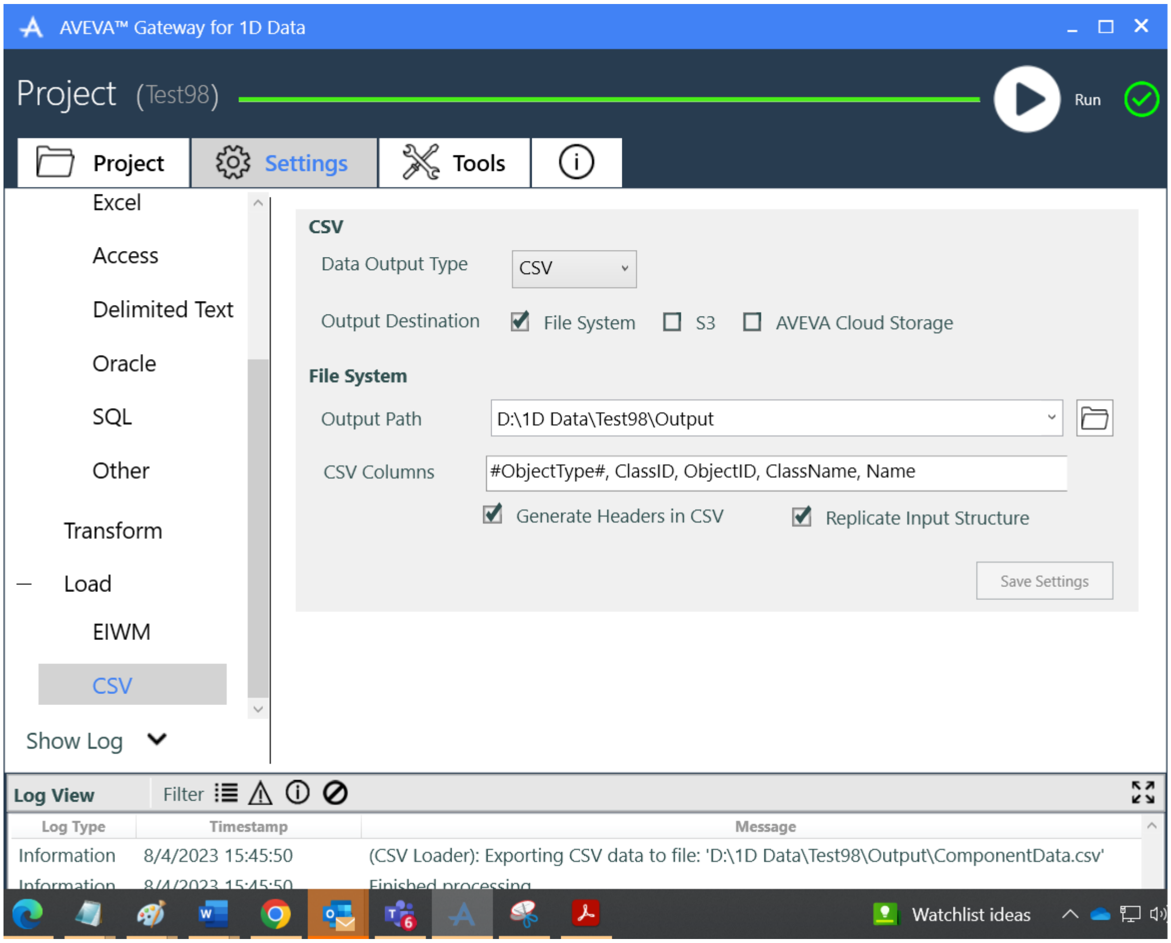
- **Context Suffix**: You can set the EIWM file name's suffix to be appended to the end of the file name. You can also set it to empty, in which case the EIWM file name suffix will be set to `_avngate` or `_null`, depending on the **projectContext value.**
- **Splitting Output File**: You can split the output across multiple files based on the number of objects identified during conversion as valid EIWM objects. This output setting can be modified using the `<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 the `<file ... suffix="_part_"` attribute in the loader configuration setting. The output for the 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 the `<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 CSV

Load exports CSV data together with metadata to CSV format. You can select the Load output type CSV format under the **Load, CSV** settings.



CSV

Data Output Type: CSV

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

File System

Output Path: D:\1D Data\Test98\Output

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

☑ Generate Headers in CSV ☑ Replicate Input Structure

```
Log Type     Timestamp           Message
Information  8/4/2023 15:45:50   (CSV Loader): Exporting CSV data to file: 'D:\1D Data\Test98\Output\ComponentData.csv'
Information  8/4/2023 15:45:50   Finished processing
```

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 ACS.
    - File System Details:
        - **Output Path**: Type the output file path or click Browse for Folder tab 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 this option.
    - **CSV Columns**: In this field, add the additional comma-separated values, for example: `ClassID`, `ObjectID`, `ClassName`, `Name` to export the objects' selected attributes in the CSV export. The selected attributes and associations from from the Tag list are exported into the CSV report. In order to export all attributes, you can use the keyword: `#Attributes`; and to export associations, use the keyword: `#Associations`.
    - **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 sub-folders with files of such type.

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

Execute Projects

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

To execute the selected project:

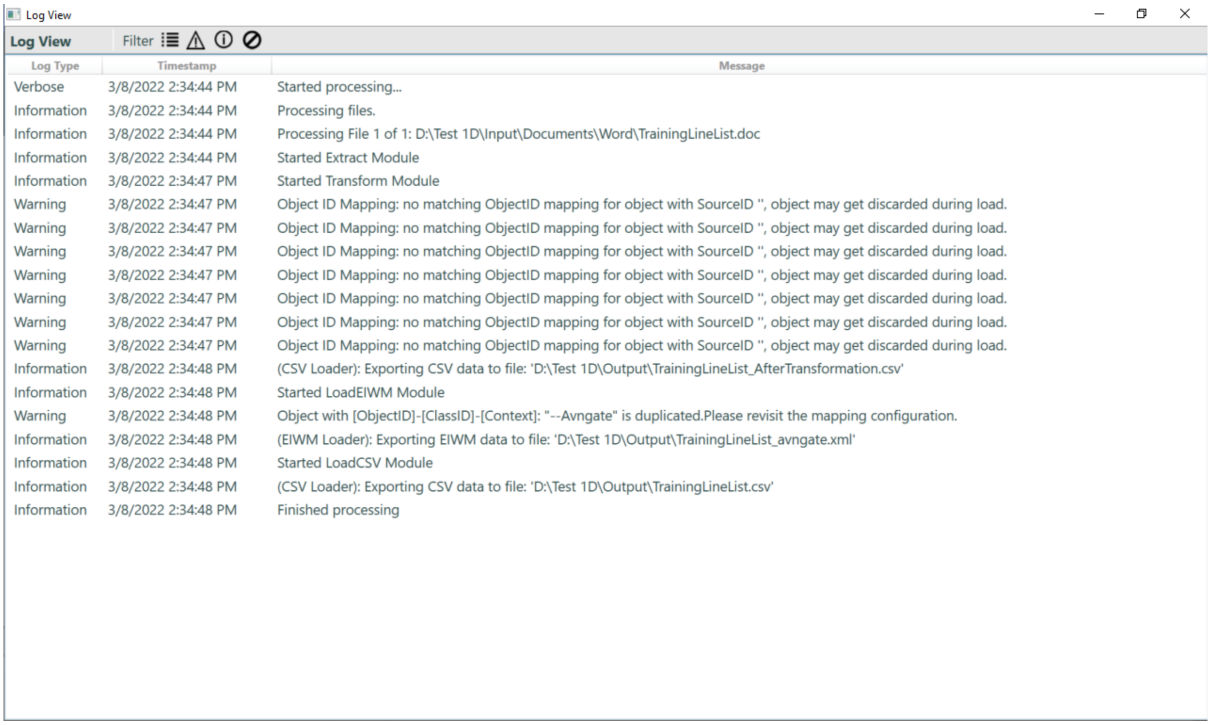
1. Click **Save Settings** to save the configuration settings of **Extract**, **Transform** and **Load**.

   After the successful saving of the configuration files, it enables the **Run** button.

2. Click **Run** to execute the project.

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

LOG: Click this button to retrieve the log information containing log type, date, time and message information, as shown below:



```
Log Type     Timestamp              Message
Verbose      3/8/2022 2:34:44 PM    Started processing...
Information  3/8/2022 2:34:44 PM    Processing files.
Information  3/8/2022 2:34:44 PM    Processing File 1 of 1: D:\Test 1D\Input\Documents\Word\TrainingLineList.doc
Information  3/8/2022 2:34:44 PM    Started Extract Module
Information  3/8/2022 2:34:47 PM    Started Transform Module
Warning      3/8/2022 2:34:47 PM    Object ID Mapping: no matching ObjectID mapping for object with SourceID '', object may get discarded during load.
Warning      3/8/2022 2:34:47 PM    Object ID Mapping: no matching ObjectID mapping for object with SourceID '', object may get discarded during load.
Warning      3/8/2022 2:34:47 PM    Object ID Mapping: no matching ObjectID mapping for object with SourceID '', object may get discarded during load.
Warning      3/8/2022 2:34:47 PM    Object ID Mapping: no matching ObjectID mapping for object with SourceID '', object may get discarded during load.
Warning      3/8/2022 2:34:47 PM    Object ID Mapping: no matching ObjectID mapping for object with SourceID '', object may get discarded during load.
Warning      3/8/2022 2:34:47 PM    Object ID Mapping: no matching ObjectID mapping for object with SourceID '', object may get discarded during load.
Warning      3/8/2022 2:34:47 PM    Object ID Mapping: no matching ObjectID mapping for object with SourceID '', object may get discarded during load.
Information  3/8/2022 2:34:48 PM    (CSV Loader): Exporting CSV data to file: 'D:\Test 1D\Output\TrainingLineList_AfterTransformation.csv'
Information  3/8/2022 2:34:48 PM    Started LoadEIWM Module
Warning      3/8/2022 2:34:48 PM    Object with [ObjectID]-[ClassID]-[Context]: "--Avngate" is duplicated.Please revisit the mapping configuration.
Information  3/8/2022 2:34:48 PM    (EIWM Loader): Exporting EIWM data to file: 'D:\Test 1D\Output\TrainingLineList_avngate.xml'
Information  3/8/2022 2:34:48 PM    Started LoadCSV Module
Information  3/8/2022 2:34:48 PM    (CSV Loader): Exporting CSV data to file: 'D:\Test 1D\Output\TrainingLineList.csv'
Information  3/8/2022 2:34:48 PM    Finished processing
```

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 are 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` attribute 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.

  Note: **Region** is a required field and cannot be empty.

- **Bucket Name**: (Case-sensitive) Define the name of the Bucket which hosts the file's location.

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

- **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/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.

Notes:

- You must restrict the AWS access credential, input files, configurations files and their directory to a specific IP address and user group/account, to protect the data from being accessed illegally.
- For security reasons, it is recommended to restrict Amazon S3 Bucket Access to a specific IAM role. You can apply the following to ensure that the S3 bucket access is restricted to a specific IAM role.

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

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

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

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

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

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

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

Access AVEVA Cloud Storage

Prerequisites:

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

Ensure that the following system environment variables are defined:

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

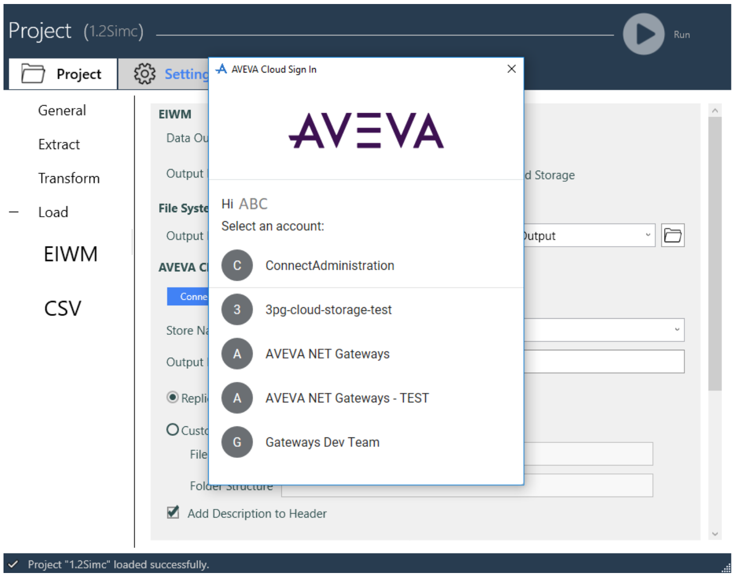
Note: The above-listed environment variables are mandatory for connecting to ACS.

ACS Gateway Configuration:

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

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

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



Hi ABC

Select an account:

- C ConnectAdministration
- 3 3pg-cloud-storage-test
- A AVEVA NET Gateways
- A AVEVA NET Gateways - TEST
- G Gateways Dev Team

☑ Add Description to Header

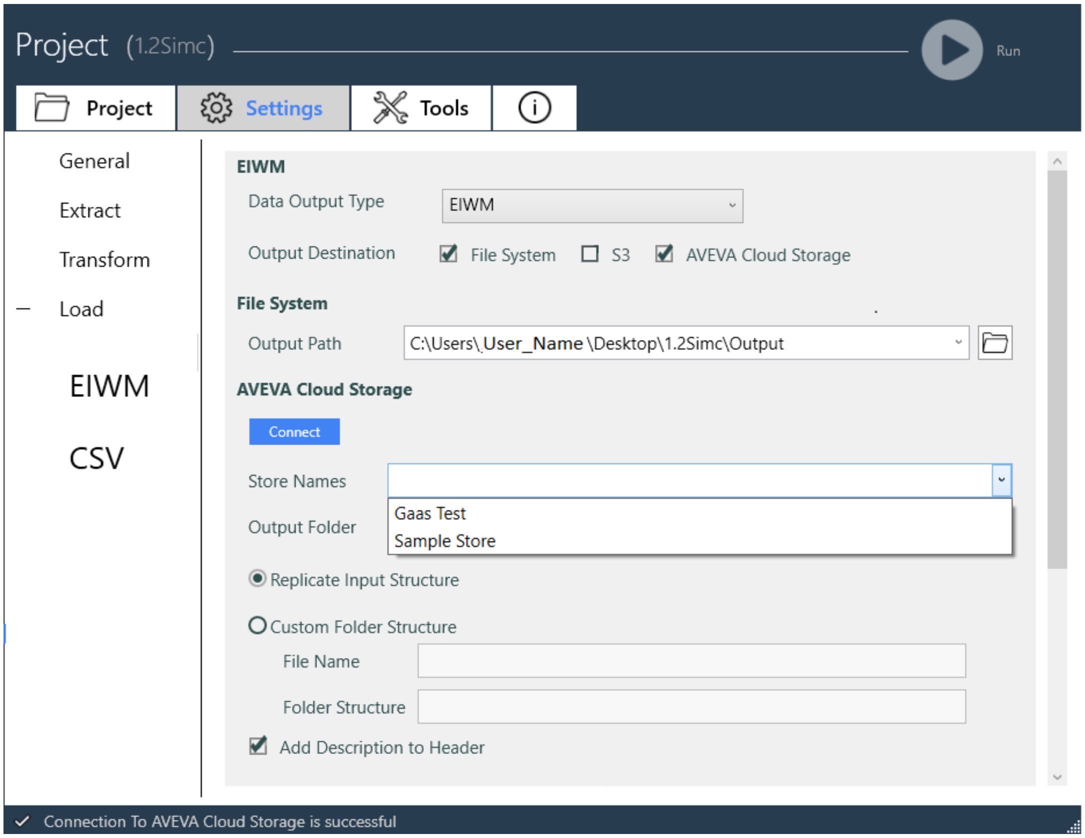
✓ Project "1.2Simc" loaded successfully.

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.



EIWM

Data Output Type: EIWM

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

File System

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

AVEVA Cloud Storage

Store Names:

Output Folder:

Gaas Test
Sample Store

● Replicate Input Structure

○ Custom Folder Structure

File Name:

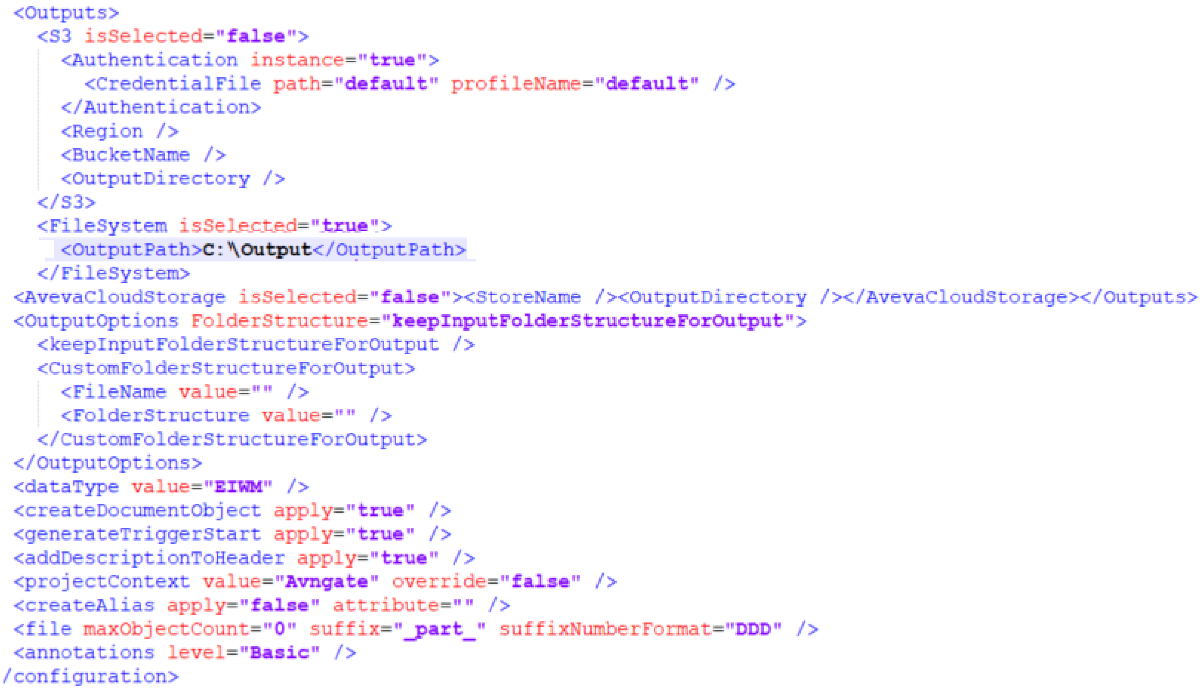
Folder Structure:

☑ Add Description to Header

✓ Connection To AVEVA Cloud Storage is successful

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

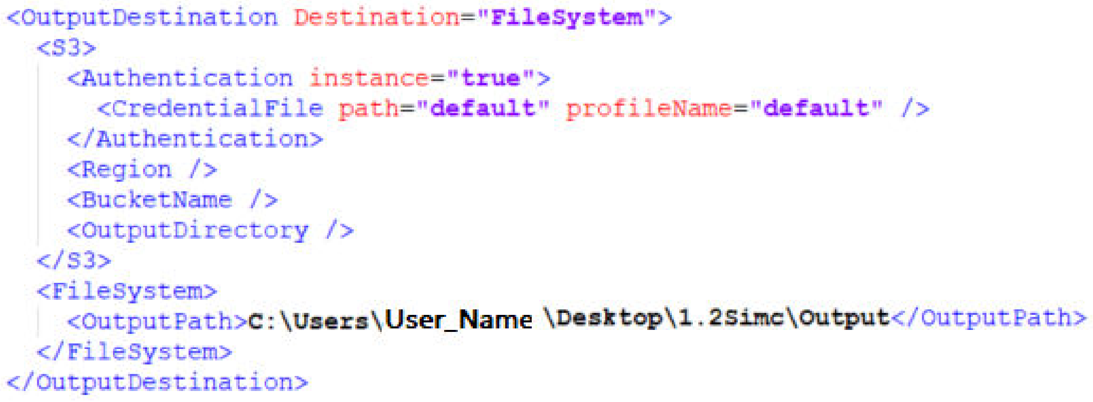
The Loaders configuration files such as EIWM and CSV files are stored in the Outputs folder in the following configuration format:



```
<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>
```

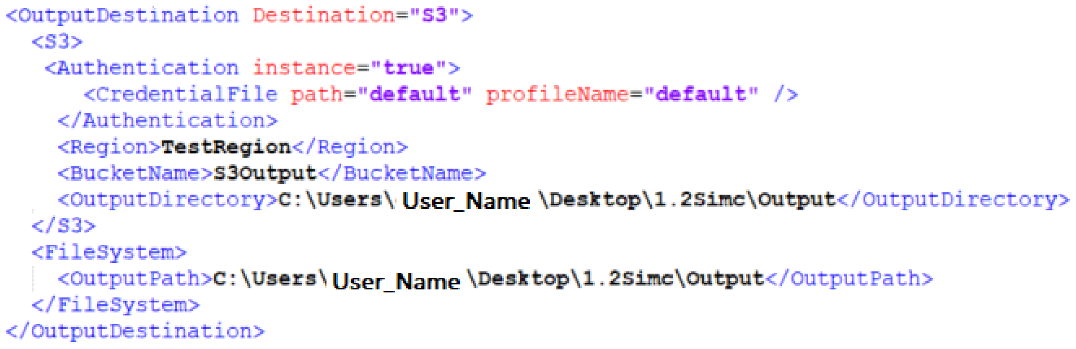
The following are the different conditions of output file destinations and the related loader configurations:

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



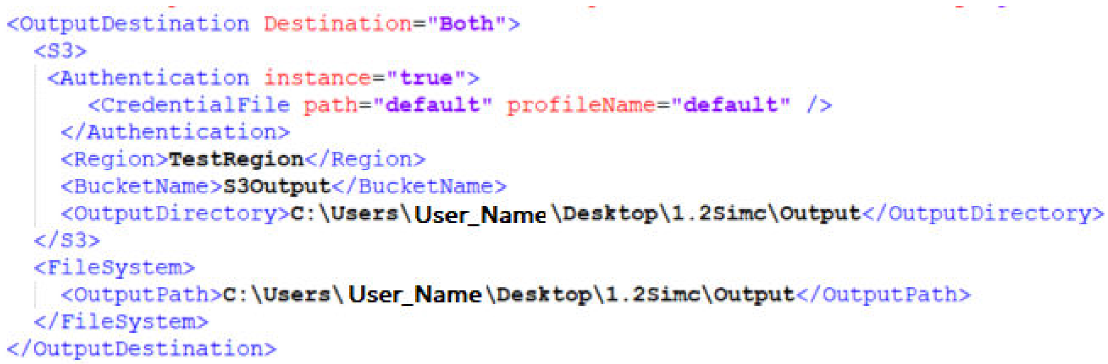
```
<OutputDestination Destination="FileSystem">
  <S3>
    <Authentication instance="true">
      <CredentialFile path="default" profileName="default" />
    </Authentication>
    <Region />
    <BucketName />
    <OutputDirectory />
  </S3>
  <FileSystem>
    <OutputPath>C:\Users\User_Name \Desktop\1.2Simc\Output</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 File System and ACS or ACS and S3 or all the three output destinations together.

Chapter 3 Run the Gateway from the Command Line

You can run the Gateway from the command line.

Note: When you run the Gateway from command line, all processing messages are directed to the log file. This allows you to use the command prompt in batch mode, 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 1D
Data\Logs\1DDataGateway_Log_<GUID>.txt
```

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

Note: A log file is generated for each input file.

At the command line, type the following:

```
"C:\Program Files\AVEVA\AVEVA NET Gateways\Gateway For 1D
Data\AVEVA.Gateways.Data1D.GUI.exe" <-cp> <-in> <-o> <-op> <-lp> <-extcp> <-trnsfcp> <-
ldrcp> <-context> <-ma> <-ma> <-timeout> <-sepproc>
```

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`".
`-op <output folder path>`	(optional) Output folder path for all the the output EIWM/CSV files.
`-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.
`-o <new output filename>`	(optional) Sets the new filename for the output EIWM file.
`-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.
`-context <project context>`	(optional) Sets the context for EIWM objects.
`-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.
`-timeout <timeout integer value>`	(optional) Sets the maximum allowed processing time (in minutes). Valid values: any positive integer, where a value of "0" means no timeout applies
`-sepproc < separate processing value>`	(optional) Sets whether ETL is a separate process to the configuration read and logging process. Valid value: "**true**" or "**false**".

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 the 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.
- Because the Gateway supports more than one type of extraction source, if the input path parameter contains only the folder name, then all files in that folder and its subfolders having the same type of extractor(s) configuration will be processed. For example, if the Excel and CSV extractors have been configured, then all file types .xlsx, .xls, and .csv will be processed.
- When `-sepproc true` is specified, the application will launch a child process for each input file or batch, improving isolation and fault tolerance. If invalid, an error is thrown.
- If a valid `-timeout` is set and reached, then
  - If `-sepproc` is **false**, the entire Gateway process stops and status is marked Timeout. If invalid, an error is thrown.
  - If `-sepproc` is **true**, the timeout applies to the child process that is executing the Extract, Transform and Load functions. This means that if processing a folder of files, the timeout applies to each file's processing, not to the entire folder's processing. For example, if `-timeout "1"` and `-sepproc "true"` are used, each file in the folder will be given 1 minute to complete in its own process.

