

Generic Import

# Generic Import

The following document details the Generic Import process, the configuration and deployment of this tool. The Generic Import is presented here in two parts, the first section will be the configuration of the Generic Import Task Executor to allow the tool to be used from IIQ Tasks. The second part is to describe how the tool can be leveraged in your own code.

# Generic Import Task Executor

All the generic importer classes are based on the AbstractGenericImport abstract classm and when instantiated via the GenericImporter class will use the following generic attributes:

|  |  |  |
| --- | --- | --- |
| Entry | Description | Example values |
| genericImportDriverClass | The Generic Import driver class used for this import, currently supports JDBC, Delimited Text and Excel | JDBC  sailpoint.services.task. genericImport.JdbcImport  Delimited Text  sailpoint.services.task. genericImport.TextFileImport  Excel  sailpoint.services.task. genericImport.ExcelSaxImport |
| importInitRule | The initialization rule used for this import. | sailpoint.object.Rule name |
| importTransformRule | The transform rule used for this import. | sailpoint.object.Rule name |
| importRowRule | The row rule used for this import. | sailpoint.object.Rule name |
| importFinalizeRule | The finalize rule used for this import. | sailpoint.object.Rule name |
| importGroupBy | Semicolon separated list columns used as an index to group multi value fields together. | id |
| importMultiValueFields | Semicolon separated list of columns which are to be represented as multi valued fields. These fields are presented in the Transform rule as a List. | groupMembership |

JDBC Configuration

The following additional entries needed in the Task Definition for JDBC access are

|  |  |  |
| --- | --- | --- |
| Entry | Description | Example values |
| genericImportDriverClass | Must be set for JDBC generic import class | sailpoint.services.task. genericImport.JdbcImport |
| driverClass | JDBC Driver class | oracle.jdbc.OracleDriver |
| password | Password used for authentication for the JDBC resource, this value can be encrypted | password  1:327377w734YQQQxYyz== |
| url | JDBC URL for the database resource | jdbc:oracle:thin:@ldap://database. location.com:3389/P121,cn= db,dc=location,dc=com |
| user | User name used for authentication for the JDBC resource |  |
| sqlQuery | The SQL Query used to iterate through. If using groupBy, ensure the result set is sorted by the OD columns |  |

Text File Configuration

The following additional entries needed in the Task Definition for Delimited Text File access are

|  |  |  |
| --- | --- | --- |
| Entry | Description | Example values |
| genericImportDriverClass | Must be set for JDBC generic import class | sailpoint.services.task. genericImport.TextFileImport |
| importFileName | Filename of the text file to be iterated over. | C:\Import.file.txt |
| importFileDelimiter | The row delimiter for the file. “,” is default. | , |
| importHasHeader | Set to true if the import file has a header | true or false |
| importRemarkToken | Remark token, will ignore this line (Currently disabled!!!) | # |
| importHeader | If no header, then populate this entry with a semi-colon delimited column name list | column1;column2;column3 |
| importFileEncoding | Encoding use to read the file |  |

Excel SAX Configuration

The following additional entries needed in the Task Definition for Delimited Text File access are

|  |  |  |
| --- | --- | --- |
| Entry | Description | Example values |
| genericImportDriverClass | Must be set for Excel generic import class | sailpoint.services.task. genericImport.ExcelSaxImport |
| excelFilename | Excel filename | C:\somefile.xls |
| excelHasHeader | Does the worksheet have a header | Y |
| excelHeaderRow | What is the header row number | 1 |
| excelSheetName | The worksheet name used to return the iterator from | Wibble |
| excelHeader | Manually specify the header | Name,Address,DOB |

# Beanshell Rules

The Generic Importer uses four beanshell rules to perform the import, they are:

* **Init rule**  
  Initialisation code is here. The Beanshell interpreter has the same state across all the rules so methods and classes existing here can be called from the other rules.
* **Transform rule**  
  Transform your customers schema into the schema expected by your code. Data is presented as a Attributes class.
* **Row rule**  
  With the transformed information, create SailPoint objects or interact with SailPoint processes. Data is presented as an Attributes class.
* **Finalize rule**  
  Run on a successful import.

The following diagram shows how they are performed:



## Init rule details

Variables instantiated in the rule:

* log - Logger for this beanshell script, logger class “sailpoint.services.task.genericImport.GenericImporter”
* Context - SailPoint context
* taskResult - sailpoint.object.TaskResult object, from the Task Executor.
* taskAttributes - sailpoint.object.Attributes object, from the Task Executor

Returns:

* Nothing

Example:

## Transform rule details

Variables instantiated in the rule:

* log - Logger for this beanshell script, logger class “sailpoint.services.task.genericImport.GenericImporter”
* Context - SailPoint context
* taskResult - sailpoint.object.TaskResult object, from the Task Executor.
* taskAttributes - sailpoint.object.Attributes object, from the Task Executor
* row - A row of data from the input source, represented as a sailpoint.object.Attributes object.
* transform – A sailpoint.object.Attributes object used to hold the transformed row

Returns:

* transformed row

Example:

## Process Row rule details

Variables instantiated in the rule:

* log - Logger for this beanshell script, logger class “sailpoint.services.task.genericImport.GenericImporter”
* Context - SailPoint context
* taskResult - sailpoint.object.TaskResult object, from the Task Executor.
* taskAttributes - sailpoint.object.Attributes object, from the Task Executor
* row - A row of data returned from the Transform rule, represented as a sailpoint.object.Attributes object

Returns:

* Nothing

Example:

## Finalize rule details

Variables instantiated in the rule:

* log - Logger for this beanshell script, logger class “sailpoint.services.task.genericImport.GenericImporter”
* Context - SailPoint context
* taskResult - sailpoint.object.TaskResult object, from the Task Executor.
* taskAttributes - sailpoint.object.Attributes object, from the Task Executor

Returns

* Returns:

Example: