

Referential Plugin

User Guide

May 3, 2019

V1.0

Version Control

|  |  |  |
| --- | --- | --- |
| Date | Authors | Version/Comments |
| 5/3/2019 | Kent Lee | V1.0 – First release |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

Table of Contents

[1. Overview 3](#_Toc8378188)

[1.1 Referential Plugin 3](#_Toc8378189)

[2. Functionalities 4](#_Toc8378190)

[2.1 Query user from referential database 4](#_Toc8378191)

[2.2 Query resource (item) from referential database 4](#_Toc8378192)

[2.3 Query resource (program) from referential database 4](#_Toc8378193)

[2.4 Maintain an in-memory cache of user attributes. 5](#_Toc8378194)

[2.5 Maintain an in-memory cache of item attributes. 5](#_Toc8378195)

[3. Component 6](#_Toc8378196)

[3.1 Package 6](#_Toc8378197)

[3.2 Deploy 6](#_Toc8378198)

[3.3 Configure User Referential Plugin 6](#_Toc8378199)

[3.3.1 General information 6](#_Toc8378200)

[3.3.2 Cache information 6](#_Toc8378201)

[3.3.3 DB information 8](#_Toc8378202)

[3.4 Configure Resource Referential Plugin 9](#_Toc8378203)

[3.4.1 General information 9](#_Toc8378204)

[3.4.2 Cache information 9](#_Toc8378205)

[3.3.3 DB information 10](#_Toc8378206)

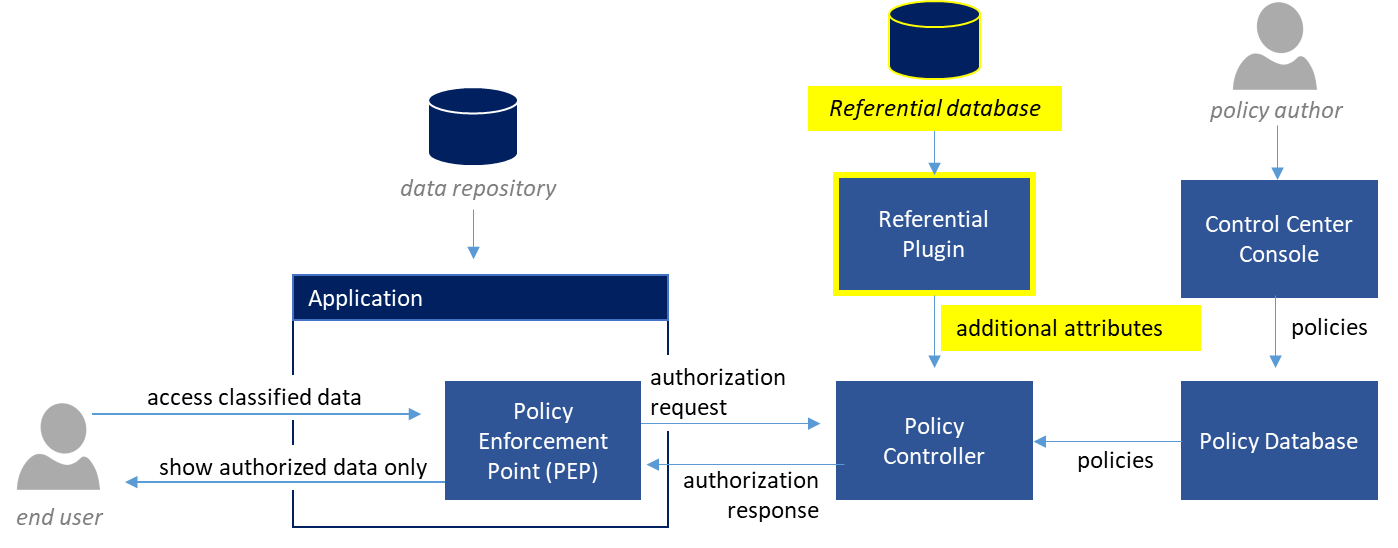
[4. Troubleshooting 13](#_Toc8378207)

[5. Appendix 14](#_Toc8378208)

[5.1 Referential Plugin, Enforcer and Enrollment 14](#_Toc8378209)

# Overview

## Referential Plugin



The Referential Plugin is a Dynamic Attribute plugin designed for the Policy Controller to fetch additional user and resource attributes from the referential database. These additional attributes will be used by the Policy Controller to evaluate authorization requests.