Examples:

- With only project configuration file: `"C:\Program Files\AVEVA\AVEVA NET Gateways\Gateway For 1D Data\AVEVA.Gateways.Data1D.GUI.exe" -cp "D:\Test\new_1\new_1.xml"`
- With project configuration file and input file: `"C:\Program Files\AVEVA\AVEVA NET Gateways\Gateway For 1D Data\AVEVA.Gateways.Data1D.GUI.exe" -cp "D:\Test\new_1\new_1.xml" -in "D:Test2\Input\Excel\EqList.xls"`
- With project configuration file `new_1.xml,` input file EqList.pdf and the new output EIWM filename NewEqlist.pdf written to the output folder, defined in the configuration file: `"C:\Program Files\AVEVA\AVEVA NET Gateways\Gateway For 1D Data\AVEVA.Gateways.Data1D.GUI.exe" -cp "D: \Test\new_1\new_1.xml" -in "D:Test2\Input\PDF\EqList.pdf" -o "NewEqlist"`

- With project configuration file and output folder: `"C:\Program Files\AVEVA\AVEVA NET Gateways\Gateway For 1D Data\AVEVA.Gateways.Data1D.GUI.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 Tabular Data\AVEVA.Gateways.TabularData.GUI.exe" -cp "D:\Test\new_1\new_1.xml" -lp "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 1D Data\AVEVA.Gateways.Data1D.GUI.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 1D Data\AVEVA.Gateways.Data1D.GUI.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 1D Data\AVEVA.Gateways.Data1D.GUI.exe" -cp "D:\Test\new_1\new_1.xml" -ldrcp "D:\Test\new_2\Configurations\LoaderConfiguration.xml" Takes Loader Configuration file from command line parameter "D:\Test\new_1\new_1.xml" -extcp "D:\Test\new_2\Configurations\LoaderConfiguration.xml"`
- With project configuration file and project context: `"C:\Program Files\AVEVA\AVEVA NET Gateways\Gateway For 1D Data\AVEVA.Gateways.Data1D.GUI.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 1D Data\AVEVA.Gateways.Data1D.GUI.exe" -cp "D:\Test\new_1\new_1.xml" -ma "#MODEL_NAME#:My Output"`
- With project configuration file and timeout: `"C:\Program Files\AVEVA\AVEVA NET Gateways\Gateway For 1D Data\AVEVA.Gateways.Data1D.GUI.exe" -cp "D:\Test\new_1\new_1.xml" -timeout “1”`
- With project configuration file, timeout and separate process: `"C:\Program Files\AVEVA\AVEVA NET Gateways\Gateway For 1D Data\AVEVA.Gateways.Data1D.GUI.exe" -cp "D:\Test\new_1\new_1.xml" -timeout “1” -sepproc “true”`

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 or not. 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:

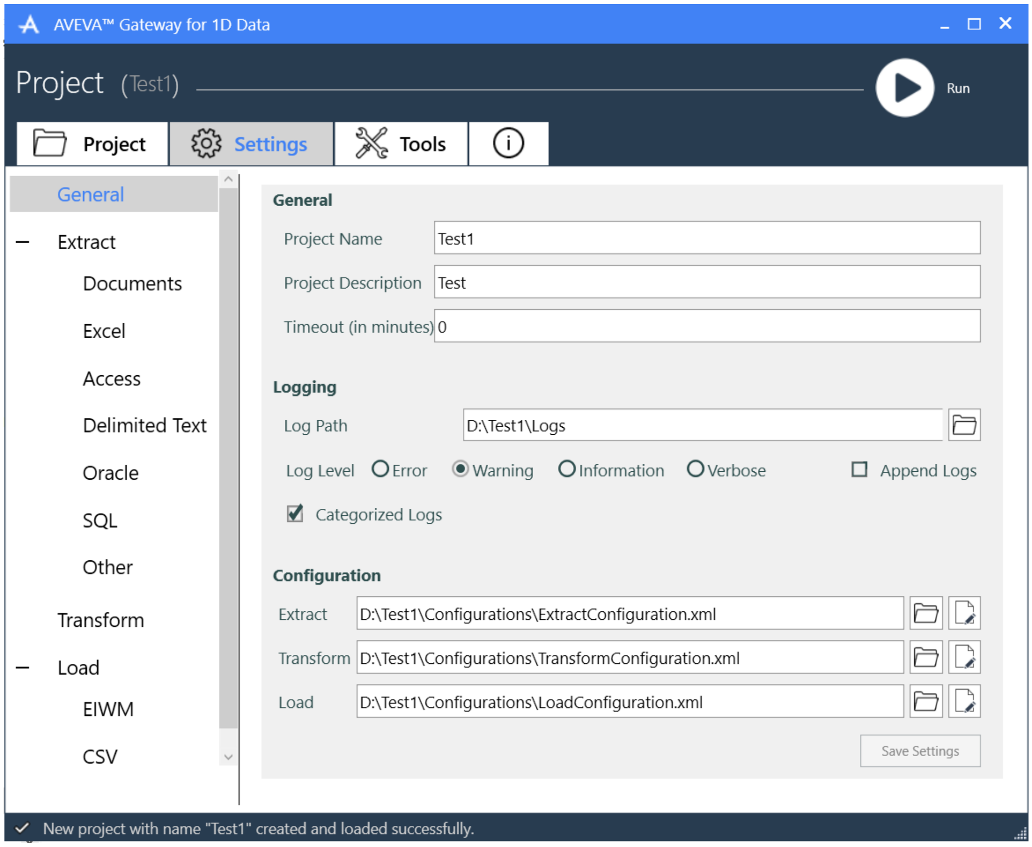
The following command line exit codes are defined for `AVEVA.Gateways.Data1D.GUI.exe`.

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
13	Processing failed in EIWM loader
17	Processing failed due to no input files
18	File processing stopped due to timeout
19	Processing failed in CSV loader
1000	Gateway processing failed

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

Chapter 4 Status Messages from the Gateway Processing

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



General

Project Name: Test1

Project Description: Test

Timeout (in minutes): 0

Logging

Log Path: D:\Test1\Logs

Log Level ○ Error ● Warning ○ Information ○ Verbose □ Append Logs

☑ Categorized Logs

Configuration

Extract: D:\Test1\Configurations\ExtractConfiguration.xml

Transform: D:\Test1\Configurations\TransformConfiguration.xml

Load: D:\Test1\Configurations\LoadConfiguration.xml

✓ New project with name "Test1" created and loaded successfully.

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

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

When set to true:

- A JSON report file is written in the same location as the log file. The level of detail in the JSON report file depends on the project settings except that messages with level 'Verbose' are never added to the JSON report file.
- If the Gateway process fails, then an additional JSON error report file containing details of the process and the error message is written in the same location, named "`<InputFileName>_Error.json`".

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

Field	Value Example	Description
`"Product"`	`"AVEVA Components Reporting Utility"`	Reporter program name.

Field	Value Example	Description
`"ReportingVersion"`	`"X.X.X.X"`	Reporter program version.
`"ReportedProduct"`	`"AVEVA Gateway for 1D Data"`	Gateway name.
`"ReportedProductVersion"`	`"Y.Y.Y.Y"`	Gateway version number.
`"StartDateTime"`	`"06-02-2020 19 06 43"`	Processing start time.
`"FinishDateTime"`	`"06-02-2020 19 06 44"`	Processing end time.
`"ElapsedTime"`	`"00 00 00 (0.97 seconds)"`	Time of the processing.
`"SourceName"`	`"Sample.dwg"`	Name of source data.
`"SuccessfulObjectsNumber"`	`1255`	Number of successfully processed objects.
`"ProcessingResult"`	`"Success"`	Overall result of the processing: Success/Failure/Timeout/Canceled
`"ProcessingResultNotes"`	`"Warnings"`	Additional note about the result of the processing.
`"ReturnCode"`	`0`	Numeric code returned from the processing
`"Configurations"`	...	List of configuration files.
`"OutputData"`	...	List of output files.
`"Options"`	...	Additional processing options.
`"Reports"`	...	List of messages (see the following note).

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

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

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

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

- "Failed in <module name>", if processing failed in specified module.

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

The JSON report then contains one or more status messages:

Field	Value Example	Description
"ID"	2	Number of the message.
"DateTime"	"06-02-2020 19 06 43"	Timestamp of the message.
"SubComponent"	"1D Data"	Gateway's component of the generated message.
"Severity"	"Warning"	Severity of the message: Debug/ Information/Warning/Error/ ErrorProcessingFailed/ErrorCritical.
"Code"	11644681	Unique ID of the message.
"CodeName"	"WAR_EXT_AUT_B1AF09"	Unique name of the message.
"Category"	"Extract"	Category of the message (see the following table.).
"Message"	"Could not write "document.csv"."	Text message.
"Remediation"	"Check that the output CSV file isn't open in another application."	Optional remediation.

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

Message Category	Message Description
Access	Issues connected with accessing provided location, file or database.
Break	Break of processing caused by a cancellation or a timeout.
Debug	Low level technical information describing processing details. Only for severity Debug.
Environment	Issue influenced by system environment such as not enough memory and read only file.
Extract	Problem with extracting some source data like wrong item or unrecognized API behavior.

Message Category	Message Description
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.
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 processing each file on the Gateway level. Therefore, it does not contain messages connected with issues, which may appear before the processing is 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 E Appendix A: Mapping Configuration

Mapping configuration files are generally defined in XML format. You can edit these files using 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.

Lookup Data Source

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

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

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

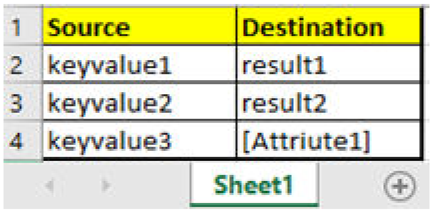
```
<MSExcelLookupDataSource
     id="Excel Map"
     file="C:\Mapping\Data Value Lookup\Mapping.xls"
     table="Sheet1"
     sourceColumn="[Source]"
     targetColumn="[Destination]" />
```

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

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

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

Sheet1 is constructed as follows:



	Source	Destination
1	Source	Destination
2	keyvalue1	result1
3	keyvalue2	result2
4	keyvalue3	[Attriute1]

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

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

Notes:

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

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

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

The following data sources are supported:

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

<MSAccessLookupDataSource
      id="MSAccessLookup"
      file="C:\Mapping\Data Value Lookup\Mapping1.accdb"
      query="SELECT [Source], [Destination] FROM [Table1]"
      provider="Microsoft Access Driver (*.mdb, *.accdb)"
      sourceColumn="Source"
      targetColumn="Destination" />
<MSExcelLookupDataSource
      id="ExcelLookup"
      file="C:\Mapping\Data Value Lookup\Mapping1.xls"
      provider="Microsoft Excel Driver (*.xls, *.xlsx, *.xlsm, *.xlsb)"
      extendedProperties="Excel 12.0; HDR=YES"
      query="SELECT [Source], [Destination] FROM [$Sheet2]"
      sourceColumn="Source"
      targetColumn="Destination" />

<MSSqlLookupDataSource
      id="MSSqlLookup"
      query="SELECT [Key], [Value] FROM [Table1]"
      connectionString="Driver={SQL Server};
      Server=serverxxx; Database=Databasexxx; UID=userxxx; PWD=pwdxxx;"
      sourceColumn="Key"
      targetColumn="Value" />
or
<MSSqlLookupDataSource
      id="MSSqlLookup"
connectionString="WZz2JoEigxXpWGjsUwdTAxjSG7N8rI61d+aHz503TrfeCNNcLPpmOg=="
      connectionStringEncrypted="true"
      query="SELECT [Key], [Value] FROM [Table1]"
      sourceColumn="Key"
      targetColumn="Value" />
<OracleLookupDataSource
      id="OracleLookup"
connectionString="WZz2JoEigxXpWGjsUwdTAxjSG7N8rI61d+aHz503TrfeCNNcLPpmOg=="
      connectionStringEncrypted="true"
      query="SELECT [Key], [Value] FROM [Table1]"
      sourceColumn="Key"
      targetColumn="Value" />
```

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

Notes:

- You can store 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 following Lookup data sources. If not specified, these defaults to the values shown below:

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

- Only one of the attributes, either **table** or **query**, can be used interchangeably in the Lookup data sources mentioned below. The values of attributes sourceColumn and targetColumn have to be column names from the mentioned table or query value.

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

- 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.

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

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, objects with `[max_pressure]` attribute present in the source will have the following characteristic in EIWM:

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

and 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="Template 2" />
  </Template>
</TemplateLookups>

<ObjectMappings regExTimeoutSecs="10">
  <Object>
    <Conditions>
      <Attribute name="ClassName" pattern="Door" />
    </Conditions>
    <TemplateID id =" template1" />
    <ObjectID value="[GlobalId]">
    </ObjectID>
  </Object>
</ObjectMappings>
```

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

Transforms

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

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

The following transformations of the source value are available:

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

Transformation	Description
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 expressions 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 specified as either `[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 Usage:

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

The 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>
```

The 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 when 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 Object attribute placeholders:

File Object Attribute Placeholder	**Value**
#TYPE#	Fixed value "manifest".
#MANIFEST#	Defines Extractor type, for example, Excel, Access, Delimited Text File, Oracle, SQL. If multiple data sources are processed, value would be "Tabular".
#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 file.
#INPUT_FOLDER#	Parent directory of the input file.
#MODEL_NAME#	Defines input file name for Excel, Access, Delimited Text File and "Oracle" and "Sql" for Oracle and SQL Server, respectively. If multiple data sources are processed, value would be "Tabular".
#KEEP_FOLDER_TREE#	TRUE, if input folder hierarchy is preserved in output generation.
#TARGET_LOCATION#	Output folder path.

Note: The following attributes should not be changed by the user:

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

- #KEEP_FOLDER_TREE#

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

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

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#".

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.

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# and #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 separate out the manifest object first from the other objects so that the common transformation applied for the other objects does not impact the manifest object:

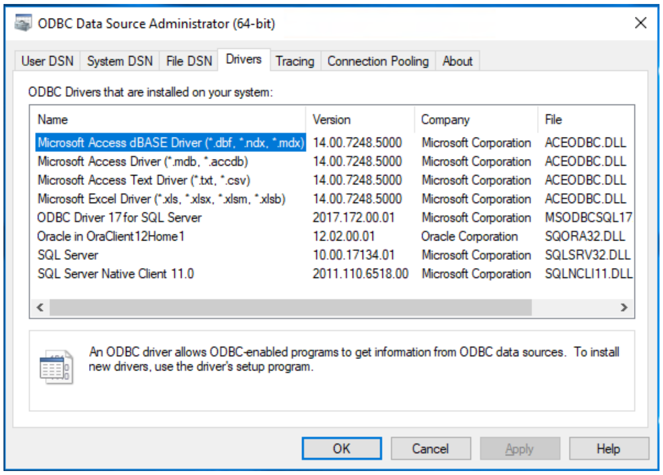
```
<ObjectMappings regExTimeoutSecs="10">
   <Object>
    <Conditions>
      <Attribute name="#TYPE#" pattern="manifest$" />
    </Conditions>
    <TemplateID id ="default" />
   </Object>
  <Object>
    <TemplateID id ="default" />
    <ObjectID value="[TAG]"/>
    <ClassID value ="[Classification]" />
   <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 must be used instead.

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

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



ODBC Drivers that are installed on your system:

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

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

Connection String Changes

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

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

The following are the connection string changes.

Excel:

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

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

This should be updated to support an ODBC driver:

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

Access:

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

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

This should be updated to support an ODBC driver:

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

CSV:

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

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

This should be updated to support an ODBC driver:

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

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

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

SQL Server:

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

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

This should be updated to support an ODBC driver:

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

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

Oracle:

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

```
<OracleLookupDataSource id="OracleLookup"
connectionString="Provider=OraOLEDB.Oracle;
```

```
Data Source=serverAddressxxx;
User ID=userxxx; password=passwordxxx;"
query="select Source, Destination from DocumentLookup"
sourceColumn="Source" targetColumn="Destination" />
```

This should be updated to support an ODBC driver:

```
<OracleLookupDataSource id="OracleLookup" connectionString="Driver={Oracle in
OraClient12Home1};
DBQ=serverAddressxxx;
UID=userxxx; PWD=passwordxxx;"
query="select Source, Destination from DocumentLookup"
sourceColumn="Source" targetColumn="Destination" />
```

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

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

Base Mapping

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

Object Mapping

The `<ObjectMappings>` node contains one or more <Object> nodes for identifying the mappings that should apply to specific objects extracted from the data source and stored in the Object Model.

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.

A generic object consists of a list of attributes that defines its characteristics. 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 or remove it.
- **Load mapping** settings to make the object ready for loading into AIM.

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_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_CSV#" />
      <Value value="custom csv name" />
    </Attribute>
    </Attributes>
</Object>
```

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

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

Attribute Mapping entries can be broadly classified into the following:

Create:

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

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

```
</Attribute>
```

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:

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

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

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

```
<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>
```

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:

```
<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>
```

Update:

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

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

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.

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

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.

```
<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.

```
<Attribute name="Tag" >
        <Remove />
</Attribute>
```

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.

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

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

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

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

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

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

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 although their mapping or the value of `keepUnmappedAttributes` attribute are 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 in any 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. To prevent this, ensure that all referenced files are not located in the input root folder or its sub-folders or process 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 1D Data handle: "handle"
- Attribute name for 1D Data parent handle: "parent handle"
- Attribute name for 1D Data layer: "layer"
- Attribute name for 1D Data block name: "block name"
- Attribute name for 1D Data object's type: "1DData type"
- Attribute name for main and each referenced ("XRefs") 1D Data 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.

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 the same 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.

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 `ObjectID`, 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.

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

Note: The `<MultiValue>` tag that designates the `''part_codes''` attribute as being a multi-value attribute whose values are separated with a semi-colon. The `<MultiValue>` tag is supported only 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.

```
<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>
```

Notes:

The `keepUnmappedAssociations` setting is **optional**. By default, the `keepUnmappedAssociations` attribute value is **true** if the setting is not provided.

- Associations that have types of `is referenced in` and `is identified by`, are considered as mandatory **and will not be removed from objects while creating output files when** `keepUnmappedAssociations` **is set to false**.
- The following Object mapping can be used for Associations of a specific `Type` to be exported to the EIWM file for all objects.

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

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

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

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

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

Expand and Interpolate:

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

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

Notes:

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

Examples:

```
   <Association attribute="object_name" attributePattern="^[A-Z]{4}$" >
   <Type value="is part of" />
   <TargetID value="P-101A/D" >
   <Expand interpolate = "false">
   <Pattern>[A-Z]/[A-Z]</Pattern>
        <Char>/</Char>
      </Expand>
```

```
  </TargetID>
</Association>
```

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

```
<Association attribute="object_name" attributePattern="^[A-Z]{4}$" >
 <Type value="is part of" />
 <TargetID value="P-101A/D" >
 <Expand interpolate = "true">
  <Pattern>[A-Z]/[A-Z]</Pattern>
  <Char>/</Char>
 </Expand>
 </TargetID>
</Association>
```

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

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

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

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

In the above example, the first `Expand` block creates the following tags `"07001-QET-003"`,`"07001-QFT-003"`,`"07001-QGT-003"`,`"07001-FE-004A/B"` and `"07001-FE-004A/C"`.

The second `Expand` block further splits tags `"07001-FE-004A/B"` and `"07001-FE-004A/C"` into `"07001-FE-004A"`, `"07001-FE-004B"` and `"07001-FE-004C"`.

Finally, six tags named "07001-QET-003", "07001-QFT-003", "07001-QGT-003", "07001-FE-004A", "07001-FE-004B" and "07001-FE-004C" are created.

Association Mapping entries can be broadly classified into the following:

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

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

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

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

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

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

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

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

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

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

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

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

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

Load Mapping Settings:

Load mappings settings contain the mappings that can be applied to a generic object during transformation phase.

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"`.

ObjectID Mapping

ObjectID 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 `ObjectID` 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 `ObjectIDs`.

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.
```

Expand Interpolate:

The Expand feature lets you create multiple objects in the EIWM output from an attribute value containing text delimited with a separator (such as comma or semi-colon). You can use any number of Expands to build the tags incrementally. A separate object is created in the output for each expanded tag and the value is used as the ID of that object.

Note: The Expand mapping pattern must contain the expansion character to consider the attribute value for the expansion.

Examples:

```
<Object>
```

```
  <Conditions>
   <Attribute name="object_class" pattern="Door" />
  </Conditions>
  <ObjectID value="P-101A/D">
   <Expand interpolate = "false">
    <Pattern>[A-Z]/[A-Z]</Pattern>
    <Char>-</Char>
   </Expand>
</ObjectID>
</Object>
```

`[A-Z]/[A-Z]` in the above example helps create two tags named `P- 101A` and `P-101D` as the interpolate setting is set to `false`.

```
<Object>
 <Conditions>
  <Attribute name="object_class" pattern="Door"/>
 </Conditions>
 <ObjectID value="P-101A/D">
 <Expand interpolate = "true">
  <Pattern>[A-Z]/[A-Z]</Pattern>
  <Char>-</Char>
 </Expand>
</ObjectID>
</Object>
```

`[A-Z]/[A-Z]` in the above example helps create four tags named `P- 101A`, `P- 101B`, `P- 101C` and `P-101D` as the interpolate setting is set to `true`.

```
<Object>
 <Conditions>
  <Attribute name="object_class" pattern="Door"/>
 </Conditions>
 <ObjectID value="ABA-D1/3">
  <Expand interpolate = "false">
   <Pattern>[A-Z]-[A-Z]</Pattern>
   <Char>-</Char>
  </Expand>
  <Expand interpolate = "true">
   <Pattern>[0-9]/[0-9]</Pattern>
   <Char>/</Char>
  </Expand>
</ObjectID>
</Object>
```

In the above example, tags named `"ABA1"`, `"ABA2"`, `"ABA3"`, `"ABD1"`, `"ABD2"` and `"ABD3"` are created. The first Expand block creates the following tags `"ABA1/3"` and `"ABD1/3"`. The second Expand block further splits tags `"ABA1/3"` into `"ABA1"`, `"ABA2"` and `"ABA3"` and `"ABD1/3"` into `"ABD1"`, `"ABD2"` and `"ABD3"`. Finally, six tags named `"ABA1"`, `"ABA2"`, `"ABA3"`, `"ABD1"`, `"ABD2"` and `"ABD3"` are created.

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 `ClassID`.

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 ClassID 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:

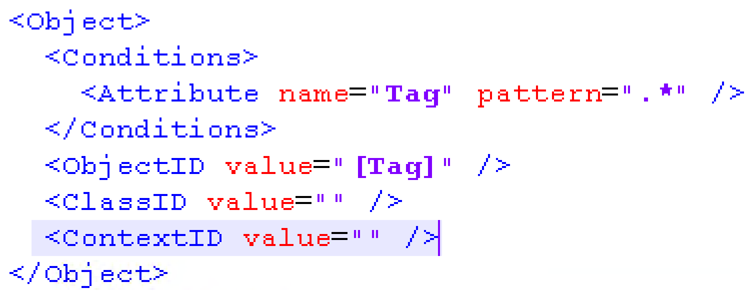
- The prefix "`associated`:" is used when referring to the associated object's attribute.
- All conditions must be met for the `ClassID` mapping to be applied.

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

Often when importing tag data into AIM the tag's ClassID is already defined. In the case where this shouldn't be changed, you should ensure that the EIWM doesn't contain any definition of ClassID for the relevant objects. This can be achieved in base mapping by the following methods:

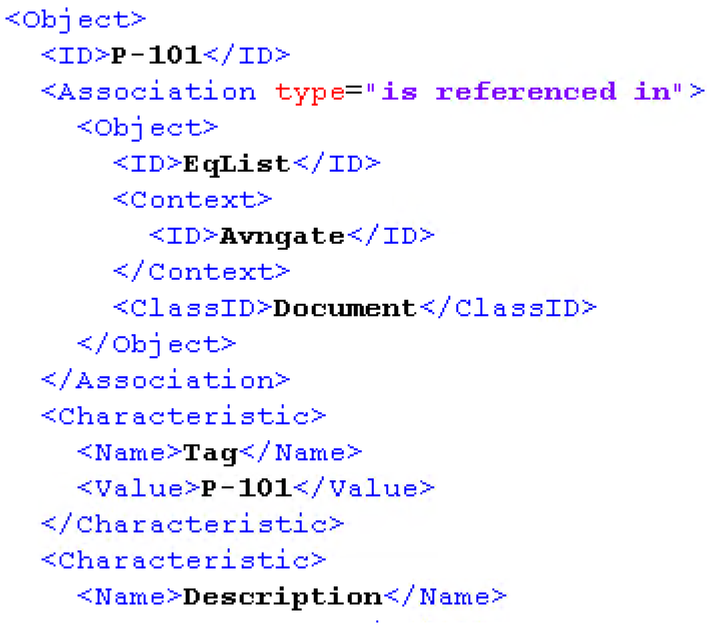
- `<ClassID value="" />`
- `<ClassID value=" " />` (Space)
- `<ClassID \>` node not provided in Base mapping

For example, using the following base mapping:



```
<Object>
  <Conditions>
    <Attribute name="Tag" pattern=".*" />
  </Conditions>
  <ObjectID value="[Tag]" />
  <ClassID value="" />
  <ContextID value="" />
</Object>
```

Will produce EIWM output without `ClassID` (nor `ContextID`) defined:



```
<Object>
  <ID>P-101</ID>
  <Association type="is referenced in">
    <Object>
      <ID>EqList</ID>
      <Context>
        <ID>Avngate</ID>
      </Context>
      <ClassID>Document</ClassID>
    </Object>
  </Association>
  <Characteristic>
    <Name>Tag</Name>
    <Value>P-101</Value>
  </Characteristic>
  <Characteristic>
    <Name>Description</Name>
```

ContextID Mapping

ContextID Mapping determines the ID of the context within which an `ObjectID` 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.

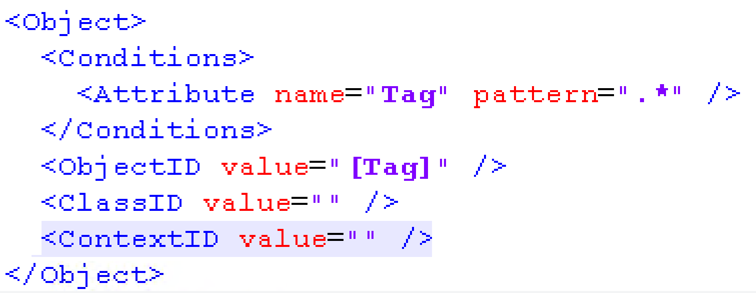
If an object does not match any of the mapping entries, it is not assigned any context. To set a default context that is used if none of the other mapping entries match, an entry with no qualifiers should be created at the bottom of the mapping file:

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

Often when importing tag data into AIM the tag's ContextID is already defined. In the case where this shouldn't be changed, you should ensure that the EIWM doesn't contain any definition of ContextID for the relevant objects. This can be achieved in base mapping by the following methods:

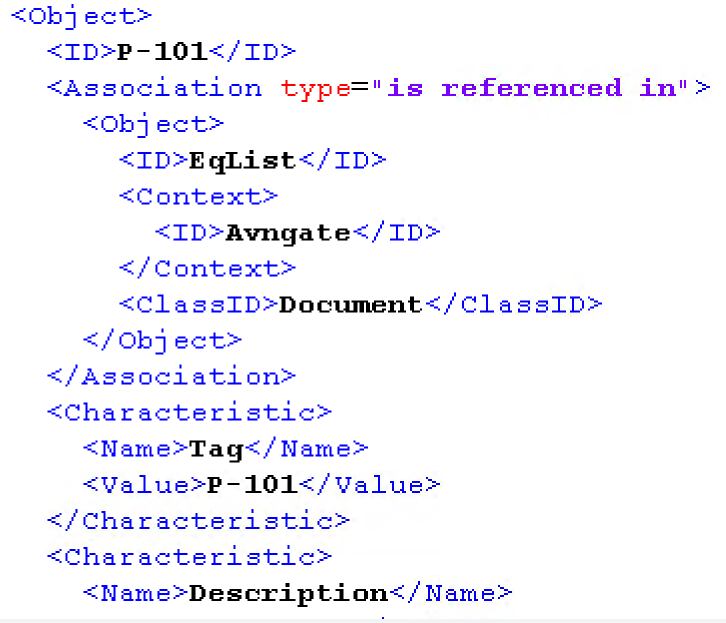
- `<ContextID value="" />`
- `<ContextID value=" " />` (Space)

For example, using the following base mapping:



```
<Object>
  <Conditions>
    <Attribute name="Tag" pattern=".*" />
  </Conditions>
  <ObjectID value="[Tag]" />
  <ClassID value="" />
  <ContextID value="" />
</Object>
```

Will produce EIWM output without ContextID (nor ClassID) defined:



```
<Object>
  <ID>P-101</ID>
  <Association type="is referenced in">
    <Object>
      <ID>EqList</ID>
      <Context>
        <ID>Avngate</ID>
      </Context>
      <ClassID>Document</ClassID>
    </Object>
  </Association>
  <Characteristic>
    <Name>Tag</Name>
    <Value>P-101</Value>
  </Characteristic>
  <Characteristic>
    <Name>Description</Name>
```

Note: If the EIWM loader has Project Context set to Avngate and **override is set to** true then in each of the above cases the ContextID will be set to Avngate in the EIWM.

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="DIG" />
    <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="DIG" />
    <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

Normally ObjectID’s are sourced from attribute values extracted per object. But when the input data has no attributes, such as is often the case for Documents, the task is to identify a number of ObjectIDs or tag values from a large string of text. An alternative method to do this efficiently is to use **Search Criteria Mapping**, which 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.

After a text matches a pattern, the relevant tag is created and then no other patterns in the mapping sequence of a search block will be applied to that tag’s text. If multiple Search blocks are configured within the

SearchCriteria, then all unmatched parts of the text will be subject to pattern mapping of the subsequent Search blocks.

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

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

If two Patterns are defined:

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

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

If the order of the Patterns is reversed:

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

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

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

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

Therefore, 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.

The identified tags will then 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.

You can use the Transform function to modify the value of a matched pattern:

```
<SearchCriteria attribute = “Content”>

<Search>
```

```
<Patterns createAlias=”false”>
<Pattern value = “[A-Z]-\d{3}[A-Z]"/>
<Transforms>
<Keep pattern=”\d{3}”/>
<Compose delimiters=”-“ constructor=”type: {1}, series: {2}”/>
<Remove pattern=”^[A-Z]{1}”/>
<LowerCase/>
<InsertBefore pattern=”^.*$” value=”InsertBefore”/>
<Replace pattern=”\d{3}” value=”619”/>
<UpperCase/>
</Transforms>
</Patterns>
</Search>
</SearchCriteria>
```

When a Transform node is within a Patterns node, then it supports an optional parameter createAlias, default value of “false”. When createAlias = “true” it will modify the value of the matched pattern and also create an alias to it in the EIWM objects, as shown below:

```
<SearchCriteria attribute = “Content”>
<Search>
<Patterns createAlias=”true”>
<Pattern value = “[A-Z]-\d{3}"/>
<Transforms>
<Keep pattern=”\d{3}”/>
</Transforms>
</Patterns>
</Search>
</SearchCriteria>
```

An additional association "is identified by" is added to the newly created objects from SearchCriteria.

When the Patterns node doesn't have a Transform node, even if the createAlias attribute is set to true, no additional associations are added to the newly created objects from SearchCriteria.

Expand Attributes Mapping

Expand Attributes mapping enables you to treat a string in an attribute as a comma-separated list with syntax to control the separation character and refer to each item in the list by its index value when assigning its value to specific attributes.

The ExpandAttributes requires you to specify the name of the attribute containing the string list. Optional parameters control the separator and refers to a specific tableIndex and headerRow.

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.

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

In this case, the separator does not need to be defined as the default value is comma.

Note: The Attribute section in the ExpandAttributes mapping is different from the Attribute section in the Attributes mapping, see Attribute Mapping, so the functionalities of Attribute in ExpandAttributes mapping will not be the same as in Attribute mapping.

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, Value and Units elements are taken directly from the index positions 1-3 of the expanded string to create a Property with Name = "Pump", Value = "10" and Units = "kPa" in the output EIWM.

New row objects can also be created from a table that has a header index such that the attributes of each object have Names derived from the relevant header row's indexed values. For example, if the input has two tables:

Table: Plant Local to Cambridge Region

Identifier	Area	Type	Number	Batch
P-101	Cambridge	Hydo	1	D-5670
Valve 26	Histon	Hydro	1	

Table: Parts Needed for Hydro Pumps

Part ID	Name	Material	Size
B_ST_8	Bolt	Stainless Steel	8 mm
G_BN_10	Gasket	Black Nylon	20 diam inner, 30 outer

The optional tableIndex and headerRow parameters are used to set which table and header rows should be used to generate the attributes for each of the non-header rows in a table. To access the headerRow's values you need to specify the headerIndex:

You can also set all the values to create a new attribute, although in this case the Content attribute becomes irrelevant. 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.

The optional tableIndex and headerRow parameters are used to set which table and header rows should be used to generate the attributes for each of the non-header rows in a table. To access the headerRow's values, you need to specify the headerIndex:

```
<ExpandAttributes name = "Content" separator = ":" tableIndex="1" headerRow="1" />
    <Attribute>
          <Name headerIndex="1" />
          <Value index="1" />
    </Attribute>
    <Attribute>
          <Name headerIndex="2" />
          <Value index="2" />
    </Attribute>
<ExpandAttributes />
<ObjectID value="[Identifier]" />
```

The value of Name element is taken from the index position 1 of the Content attribute value of table 1 and Value from the index position 2 of the headerRow object of table 1. The ObjectID is set to the value of the Identifier attribute previously created in the ExpandAttributes.

The above mapping will result in new characteristics to be created for each non-header row of table 1:

```
 <Object>
     <ID> P-101 </ID>
    <Characteristic>
        <Name> Identifier </Name>
        <Value> P-101 </Value>
      </Characteristic>
     <Characteristic>
        <Name> Area </Name>
        <Value> Cambridge </Value>
      </Characteristic>
</Object>
<Object>
     <ID> Valve 26 </ID>
    <Characteristic>
        <Name> Identifier </Name>
        <Value> Valve 26 </Value>
      </Characteristic>
     <Characteristic>
        <Name> Area </Name>
        <Value> Histon </Value>
      </Characteristic>
</Object>
```

More than one table in a page can be processed by either specifying the relevant `tableIndex` values:

```
<ExpandAttributes name = "Content" separator = ":" tableIndex="1, 2" headerRow="1" />
```

Or choosing all the tables:

```
<ExpandAttributes name = "Content" separator = ":" tableIndex="all" headerRow="1" />
```

Either index or `headerIndex` attributes can be used with Name, Value and Units element. Both `index` and `headerIndex` cannot be used together.

Either value or index attribute can be used with `Name`, `Value` and `Units` element. Both index and value cannot be used together.

Limitation:

The `Content` attribute should have at least two substring elements for the `ExpandAttribute` mapping to be applied. This is also applicable in case of `headerRow` content as well.

Extracting Table Data from PDF Documents

When a PDF document contains data in a table format the rows of the table can be extracted as row objects with attributes named according to the header values in the table by:

- Selecting to Extract Contents by Word, see Extract Settings.
- Using the Object Merge Mapping Extension to convert the words in each row into row objects that have an attribute Content containing a comma separated list of the row values, see .Object Merge Mapping
- Using Expand Attributes Mapping to identify the relevant table and header row and for each row object create attributes named from the header row cells and the values provided in the row cells.

Object Merge Mapping

PDF documents often contain tabular-like data where related information is displayed in multiple rows or columns. This therefore requires the text objects to be extracted as words with Xmin and Ymin positions, see Extract Settings to Select Extract contents By Word. The `<MergeCriteria>` function may then be used to combine related elements into row or column-like objects based on the X, Y positions of each element on the page. Its configuration defines the criteria used to determine how elements are merged in the mapping process. It contains the following key elements:

- `Tolerance`
- `MaxExtentX`
- `MaxExtentY`
- `Separator`

Each of these elements controls a specific aspect of the merge logic, particularly for spatial or geometric data.

Element Definitions

- `<Tolerance>`

Specifies the allowable tolerance (usually in units such as pixels or millimeters) when comparing object positions or attributes during the merge process. Objects within this tolerance are considered equivalent for merging purposes.

ListType property determines how objects are grouped for merging—typically by "Row" or "Column". The Tolerance property is used to specify how close two values (such as coordinates) must be to be considered part of the same group.

Interpretation:

- ListType = "Row"**:**

  Tolerance is usually applied to the Y-coordinate (vertical position).

  Objects are grouped as being in the same "row" if their Y values differ by less than or equal to the specified Tolerance.

  Example: If Tolerance = 2, objects with Y-coordinates within 2 units are grouped together.

- ListType = "Column":

  Tolerance is usually applied to the X-coordinate (horizontal position).

  Objects are grouped as being in the same "column" if their X values differ by less than or equal to the specified Tolerance.

  Example: If Tolerance = 2, objects with X-coordinates within 2 units are grouped together.

  Tolerance is measured from either YMin or XMin, depending on the ListType specified in your MergeCriteriaType:

  - If ListType is "**Row**", tolerance is measured from the YMin attribute of each object.
  - If ListType is "**Column**", tolerance is measured from the XMin attribute of each object.

- <MaxExtentX>

  Defines the maximum allowed extent (width) in the X direction for the objects being merged. This is typically used to filter or limit the merge operation to objects that do not exceed this width.

- <MaxExtentY>

  Defines the maximum allowed extent (height) in the Y direction for the objects being merged. This is typically used to filter or limit the merge operation to objects that do not exceed this height.

- <Separator>

  Specifies the string used to separate multiple values when merging attributes or concatenating data fields. This is useful when combining lists or sets of values into a single string.

Example Usage

When a PDF document has spatially related text strings extracted as Words, such as items in a table's row, you can merge them into a single object based on their X and Y positions. The Gateway identifies the text strings that align in a row (or column) if their Ymin (or Xmin) position is within the tolerance value of the first string's Ymin (or Xmin) value. If a string's Xmin position is within the MaxExtentX value of the first string's Xmax (first string Xmax + MaxExtentX >= second string Xmin), then the two strings will be concatenated into one string separated by a single space. Otherwise, the next string in the X (or Y) direction of the row (or column) will be treated as the next element in the list, and concatenated with the first string but with the insertion of a defined separator.

Configuration:

```
<Object >
 <MergeCriteria>
  <ListType Value="Row" />
  <Tolerance Value="2" />
 <MaxExtentX Value="8" />
  <Separator Value=":" />
 </MergeCriteria>
</Object>
```

Notes:

- Irrespective of the `ListType` configuration (Row/Column), the `MaxExtentX` value only applies in the X direction.
- Blank values are not considered.
- `ListType` values are case insensitive.

Example:

Input text is as follows:

Identifier	Area	Type	Number	Batch
P-101	Cambridge	Hydro	1	Batch 1
Valve 26	Histon	Hydro	1	

When the above configuration is applied, the Gateway will merge all objects on each row as `ListType = "Row"`, where each concatenated value is separated by " : ".

Note: The objects in the last row are extracted with their X and Y extents as:

- Valve (Xmin 5, Xmax 10, Ymin 6, Ymax 6)
- 26 (Xmin 12, Xmax 14, Ymin 6, Ymax 6)
- Histon (Xmin 24, Xmax 30, Ymin 6.2, Ymax 6.2)
- Hydro (Xmin 40, Xmax 45, Ymin 5.8, Ymax 5.8)
- 1 (Xmin 55, Xmax 56, Ymin 6, Ymax 6)

The "Valve" and "26" objects are separated with a single space as both are within the radius of the `MaxExtentX` value.

The separate string objects are replaced by 3 new list objects with attribute values of:

Object 1 attribute value = "Identifier : Area : Type : Number : Batch"

Object 2 attribute value = "P-101 : Cambridge : Hydro : 1 : Batch 1"

Object 3 attribute value = "Valve 26 : Histon : Hydro : 1 "

Processing Multiple Tables in a Page

After texts are aligned in a row then TableIndex (1 ... N) and RowIndex (1 ... M) attributes will be added to each Row object. The TableIndex value is set to 1 for the first row in a Page. The `MaxExtentY` value is used to determine which rows belong to which table. If the difference in Y position between rows is less than `MaxExtentY`, then the subsequent row is assigned to the current table and RowIndex for that table is incremented by 1. If the difference is greater than `MaxExtentY`, then a new table is created (the TableIndex value will be incremented) for the subsequent row, and RowIndex for this new table will be reset to 1.

Configuration:

```
<Object >
 <MergeCriteria>
   <ListType Value="Row" />
   <Tolerance Value="2" />
    <MaxExtentX Value="8" />
    <MaxExtentY Value="20" />
   <Separator Value=":" />
 </MergeCriteria>
</Object>
```

For example, if the input text looks like:

Identifier	**Area**	**Type**	**Number**	**Batch**
P-101	Cambridge	Hydro	1	Batch 1
Valve 26	Histon	Hydro	1	

Part ID	**Name**	**Material**	**Size**
Part 1	Bolt	Stainless Steel	8 mm
Part 2	Gasket Black	Nylon	20 diam inner, 30 outer

The result of Object Merge mapping will be 6 row objects in two tables with concatenated lists in their Content attribute:

Row object 1: TableIndex=1, RowIndex=1, Content="Identifier : Area : Type : Number : Batch"

Row object 2: TableIndex=1, RowIndex=2, Content="P-101 : Cambridge : Hydro : 1 : Batch 1"

Row object 3: TableIndex=1, RowIndex=3, Content="Valve 26 : Histon : Hydro : 1 "

Row object 4: TableIndex=2, RowIndex=1, Content="Part ID : Name : Material : Size"

Row object 5: TableIndex=2, RowIndex=2, Content="Part 1 : Bolt : Stainless Steel : 8 mm"

Row object 6: TableIndex=2, RowIndex=3, Content="Part 2 : Gasket : Black Nylon : 20 diam inner, 30 outer"

Below is another example of how to use these elements within a `<MergeCriteria>` block in your XML configuration when texts in a cell are in different rows:

```
<MergeCriteria>
  <ListType Value="ROW"/>
  <Tolerance Value="5"/>
  <MaxExtentX Value="100"/>
  <MaxExtentY Value="50"/>
  <Separator Value=";"/>
</MergeCriteria>
```

Explanation:

- `ListType Value="ROW"`: Specifies the type of list (for example, `"ROW"` or `"COLUMN"`).

- Tolerance Value="5": Objects within 5 units are considered for merging.
- MaxExtentX Value="100": Only objects with a width up to 100 units are merged.
- MaxExtentY Value="50": Only objects with a height up to 50 units are merged.
- Separator Value=";": When merging multiple attribute values, they are joined using a semicolon.

Example:

Tag	Description	Classification	Discipline	Attribute
P-101	Pump 101	Pump	Process Management	TestAttribute1
P-102	Pump 102	Pump	Process Controlling	TestAttribute2
P-103	Pump 103	Pump	Process Alignment	TestAttribute3

If we use the below values in **ObjectMergeMapping.xml**:

```
    <MergeCriteria>
      <ListType Value="Row" />
      <Tolerance Value="15" />
      <MaxExtentX Value="20" />
      <MaxExtentY Value="10" />
      <Separator Value="," />
    </MergeCriteria>
```

Here, ListType="Row" means objects are grouped by their YMin value (vertical position).

Tolerance=15 means any objects whose YMin values are within 15 units of each other are considered to be in the same row group.

What happens:

The YMin of each object is checked. If two objects have YMin values that differ by 15 or less, they are grouped together.

MaxExtentX=20: When merging horizontally, objects are merged if their XMin/XMax values are within 20 units.

Objects in each row group by XMin (left to right) are sorted. It then checks if the next object's XMin is within MaxExtentX of the current object's XMax. If so, they are merged into a single object.

The "**Discipline**" column for P-101 is "Process Management" (split across two cells/objects: "Process" and "Management").

- Similarly, for P-102: "Process Controlling" (split as "Process" and "Controlling").
- For P-103: "Process Alignment" (split as "Process" and "Alignment").

Merging logic:

All objects in the same row (YMin within 15 units) are grouped.

"Process" and "Management" are close enough horizontally (XMin/XMax within 20 units), so they are merged into one object: "Process Management".

The same happens for "Process" + "Controlling" and "Process" + "Alignment".

"Process" and "Management" become a single merged object: "Process Management".

"Process" and "Controlling" become "Process Controlling".

"Process" and "Alignment" become "Process Alignment".

The tolerance allows for small misalignments in Y (row grouping).

The max extent in X allows for merging adjacent words/cells that are part of the same logical value but may have been split during extraction.

Key Points:

- YMin + Tolerance: Groups objects into rows.
- XMin/XMax + MaxExtentX: Merges horizontally adjacent objects in the same row.
- Result: Split words/cells are combined into the intended full value.

Let’s consider the same example with ListType="Column":

```
<MergeCriteria>
 <ListType Value="Column" />
 <Tolerance Value="15" />
 <MaxExtentX Value="20" />
 <MaxExtentY Value="34" />
 <Separator Value="," />
</MergeCriteria>
```

ListType="Column": Objects are grouped by their XMin value (horizontal position).

Tolerance=15: All objects whose XMin values are within 15 units are grouped as a single column.

This allows for small misalignments in the X direction. All objects that are visually in the same column (even if slightly misaligned) are grouped together.

Within each column group, objects are sorted by YMin (top to bottom).

MaxExtentY=34: Objects whose YMin/YMax values are within 34 units vertically are merged together.

This merges vertically adjacent fragments in the same column.

The "Discipline" column for P-102 and P-103 is split into two objects each:
"Process" , "Management" for P-101

"Process" , "Controlling" for P-102

"Process" , "Alignment" for P-103

All "Process", "Management ", "Controlling", and "Alignment" objects in the same column (XMin within 15 units) are grouped.

Within that column, the code checks the YMin/YMax of each object:

"Process" and "Controlling" are within 34 units vertically, they are merged as "Process Controlling".

"Process" and "Alignment" are within 34 units, they are merged as "Process Alignment".

The split values are merged into single objects: "Process Controlling" for P-102 and

"Process Alignment" for P-103.

Appendix F 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.

CSV report can be configured and generated in the following two scenarios:

- Reporting during transformation configured in Transform configuration file:

  This feature generates the CSV report during transformation and customizes the type of the information written to the CSV file. 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:ContainedInStructure"
 />
</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>
```

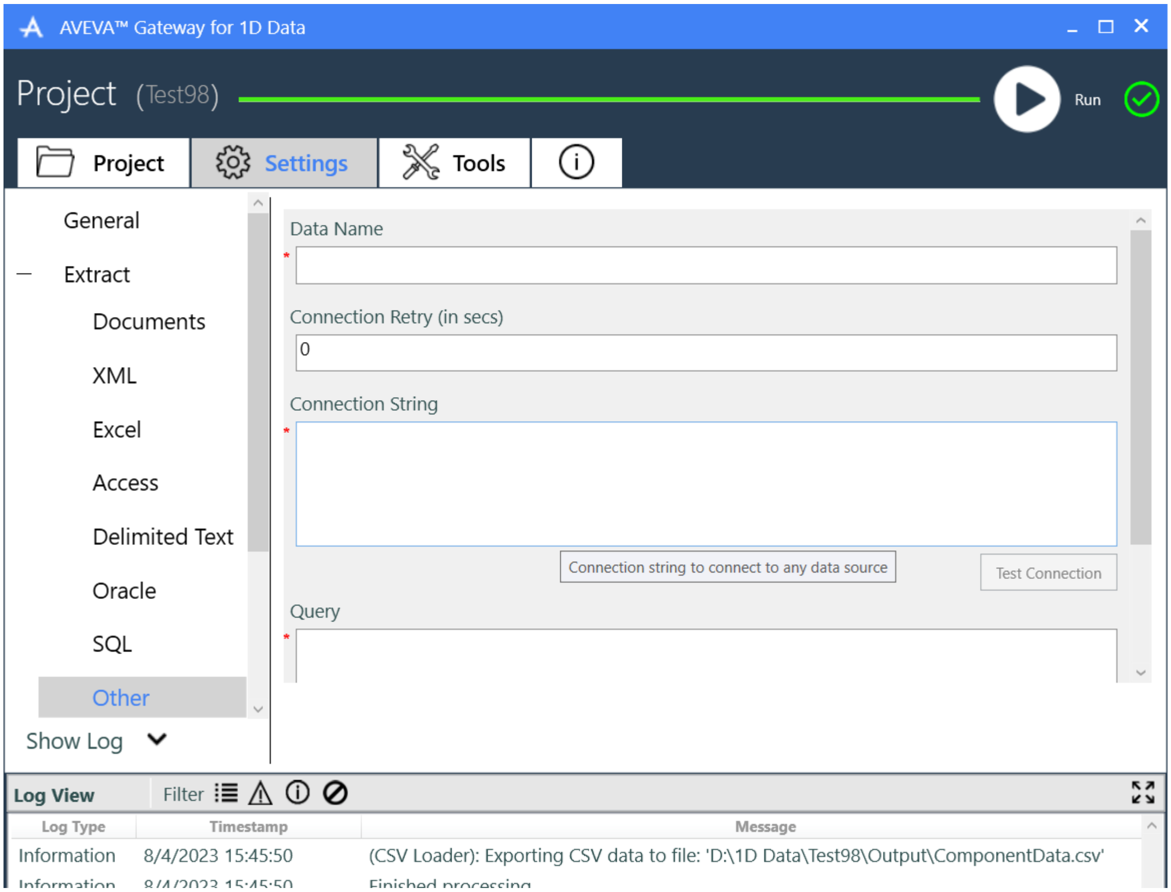
```
</extension>
```

Note: The optional 'annotations' feature 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.

