IdentityIQ Object Exporter Plugin User Guide

Version 4.1

Document Revision History

Revision Date	Written/Edited By	Comments
May 2015	Paul Wheeler	Initial Release
June 2016	Paul Wheeler	Minor additions and changes for release 1.1
August 2016	Paul Wheeler	Details for CDATA and Merge file options in release 2
June 2017	Paul Wheeler	Minor update for version 2.1, SSD v4
September 2017	Paul Wheeler	New option for pattern-matching object names for export, for version 3.0, SSD v5.
May 2018	Paul Wheeler	Version 3.1, SSD v6: added option to strip metadata that is typically not useful when migrating objects between environments.
December 2020	Paul Wheeler	Version 4.0: Object Exporter is now a plugin and is no longer part of the SSD. Enhancements for IIQDA-compatible filenames and XPath-based reverse-tokenization.
January 2023	Paul Wheeler	Version 4.1: Fixes and enhancements

Table of Contents

Overview 4

Installation 6

Using the Object Exporter 7

Overview

The Object Exporter is a custom IdentityIQ task for quick and easy export of XML objects such as applications, rules, workflows, configurations and any other objects that are managed in IdentityIQ. It is deployed as an IdentityIQ plugin. The task has the following features:

- Exports XML objects to the same filesystem structure used by the Services Standard Build (SSB)
- Allows the user to specify which classes get exported
- Allows the user to specify which objects get exported by matching their names with a regular expression
- Can optionally export only the objects that have been created or modified after a given date
- Can optionally remove IDs and timestamps from the exported objects
- Can optionally remove metadata that is typically not useful when migrating objects between environments
- Allows definition of a naming format for the exported XML files, defaulting to the same format used by the IdentityIQ Deployment Accelerator (IIQDA)
- Can optionally produce tokenized XML files ready for use in the SSB build, using the same XPath-based tokenization used by the IdentityIQ Deployment Accelerator in addition to reverse-lookup of tokens in a target.properties file
- Can add CDATA sections and unescape BeanShell code and other escaped elements
- Can create "merge" files by comparing the XML of objects being exported with previously exported base versions of the objects (for supported object types).

The tool is designed to help developers working in an IdentityIQ sandbox or development environment, particularly when working directly within IdentityIQ rather than working on external master files and importing those. When working directly within IdentityIQ it can be easy to lose track of what was changed and when, making it difficult to keep the build up-to-date. The Object Exporter provides the following advantages over other export methods:

- Allows the developer to easily export everything they have been working on over a given period, or since the start of a project, removing the need to maintain scripts used to export objects from the console.
- By maintaining a standard naming format for the resulting files it also helps to prevent the creation of multiple copies of the same object XML when there is more than one developer working on an IdentityIQ project. Compatibility with the naming format used by the IdentityIQ Deployment Accelerator allows both tools to be used as needed without filename conflicts.
- Avoids the need to tediously replace text in the XML files with the tokens used in the SSB.
- Avoids the need to create the SSB folder structure when exporting objects.
- The resulting XML files can be added directly to the SSB since they are in the correct folder structure and have the correct tokens.
- The simple process also makes it easy to provide exported files that can be uploaded to a code repository or just stored outside of IdentityIQ as a quick and simple backup of everything that has changed.
- Fixes issues that the console checkout process has with exporting IDs rather than names for objects referenced by certain classes.
- The use of CDATA sections with unescaped code aids the readability of the code when viewed offline.
- Allows the export of "merge" files that contain only the additions and changes in certain types of objects when compared with base exports.

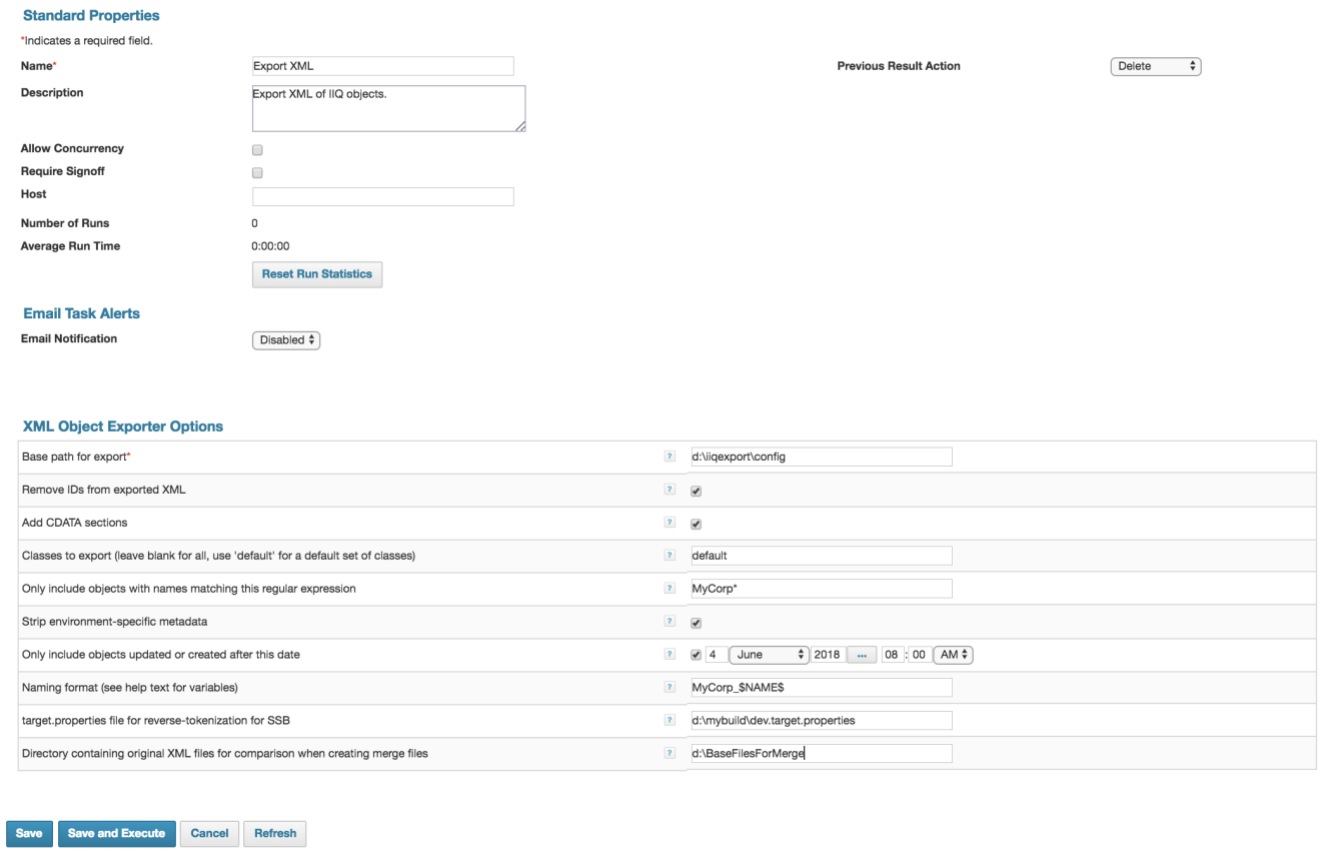
Installation

The Object Exporter is installed using the *Plugins* menu item under the gear icon. Click on the *New* button, then drag and drop the object-exporter-plugin-4.1.0.zip file. The plugin will only work alongside the IdentityIQ Plugin Framework in IdentityIQ 7.2 and above.

Note that if you are upgrading from a previous non-plugin version of the Object Exporter (earlier than version 4) you should first remove that version (delete the ExportXML.class file and the imported TaskDefinition) before installing the plugin.

Using the Object Exporter

To access the Object Exporter in IdentityIQ, go to Monitor -> Tasks. Create a new task of type "XML Object Exporter". The task has several user-configurable options:



Base path for export

Enter the path to which you want to export the XML files. This must be a path that can be accessed by the server that runs the task.

Remove IDs from exported XML

Defines whether the exported XML files will include the IDs and created/modified timestamps for the objects in the environment that you are exporting from. In most cases this should be set to True.

Add CDATA sections
Encloses code in CDATA sections and unescapes the code for ease of reading in the exported file. This works for BeanShell code in rules and scripts, HTML formatting in email templates and SOAP messages in IntegrationConfig objects

Classes to export
This should be a comma-separated lists of object types to export. If left blank it will export all object types. If set to "default" it will export only the types of objects that are commonly exported during an IdentityIQ project. It can also be a combination of "default" and other classes – for example you could use "default,Scope".

Only include objects with names matching this regular expression
Specify a regular expression if you want to only export objects with names matching a given pattern. For example, if you just want to export all objects that start with "MyCorp" you would enter "MyCorp.*". Leave this blank if you do not wish to filter by object name.