The plugin consists of the user attribute plugin and resource attribute plugin.

**How it works**

1. When a user tries to access data, a PEP or enforcer will intercept the request and will submit an authorization request to the Policy Controller. The authorization request will contain as much metadata about the user and resource (data being accessed). In most cases, the amount of metadata available to the application and enforcer is limited.
2. The Policy Controller will compare the metadata with the information control policies deployed. If the metadata required to evaluate a policy is not available from the authorization request, the policy controller will use the referential plugin to fetch additional attributes (or metadata) from the referential database.
   * note: the metadata collected from the referential database will be temporarily stored in the cache of the referential plugin to improve throughput and network bandwidth usage
3. The Policy Controller will then use all the metadata it has collected to evaluate the policies and return an authorization response to the PDP

# Functionalities

The Referential plugin is designed and developed to provide the following functionalities gathered from Airbus Helicopters.

## Query user from referential database

* A user can be identified by more than one identifier (ID). These IDs depend on the ID that the PEPs use to send request.
* The plugin should return the same user or group regardless of which ID the PEP passes. This means two PEPs using different attributes in referential for ID can work with one PDP.
* **All IDs must be unique across all database configured.**
* More than one user attribute can be configured to be retrieved. Each attribute is either single-valued or multi-valued and such information should be specified in the plugin’s properties file.
* The plugin would return EvalValue.NULL if user is not found.
* The plugin would return EValValue.NULL or MultiValue.EMPTY if the attribute is not found, depending on the attribute’s cardinality.
* The plugin connects to database using connection pool.
* The plugin loads all user information from the database in to the cache and also reloads the cache based on the cache\_refresh\_period property value.

## Query resource (item) from referential database

* An item can be identified with resource id, which is OBID in this case.
* OBID is unique across referential database.
* More than one item attribute can be configured to be retrieved. Each attribute is either single-valued or multi-valued and such information should be specified in the plugin’s properties file.
* The plugin would return EvalValue.NULL if item is not found.
* The plugin would return EValValue.NULL or MultiValue.EMPTY if the attribute is not found, depending on the attribute’s cardinality.
* The plugin connects to database using connection pool.
* The plugin loads all item information from the database in to the cache and also reloads the cache based on the cache\_refresh\_period property value.

## Query resource (program) from referential database

* A program can be identified with resource id, which is ID\_HD in this case.
* ID\_HD is unique across referential database.
* More than one program attribute can be configured to be retrieved. Each attribute is either single-valued or multi-valued and such information should be specified in the plugin’s properties file.
* The plugin would return EvalValue.NULL if program is not found.
* The plugin would return EValValue.NULL or MultiValue.EMPTY if the attribute is not found, depending on the attribute’s cardinality.
* The plugin connects to database using connection pool.
* The plugin loads all item information from the database in to the cache and also reloads the cache based on the cache\_refresh\_period property value.

## Maintain an in-memory cache of user attributes.

* On PDP start/restart, the plugin would contain no data about user.
* Cache entry has a configurable expiration (time\_to\_live), which can be from 0 to **INFINITE**. This expiration is calculated from the time the entry register itself into cache. Supported units are: SECS, MINS,HRS,DAYS
* When time\_to\_live expires, cache will be missed on the next request and the plugin will query the user from database and put back the user to cache.
* On PDP shutdown, the cache will be destroyed.

## Maintain an in-memory cache of item attributes.

* The cache can be configure to work on live mode or purge mode.
* On PDP start/restart, the plugin would contain no data about item.
* When in live mode, cache entry has a configurable expiration (time\_to\_live), which can be from 0 to **INFINITE**. This expiration is calculated from the time the entry register itself into cache. Supported units are: SECS, MINS,HRS,DAYS
* When time\_to\_live expires, cache will be missed on the next request and the plugin will query the item from database and put back the item to cache.
* When in purge mode, the cache refresh process will be started at a configurable fixed timestamp and subsequent refreshes will happen at a fixed rate calculated from the start timestamp. All the data in cache will be purge.
* In refresh process, if the refresh mode is turn on, the plugin will pull all the item information and put into cache after purging.
* When encounter exceptions, the cache refresh process will attempt a configurable number of retries between a configurable period
* On PDP shutdown, the cache will be destroyed