- 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 Engineering Tab under Load:



Data Name

Connection Retry (in secs)
0

Connection String

Query

```
Log Type      Timestamp            Message
Information   8/4/2023 15:45:50    (CSV Loader): Exporting CSV data to file: 'D:\1D Data\Test98\Output\ComponentData.csv'
Information   8/4/2023 15:45:50    Finished processing
```

Settings:

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

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

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

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

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

- `suffix = "<file name suffix>`: (**Optional**) Defines a suffix to add to the CSV file name. It is used only in Transform configuration. It is ignored when `outputFilePath` is set.
- `outputFolder = "<output folder path>"`: (**Optional**) Stores the CSV files in the location indicated by this setting. Otherwise, files are passed to the output folder set in Load settings. It is used only in Transform configuration. It is ignored when `outputFilePath` is set.
- `addHeaderRow = "<true/false>"`(in **Generate Headers in CSV** checkbox): (Optional) Sets to "**True**" to write the names of the attributes or associations to the first row of the CSV file. This setting is enabled by default.
- `append = "<true/valse>"`: (Optional) Adds current CSV data to file set by **ouputFilePath**. When option **addHeaderRow** is enabled, then file to append must contain columns with the same names in the header

row. If **addHeaderRow** is disabled, then number of the columns in the file must be the same as in the data to append.

In append mode, user cannot provide generic types of columns like #Attributes or #Associations because they may deliver different list of attributes and associations between files. In append mode, list of columns for all processed files is constant.

- `outputFilePath = “<output file path>”`: This attribute must be set when append mode is enabled. It can be provided absolute or path relative to Transform configuration location. When **outputFilePath** is set then attributes: **outputFolder** and suffix are ignored. This setting can be also used with disabled append mode.

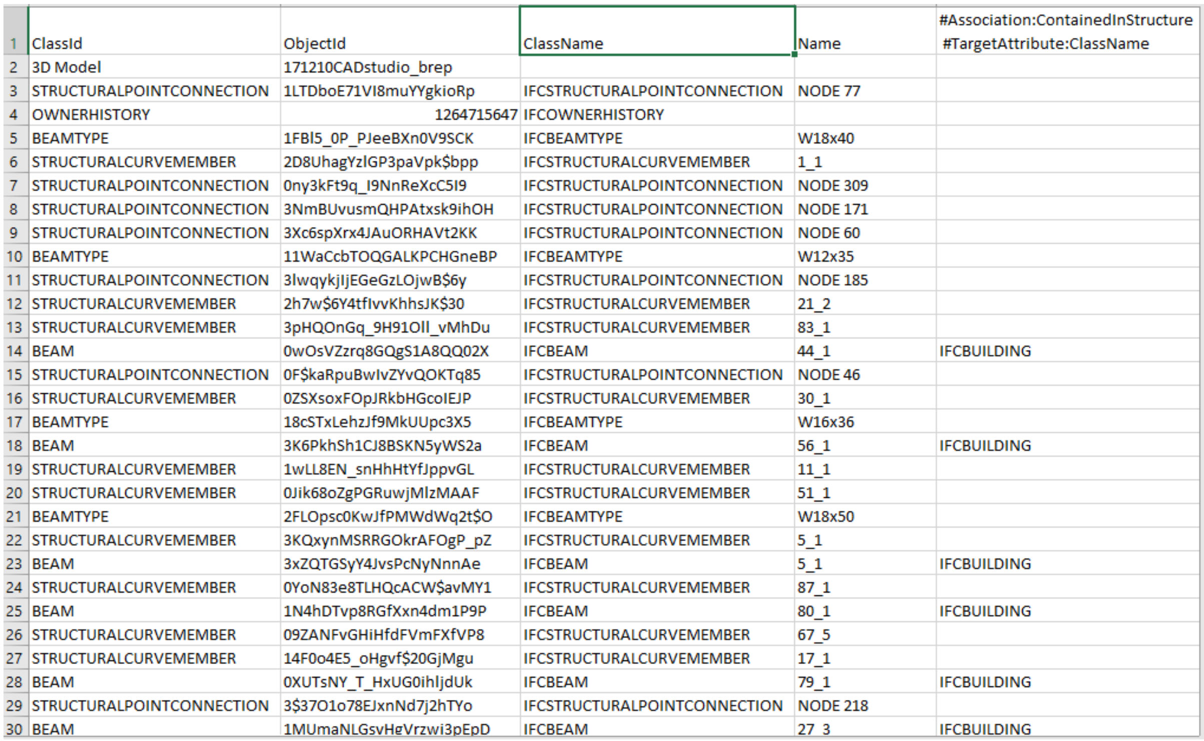
Tracking attributes names which can be used with keyword #Tracking:

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

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

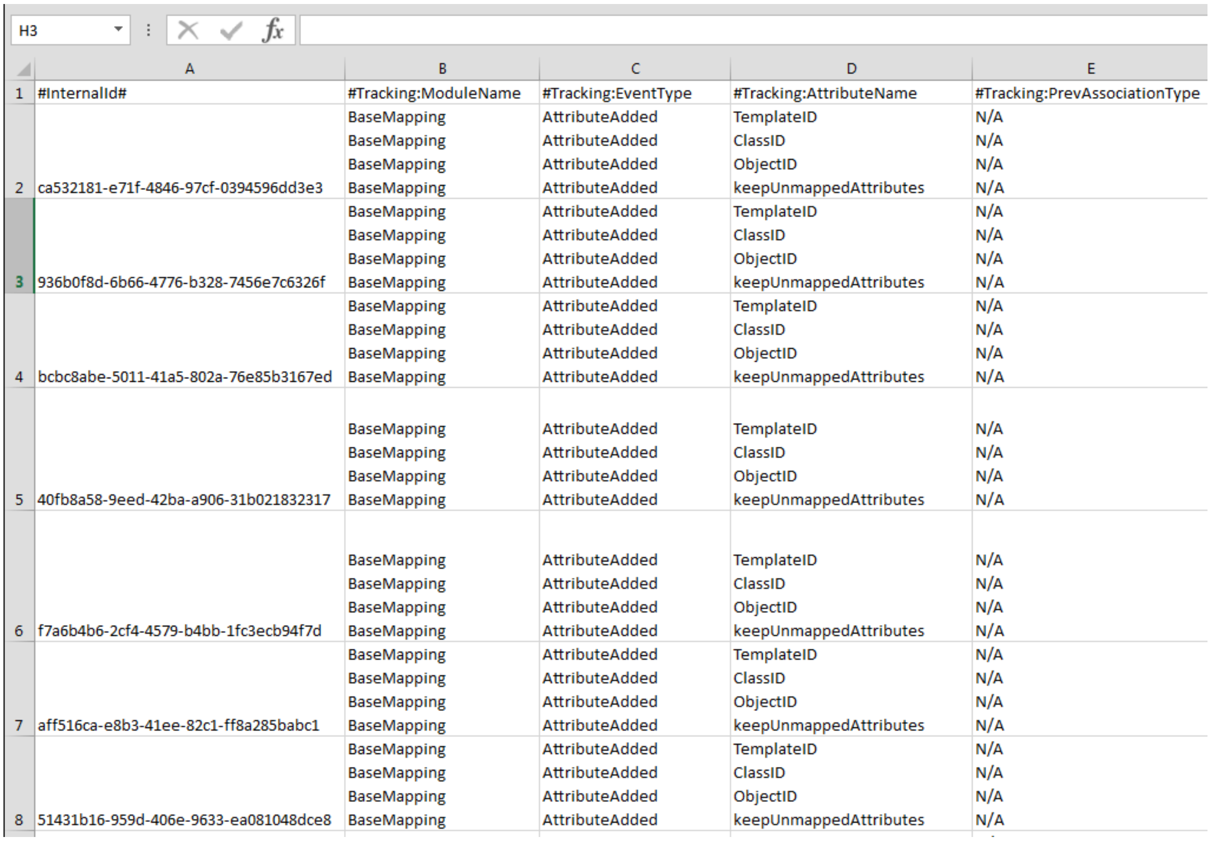
- `ObjectModified`, `ObjectAdded` and `ObjectRemoved`
- `AssociationAdded`, `AssociationChanged`, `AssociationReplaced`, `AssociationRemoved`, `AssociationMoved` and `AssociationsCleared`
- `AttributeAdded`, `AttributeChanged`, `AttributeReplaced`, `AttributeRemoved`, `AttributeMoved` and `AttributesCleared`

Example 1 of CSV document viewed in Excel



	ClassId	ObjectId	ClassName	Name	#Association:ContainedInStructure #TargetAttribute:ClassName
2	3D Model	171210CADstudio_brep			
3	STRUCTURALPOINTCONNECTION	1LTDboE71VI8muYYgkioRp	IFCSTRUCTURALPOINTCONNECTION	NODE 77	
4	OWNERHISTORY	1264715647	IFCOWNERHISTORY		
5	BEAMTYPE	1FBl5_0P_PJeeBXn0V9SCK	IFCBEAMTYPE	W18x40	
6	STRUCTURALCURVEMEMBER	2D8UhagYzlGP3paVpk$bpp	IFCSTRUCTURALCURVEMEMBER	1_1	
7	STRUCTURALPOINTCONNECTION	0ny3kFt9q_I9NnReXcC5I9	IFCSTRUCTURALPOINTCONNECTION	NODE 309	
8	STRUCTURALPOINTCONNECTION	3NmBUvusmQHPAtxsk9ihOH	IFCSTRUCTURALPOINTCONNECTION	NODE 171	
9	STRUCTURALPOINTCONNECTION	3Xc6spXrx4JAuORHAVt2KK	IFCSTRUCTURALPOINTCONNECTION	NODE 60	
10	BEAMTYPE	11WaCcbTOQGALKPCHGneBP	IFCBEAMTYPE	W12x35	
11	STRUCTURALPOINTCONNECTION	3lwqykjIjEGeGzLOjwB$6y	IFCSTRUCTURALPOINTCONNECTION	NODE 185	
12	STRUCTURALCURVEMEMBER	2h7w$6Y4tfIvvKhhsJK$30	IFCSTRUCTURALCURVEMEMBER	21_2	
13	STRUCTURALCURVEMEMBER	3pHQOnGq_9H91Oll_vMhDu	IFCSTRUCTURALCURVEMEMBER	83_1	
14	BEAM	0wOsVZzrq8GQgS1A8QQ02X	IFCBEAM	44_1	IFCBUILDING
15	STRUCTURALPOINTCONNECTION	0F$kaRpuBwIvZYvQOKTq85	IFCSTRUCTURALPOINTCONNECTION	NODE 46	
16	STRUCTURALCURVEMEMBER	0ZSXsoxFOpJRkbHGcoIEJP	IFCSTRUCTURALCURVEMEMBER	30_1	
17	BEAMTYPE	18cSTxLehzJf9MkUUpc3X5	IFCBEAMTYPE	W16x36	
18	BEAM	3K6PkhSh1CJ8BSKN5yWS2a	IFCBEAM	56_1	IFCBUILDING
19	STRUCTURALCURVEMEMBER	1wLL8EN_snHhHtYfJppvGL	IFCSTRUCTURALCURVEMEMBER	11_1	
20	STRUCTURALCURVEMEMBER	0Jik68oZgPGRuwjMlzMAAF	IFCSTRUCTURALCURVEMEMBER	51_1	
21	BEAMTYPE	2FLOpsc0KwJfPMWdWq2t$O	IFCBEAMTYPE	W18x50	
22	STRUCTURALCURVEMEMBER	3KQxynMSRRGOkrAFOgP_pZ	IFCSTRUCTURALCURVEMEMBER	5_1	
23	BEAM	3xZQTGSyY4JvsPcNyNnnAe	IFCBEAM	5_1	IFCBUILDING
24	STRUCTURALCURVEMEMBER	0YoN83e8TLHQcACW$avMY1	IFCSTRUCTURALCURVEMEMBER	87_1	
25	BEAM	1N4hDTvp8RGfXxn4dm1P9P	IFCBEAM	80_1	IFCBUILDING
26	STRUCTURALCURVEMEMBER	09ZANFvGHiHfdFVmFXfVP8	IFCSTRUCTURALCURVEMEMBER	67_5	
27	STRUCTURALCURVEMEMBER	14F0o4E5_oHgvf$20GjMgu	IFCSTRUCTURALCURVEMEMBER	17_1	
28	BEAM	0XUTsNY_T_HxUG0ihljdUk	IFCBEAM	79_1	IFCBUILDING
29	STRUCTURALPOINTCONNECTION	3$37O1o78EJxnNd7j2hTYo	IFCSTRUCTURALPOINTCONNECTION	NODE 218	
30	BEAM	1MUmaNLGsvHgVrzwi3pEpD	IFCBEAM	27_3	IFCBUILDING