Strip environment-specific metadata
Remove data from objects that is not typically useful when migrating objects between environments. The following data will be removed:

- Statistical data on TaskDefinitions, such as number of runs and average run time
- Timestamps on TaskSchedule objects for last and next execution
- Entries in Application objects for account aggregation start and end timestamps, delta aggregation cookie data for Active Directory applications

Only include objects updated or created after this date
If enabled, this option allows the user to enter a date and time which defines the earliest creation and modification timestamps by which we will filter the objects to be included in the export. If disabled, creation and modification timestamps will be ignored.

Naming format
Allows the user to define a format for naming the exported XML files. There are three optional variables that can be used:

- $Name$ The name of the XML object
- $Class$ The name of the object class, e.g. Application, Rule etc.
- $Default$ The naming format used by the IdentityIQ Deployment Accelerator

The filename is appended with ".xml". Using the variables, you can build a custom naming format for the exported files. For example, "MyCorp_$Class$_$Name$" would export an application called "ActiveDirectory" with the file name "MyCorp_Application_ActiveDirectory.xml".

Leaving this blank will use the same naming format used by the IdentityIQ Deployment Accelerator (i.e. it will assume $Default$). Note that this changed in version 4; in previous versions it used the name of the XML object when left blank, so if you want to continue using this format for compatibility with existing filenames created this way you should use $Name$.

Illegal characters (anything that is not an alphanumeric character, dot or hyphen) and spaces will be replaced with underscores, unless the default format is selected, in which case spaces are removed.

target.properties file for reverse-tokenization for SSB
Optionally defines the path to a target.properties file that provides token values for the reverse-tokenization functionality.

The term "reverse-tokenization" may be considered something of a misnomer since the purpose it serves is to replace values in the exported files with tokens so that they can be used in the SSB across environments, so it might be more appropriate to call it "tokenization". However, "reverse-tokenization" has become common terminology for this process amongst IdentityIQ implementers, so this document will continue to use it.

There are two supported types of reverse-tokenization:

- XPath (as also used by the IdentityIQ Deployment Accelerator)
- Reverse-lookup (which can be considered as legacy functionality).

Both require a target.properties file that defines the tokens and how they will be added to the XML files, and the same target.properties file can be used to support both types of reverse-tokenization simultaneously if needed. The IIQDA requires this file to be called reverse.target.properties and it must exist at the root of the SSB/D. If you are using the IIQDA in addition to the Object Exporter you should use this filename (and include the path in this option).

Using XPath reverse-tokenization, tokens should be defined in XPath format like this example:

```
/Application[@name\='Active\ Directory']/Attributes/Map/entry[@key\='user']/@value=%%USERNAME%%
```

In this example, the value found in the 'user' entry of the attributes map of the 'Active Directory' application will be substituted with the token '%%USERNAME%%'. Note that spaces in object names need to be escaped with a backslash.

Using the reverse-lookup type of reverse-tokenization, any text found in the XML that matches a token value found in the target.properties file will be replaced with the corresponding token, where the tokens are defined in the same way as a standard target.properties file. For example, a line like this would replace any text in any XML file that matches 'server1.example.com' with the token '%%AD_IQSERVICE_HOST%%':

```
%%AD_IQSERVICE_HOST%%=server1.example.com
```

Note that tokens in the target.properties file that have values of "true" or "false" will be ignored when using reverse-lookup since these values are present in many objects and replacing them all with a token is not usually the intention. The same applies to token values that are blank or just whitespace. In general, the reverse-lookup type of reverse-tokenization should be used with care as it can result in multiple values unintentionally being replaced with tokens if the value is not unique across all XML files.

If this option is left blank, no tokenization of the export files will take place.

Directory containing original XML files for comparison when creating merge files
Optionally defines the location under which the "base" XML files can be found for use when creating "merge" files. These are files that contain only the XML entries that have been added or changed when compared with the base files and will be exported with the ImportAction property set to "merge". The task compares the current XML being exported with the base files and creates merge files based on the differences. When imported into IdentityIQ, the entries in these files will be merged with an existing object that has the same name, rather than overwriting it. Merge files should be used for certain types of objects (such as Configuration and UIConfig objects) where an upgrade to IdentityIQ may result in the addition of new entries in these objects, which would otherwise get removed when importing a full object that does not have an ImportAction of "merge".

At the start of a project it is useful to export all the objects that support merging to obtain the base files that are referenced here and place these in a single location. When looking for merge files in the provided location, the task will recursively search in subfolders for any files that have .xml extensions and use these as the base files.

The task currently supports objects of the following types for merging:

- Configuration
- UIConfig
- ObjectConfig
- AuditConfig
- Dictionary

After selecting the desired options, click "Save and Execute". The task will start running and its progress can be monitored in the Task Results. When completed, the results will show a summary of the number of XML files exported for each class and a total of all objects exported.