SSF Tools: Certification Exclusion Framework User Guide

Document Revision History

Revision Date	Written/Edited By	Comments
May 2017	Paul Wheeler	Initial release with SSD v4

Table of Contents

Introduction

Certification Exclusion rules are often used in IdentityIQ certifications to define the entities and items that should not be included in the certification. These rules help customers have more control over the contents of certifications, allowing more granularity than can be provided via the UI. However, these rules can get very long and complex, making them difficult to understand, cumbersome to maintain and error-prone. Additionally, customers sometimes want to define specifically what gets included in a certification rather than the items that get excluded, which can add further complexity. The Certification Exclusion Framework aims to address these issues, replacing a rule that may contain hundreds of lines of complex custom code with one that contains a couple of standard lines of code, with the logic maintained in a separate filter-driven configuration that is more intuitive.

The Certification Exclusion Framework can be used with all certification types supported in IdentityIQ. It will work with IdentityIQ version 6.3 and higher.

Components

The Certification Exclusion Framework includes seven Java classes:

- Exclusioner
- ExclusionSet
- FilterSelector
- EntitySelector
- ItemSelector
- CertifiableEntityMatcher
- CertifiableItemMatcher

In addition, there should be at least one Custom object that contains the configuration for the set of entities and items that will be excluded, and at least one Exclusion Rule that will call the Java code that references this configuration and processes the exclusions. A number of sample Custom objects and Exclusion Rules are provided.

Installation

If you are using the Services Standard Deployment (SSD), the Certification Exclusion Framework will be automatically deployed. To install it manually:

1. Compile the Java classes.
2. Create the folder WEB-INF/classes/sailpoint/services/standard/exclusionframework on each of your IdentityIQ application servers and copy all the Java classes to this folder.
3. Restart the application servers.

Whether you are installing it with the SSD or manually, you will also need one or more Custom objects that define the set of entities and items to be excluded, and an Exclusion Rule to reference the Exclusion Framework. The sample Custom objects and Exclusion Rules included with the framework will not be installed with the SSD by default and are provided as examples only. They can be adapted for your requirements and moved out of the “Samples” folder so that they will be deployed with the SSD. Alternatively they can be Imported manually, either using the ‘import’ command in iiq console or the “Import from File” functionality under Global Settings (IdentityIQ 7.0 and later) or System Setup (previous versions) in the IdentityIQ UI.

The sample Custom objects and Exclusion rules can be found in the SSD under the config/SSF_TOOLS/CertificationExclusionFramework/Samples folder.

Certification Entities and Items

IdentityIQ offers a number of different types of certification, and each uses specific entities (defining the object being certified) and items (describing the access or properties of the entity that the approve/revoke decisions are made on). For example, in a Manager certification the entity is always an Identity object, while the items can be Bundles (roles), EntitlementGroups (individual entitlements) and PolicyViolations. So a manager will typically review the roles, entitlements and policy violations that are associated with each of their subordinate identities.

To use the Certification Exclusion Framework it is important to understand which entities and items are available for each certification type within the framework. These are given below.

Certification type	Entity	Items
Manager	Identity	EntitlementGroup, Bundle, PolicyViolation
Application Owner	Identity	EntitlementGroup, Bundle, PolicyViolation
Entitlement Owner	DataOwnerCertifiableEntity	Entitlements, Identity
Advanced	Identity	EntitlementGroup, Bundle, PolicyViolation
Role Membership	Identity	Bundle
Role Composition	Bundle	Bundle, Profile
Account Group Permissions	ManagedAttribute	EntitlementGroup
Account Group Membership	ManagedAttribute	EntitlementGroup

Note that some of the items listed are translated by the framework from other types. For example, a Role Composition certification usually has items of type RoleCertifiable and Profile, but the framework exposes the Bundle enclosed in the RoleCertifiable object and uses that in its place as it is more useful for filtering purposes. An Entitlement Owner certification has items of type DataOwnerCertifiable, each of which contain an Identity object and an Entitlements object, and these are exposed by the framework for filtering. Details on how to reference these types of items in filters is provided in the next section under “Item Selectors”.

Exclusion Set Custom objects

The Exclusion Set defines the filters that are used to exclude entities and items from the certification. It also defines other data used in the exclusion, including the reason for exclusion and an exclusion mode. The Exclusion Set is held in an object of type Custom.

Detailed information about configuration of Filters in XML objects is not covered in this guide, but more information can be found in the “Filters and Filter Strings” documentation on Compass: https://community.sailpoint.com/docs/DOC-6922. See the “Filter Element in XML” section.

A sample Exclusion Set defined by a Custom object is given below. This example is for a Manager Certification; however, it could also be used for an Application Owner or Advanced certification since those certification types have the same entity and item types as a Manager certification.

```
<?xml version='1.0' encoding='UTF-8'?>
<!DOCTYPE Custom PUBLIC "sailpoint.dtd" "sailpoint.dtd">
<Custom name="Sample Manager Certification Exclusion Set">
 <Attributes>
  <Map>
   <entry key="entitySelectors">
    <value>
     <List>
      <CompositeFilter operation="AND">
       <Filter ignoreCase="true" operation="EQ" property="name" value="Adam.Kennedy"/>
      </CompositeFilter>
      <CompositeFilter operation="AND">
       <Filter ignoreCase="true" operation="EQ" property="dept" value="Accounting"/>
      </CompositeFilter>
     </List>
    </value>
   </entry>
   <entry key="itemSelectors">
    <value>
     <List>
      <CompositeFilter operation="AND">
       <Filter ignoreCase="true" operation="EQ" property="name" value="Customer Service"/>
       <Filter operation="EQ" property="class" value="Bundle"/>
      </CompositeFilter>
      <CompositeFilter operation="AND">
       <Filter ignoreCase="true" matchMode="START" operation="LIKE" property="name" value="Accounts"/>
       <Filter operation="EQ" property="class" value="Bundle"/>
      </CompositeFilter>
      <CompositeFilter operation="AND">
       <Filter ignoreCase="true" operation="EQ" property="policyName" value="Account Policy"/>
       <Filter operation="EQ" property="class" value="PolicyViolation"/>
      </CompositeFilter>
      <CompositeFilter operation="AND">
       <Filter matchMode="START" operation="LIKE" property="memberOf" value="CN=Development"/>
       <Filter operation="EQ" property="class" value="EntitlementGroup"/>
       <Filter operation="EQ" property="application" value="Corporate AD"/>
      </CompositeFilter>
      <CompositeFilter operation="AND">
       <Filter matchMode="START" operation="LIKE" property="ma.displayName" value="Training"/>
       <Filter operation="EQ" property="class" value="EntitlementGroup"/>
       <Filter operation="EQ" property="application" value="TimesheetSystem"/>
      </CompositeFilter>
     </List>
    </value>
   </entry>
   <entry key="mode" value="EXCLUDE"/>
   <entry key="reason" value="Exclude identities, roles, entitlements and policy violations example"/>
  </Map>
 </Attributes>
</Custom>
```

The Custom object is a map of entries for the Entity Selectors, Item Selectors, mode and reason.

Entity Selectors

Entity Selectors are defined in the “entitySelectors” entry in the Custom file. Entity Selectors are made up of Filters which define which entities will be excluded. The example is for a Manager Certification, for which an entity is always an Identity. In this case we are excluding a single identity “Adam.Kennedy” and any identities that have the “dept” attribute set to “Accounting” and the “location” attribute set to “London”.

```
<entry key="entitySelectors">
 <value>
  <List>
   <CompositeFilter operation="AND">
    <Filter ignoreCase="true" operation="EQ" property="name" value="Adam.Kennedy"/>
   </CompositeFilter>
   <CompositeFilter operation="AND">
    <Filter ignoreCase="true" operation="EQ" property="dept" value="Accounting"/>
                    <Filter ignoreCase="true" operation="EQ" property="location" value="London"/>
   </CompositeFilter>
  </List>
 </value>
</entry>
```

There are two CompositeFilters here. The first contains a single Filter defining that the specific identity “Adam.Kennedy” will be excluded. The other contains two Filters with an “AND” operation, meaning both must be true for the exclusion to be valid; if the “dept” attribute is “Accounting” AND the “location” attribute is “London”, the identity will be excluded. The two CompositeFilters can be considered to be combined with an operation of “OR”, meaning either can be evaluated to a true condition for the exclusion to occur.

Item Selectors

Item Selectors are defined in the “itemSelectors” entry in the Custom file. Item Selectors are made up of Filters which define which certifiable items will be excluded. The example Manager Certification could have items of classes EntitlementGroup, Bundle and PolicyViolation. In this case we are excluding any of the following from the certification:

- The “Customer Service” role
- Any role with a name that starts with “Accounts”
- The Policy Violation named “Account Policy”
- An entitlement in the “Corporate AD” application that is a “memberOf” attribute that starts with “CN=Development”
- An entitlement in the application “Timesheet System” that has a display name that starts with “Training”

```
<entry key="itemSelectors">
 <value>
  <List>
   <CompositeFilter operation="AND">
    <Filter ignoreCase="true" operation="EQ" property="name" value="Customer Service"/>
    <Filter operation="EQ" property="class" value="Bundle"/>
   </CompositeFilter>
```

```
      <CompositeFilter operation="AND">
        <Filter ignoreCase="true" matchMode="START" operation="LIKE" property="name" value="Accounts"/>
        <Filter operation="EQ" property="class" value="Bundle"/>
      </CompositeFilter>
      <CompositeFilter operation="AND">
        <Filter ignoreCase="true" operation="EQ" property="policyName" value="Account Policy"/>
        <Filter operation="EQ" property="class" value="PolicyViolation"/>
      </CompositeFilter>
      <CompositeFilter operation="AND">
        <Filter matchMode="START" operation="LIKE" property="memberOf" value="CN=Development"/>
        <Filter operation="EQ" property="class" value="EntitlementGroup"/>
        <Filter operation="EQ" property="application" value="Corporate AD"/>
      </CompositeFilter>
      <CompositeFilter operation="AND">
        <Filter matchMode="START" operation="LIKE" property="ma.displayName" value="Training"/>
        <Filter operation="EQ" property="class" value="EntitlementGroup"/>
        <Filter operation="EQ" property="application" value="TimesheetSystem"/>
      </CompositeFilter>
    </List>
  </value>
</entry>
```

This consists of five CompositeFilters, any of which can be evaluated to a true condition for the exclusion to occur.

Note that the final CompositeFilter contains this filter:

```
<Filter matchMode="START" operation="LIKE" property="ma.displayName" value="Training"/>
```

The use of the “ma.” prefix has a special meaning here; it means we are looking for an attribute (“displayName” in this case) on the ManagedAttribute object that corresponds to the EntitlementGroup. This is useful because the ManagedAttribute holds metadata that is not available on the EntitlementGroup which may be useful in defining exclusions. The “ma.” prefix only applies to items of type EntitlementGroup.

For an Entitlement Owner certification, the framework exposes the Identity and Entitlements objects that are enclosed in a DataOwnerCertifiable object, and both the Entitlements and Identity objects can be filtered. The Entitlements objects are filtered by directly referencing the properties of the Entitlements objects, but Identity objects must be prefixed with “identity.” as shown in this example:

```
<entry key="itemSelectors">
  <value>
    <List>
      <CompositeFilter operation="AND">
        <Filter operation="EQ" property="nativeIdentity" value="djones"/>
        <Filter operation="EQ" property="application" value="Active_Directory"/>
      </CompositeFilter>
      <CompositeFilter operation="AND">
        <Filter operation="EQ" property="identity.name" value="Adam.Kennedy"/>
      </CompositeFilter>
    </List>
  </value>
</entry>
```

Here, the item is excluded if the Entitlements object has a nativeIdentity attribute of “djones” and an application attribute of “Active_Directory”, OR if the Identity object has a name of “Adam.Kennedy”.

Reason

The entry for “reason” in the Custom object defines the wording given as the reason for the exclusion. This is displayed in the UI under “View Exclusions” in the certification and can be useful for reporting purposes.

```
<entry key="reason" value="Exclude identities, roles, entitlements and policy violations example"/>
```

Mode

By default, the Exclusion Framework works in “exclude mode”, where it will exclude the entities and items defined in the Custom object. However, it is possible to change this behavior so it operates in “include mode”, where everything is excluded except these entities and items. The example shows the “mode” entry set to “EXCLUDE” for exclude mode.

```
<entry key="mode" value="EXCLUDE"/>
```

To change to “include mode”, modify this entry to:

```
<entry key="mode" value="INCLUDE"/>
```

Exclusion Rule

The Exclusion Rule calls a method on the Exclusioner Java class, which processes the exclusions. In its simplest form, the Exclusion Rule looks like this:

```
import sailpoint.services.standard.exclusionframework.Exclusioner;

return Exclusioner.processExclusions("My Exclusion Set", entity, items, itemsToExclude);
```

In this example, “My Exclusion Set” is the name of the Custom Object that defines the Exclusion Set for this certification. The “entity”, “items” and “itemsToExclude” are existing variables that are passed into the Exclusion Rule by IdentityIQ. The object returned from this method is a String containing the reason for the exclusion as defined in the Custom object.

Limitations

The Certification Exclusion Framework will not work with versions of IdentityIQ earlier than 6.3.