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



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

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

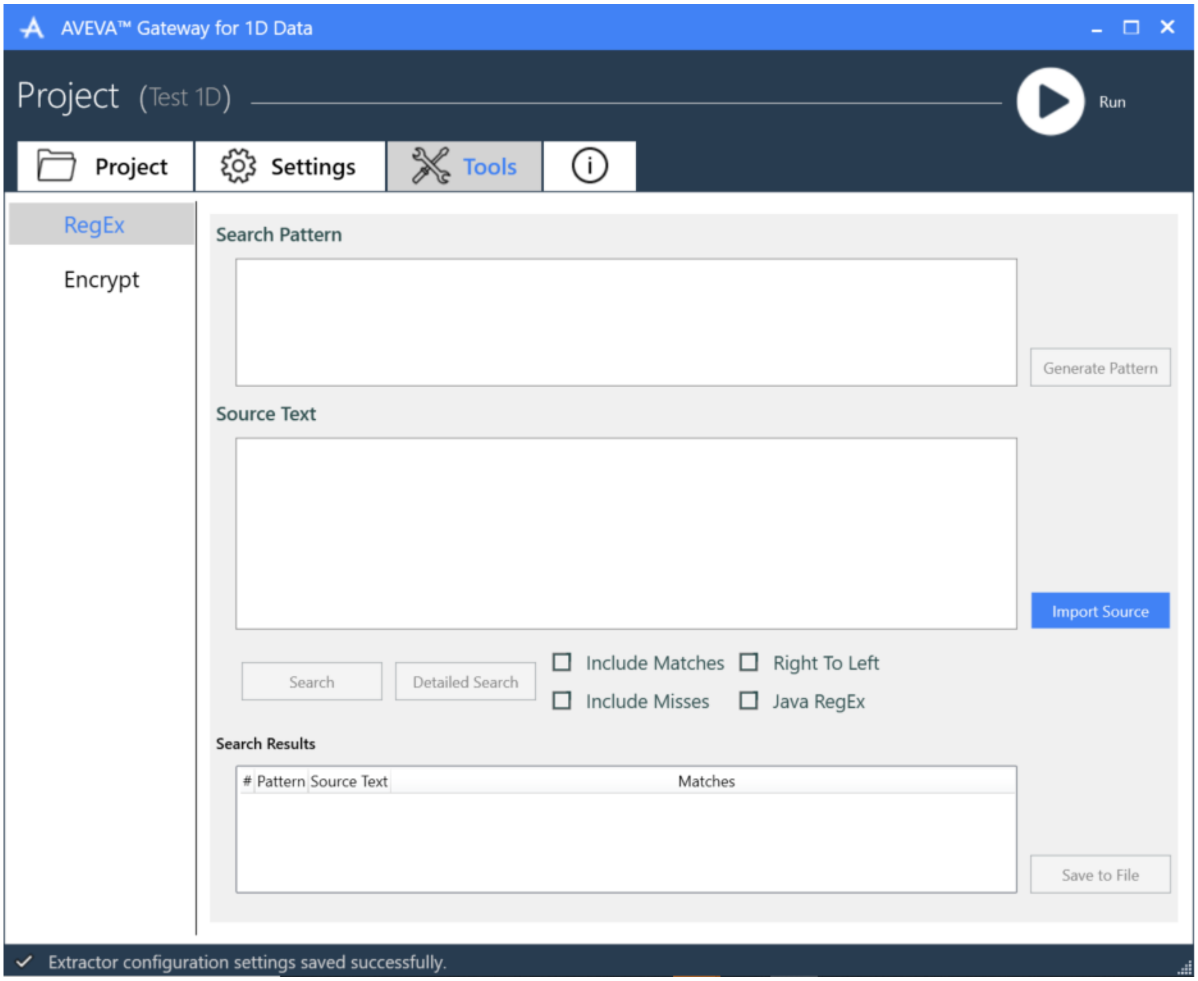
Object Parameters Available for the CSV Report

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

- `ObjectID`
- `ClassID`
- `ClassName`
- `#Attributes (selects all attributes)`
- `#Associations (selects all associations)`
- `#InternalID#`
- `#SourceID#`
- `#ObjectType#`
- Column name from tabular data source

Appendix G Appendix C: RegEx Utility

A regular expression (RegEx) is a special text string to describe a search pattern.



✓ Extractor configuration settings saved successfully.

The following table lists the various RegEx utility options and their various user actions.

RegEx Utility Options	**User Action**
Search Pattern	Select this to specify a regular expression search pattern or patterns. Multiple patterns can be specified by putting each pattern on a new line.
Generate Pattern	Select this to generate a regular expression search pattern.
Source Text	Type or paste the source text in the s**ource text** section. This activates the Generate Patterns button and generates Regular Expression patterns in the Search Pattern box that match the lines in the Source Text box.
Import Source	Click this to import the source text.

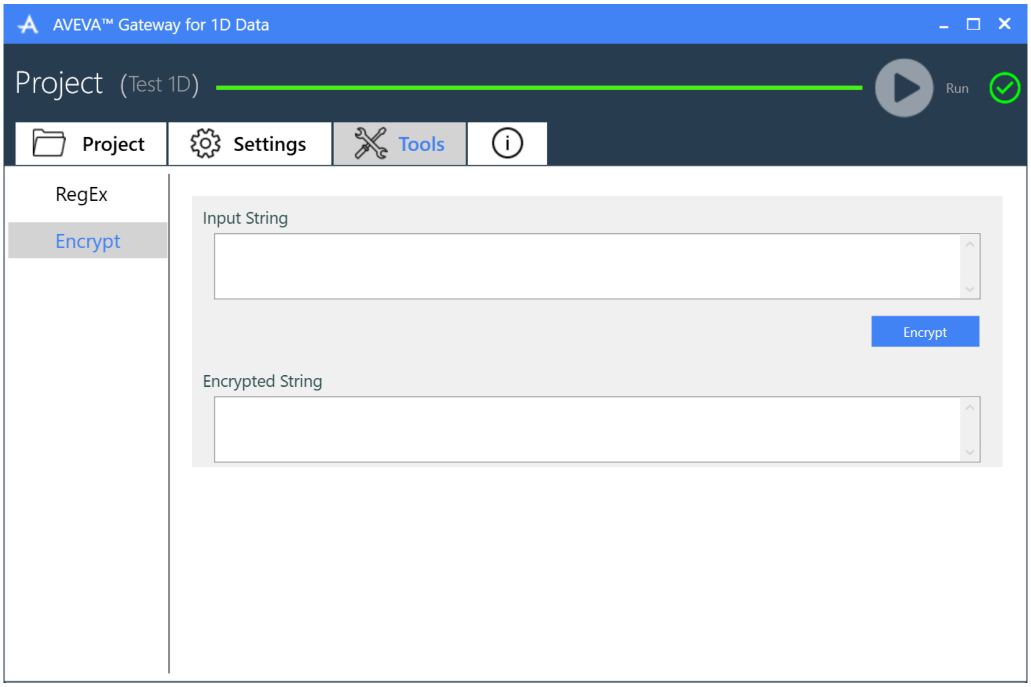
RegEx Utility Options	**User Action**
Search	Select this to apply the patterns to the source text. The summary of searches is listed in the Search Results box.
Detailed Search	Select this to search each line of the source text in each pattern. If a pattern finds a match on a line that line is no longer considered by subsequent patterns.
Include Misses	Select this to include the lines where a match is not found for specified pattern(s) in the search results.
Include Matches	Select this option to include the lines where a match is found for specified pattern(s) in the search results.
Right to Left	Select this option to allow the patterns to start searching from the end of lines in the source text and not the start.
Java RegEx	Select this option to employ the Java flavor of RegEx.
Search Results	Use the search results that is displayed indicating the patterns, source text, total source lines and matches.
Save to File	Click this to save the search results file.

Appendix H Appendix D: 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. Determine the connection string.
2. Select the option **Tools > Encrypt** and enter the connection string.
3. Click **Encrypt** to get the encrypted version of the connection string.
4. Take a copy of the encrypted string and paste it in place of the connection string in all supported mapping configuration sections.



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

Appendix I Appendix E: 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 prior to 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 which contains 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 J Appendix F: Tracking Changes in Data

The optional **'annotations'** feature allows you to store additional data about how each object, attribute or association have been changed due to processing of data under Extract, Transform and Load components.

These tracking attributes store data about creation, modification and deletion operations and can be inspected via CSV reporting. You can collect this information for the CSV report by setting XML <`annotations`> node to the respective module configuration in the Extract or Load settings or to extension configuration for the Transformer. The available annotations level values are `'None'` and `'Basic'`. When the 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.

Example for Extractor and Loaders:

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

Example for Transformer's Extensions:

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

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

Appendix K 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.

The system attributes can be either of the following:

- R – The attribute is read only. It is not possible to modify its value.
- R/W – The attribute can be modified in transformer configuration.

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

Attribute	Properties (Occurs in Object Type)	Usage		
		Base Mapping	Presentation Mapping	EIWM Loader
#TYPE#	manifest	R	R	-
#MANIFEST#	manifest	R	R	EXPORT
#INPUT_LOCATION#	manifest	R	R	EXPORT
#INPUT_FOLDER#	manifest	R	R	EXPORT
#GRAPHICS_FORMAT#	manifest	R	R	-
#UNITS#	manifest	R	R	EXPORT
#DEFINED_SEGMENTS#	manifest	R	R/W	-
#KEEP_FOLDER_TREE#	manifest	R	R	-
#TARGET_LOCATION#	manifest	R	R	EXPORT
#FILE_TIME_STAMP#	manifest	R	R	-
#UNITS-BORE#	manifest	R	R	EXPORT
#UNITS-CO-ORDS#	manifest	R	R	EXPORT
#UNITS-BOLT-LENGTH#	manifest	R	R	EXPORT
#UNITS-BOLT-DIA#	manifest	R	R	EXPORT
#UNITS-WEIGHT#	manifest	R	R	EXPORT
InfoLocator	manifest	R	R	EXPORT
#material_ambient_colour_RGB	Engineering Object	R	R/W	EXPORT

Attribute	Properties (Occurs in Object Type)	Usage		
		Base Mapping	Presentation Mapping	EIWM Loader
#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

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.

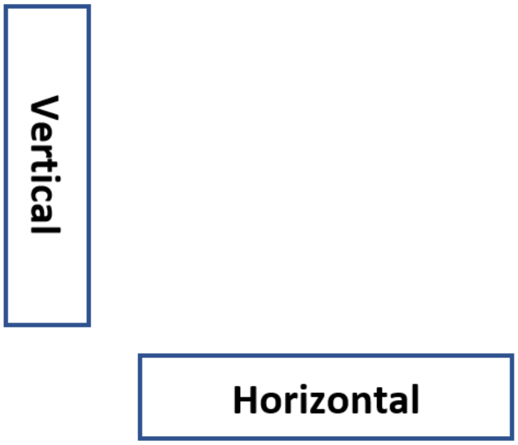
```
        <Attribute>
          <Conditions>
              <Attribute name="#GRAPHICS_FORMAT#" pattern="^.*$"/>
          </Conditions>
          <Name value="GRAPHICS_FORMAT" />
          <Value value="[#GRAPHICS_FORMAT#]"/>
     </Attribute>
```

The above example will add the engineering attribute named GRAPHICS_FORMAT.

Appendix L Appendix H: Limitations of the Gateway

Limitation 1:

The Gateway does not support the vertical text extraction or column-wise text extraction by default extraction (that is, `IFilter apply="false"` or is undefined in the Extract configuration XML file).



The extraction of texts is done from top-left to bottom-right resulting in the text objects, as shown below:

V

e

r

t

i

c

a

l

Horizontal

A total of nine text objects are extracted. Text on the vertical axis is not recognized as a single object as the extraction is done horizontally.

Note: The text can be extracted as a single object when using iFilter.

Limitation 2

Default processing of PDF files (that is, `IFilter apply="false"` or undefined in the Extract configuration XML file) results in objects with Page attributes. There is a limitation where the Page index starts with 0 irrespective of the page numbers displayed in the PDF Document.

Limitation 3

The Gateway is unable to detect whether the text in the input PDF is encoded and can only extract the text 'as is'. You can determine if the text is encoded by opening the file in a PDF reader, selecting some of it and copying it to a plain text application such as Notepad. If the plain form of the text is a set of scrambled symbols, then the input document's text is encoded.

Limitation 4 - Extracting Large Strings of Excel Files Using ODBC

Text can be cropped when using ODBC to extract large strings from Excel files.

Cells in Excel can have more than 255 characters, but ODBC expects such large cells based only on cells in the first N rows, defined by `TypeGuessRows` (`default = 8`) in the Registry. This is one of the Access Connectivity Engine parameters located in the Regsitry at:
`Computer\HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Office\ClickToRun\REGISTRY\MACHINE\Software\Microsoft\Office\16.0\Access Connectivity Engine\Engines\Excel`.

Therefore, if `TypeGuessRows` is set to 8 and the first large cell is in a row after row 8, then the extraction from ODBC will only read the first 255 characters.

You can use one of the two workarounds:

- Use a Datasource of 'Excel' rather than the ODBC driver, this requires MS Office to be installed.
- Set `TypeGuessRows` in the Registry Editor to 0, but this may increase the processing time and/or memory consumption.

Limitation 5 - 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.