# Component

## Package

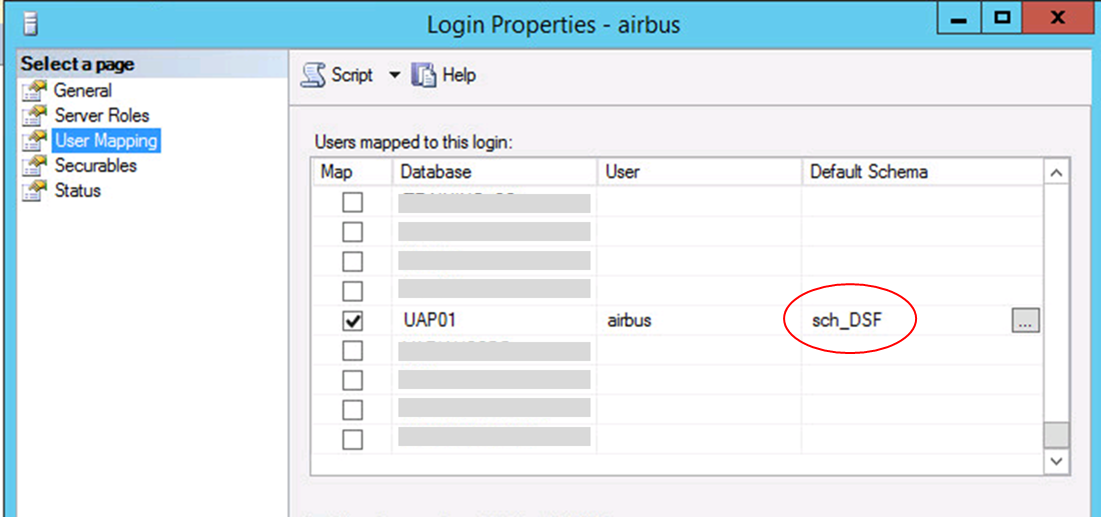
The referential plugin package contains the following components:

* jars/UserAttributeProvider.jar
* jars/ResourceReferentialPlugin.jar
* config/ResourceReferentialPlugin.properties
* config/UserReferentialPlugin.properties
* xlib/\*.jar

## Deploy

To deploy the Referential Plugin, perform the following tasks in the Policy Controller servers.

1. Stop the Policy Controller service if it is running.
2. Copy **ResourceReferentialPlugin.properties** and **UserReferentialPlugin.properties** to JService Configuration folder. The location of the JService Configuration folder varies depending on the version of the Policy Controller being used.
   * Windows Policy Controller: [Policy Controller]/jservice/config
   * Java Policy Controller: [Tomcat folder]/nextlabs/dpc/jservice/config
3. Copy **UserAttributeProvider.jar** and **ResourceReferentialPlugin.jar** to a convenient folder. It’s recommended to copy the jar to default JService Jar folder. The default location of JService Jar Folder varies depending on the version of the Policy Controller being used.
   * Windows Policy Controller: [Policy Controller]/jservice/jar
   * Java Policy Controller: [Tomcat folder]/nextlabs/dpc/jservice/jar
4. Copy all the jar file under xlib into the NextLabs Shared Library folder. The location of the Shared Library folder varies depending on the version of the Policy Controller being used.
   * Windows Policy Controller: [Policy Controller]/nextlabs/shared\_lib
   * Java Policy Controller: [Tomcat folder]/nextlabs/shared\_lib
5. Modify the **ResourceReferentialPlugin.properties** and **UserReferentialPlugin.properties.** Please refer to sections 3.3 and 3.4 for more information about how to configure the properties file. Pay close attention to the following settings:
   * The jar-path setting in each properties file must match the location where the corresponding jar files are copied to in step 3.
   * Make sure that the user account used to connect to the database is **a local SQL account** and the default schema of the user account must be mapped to **sch\_DFS**



## Configure User Referential Plugin

Open **UserReferentialPlugin.properties** to configure the plugin

### 3.3.1 General information

|  |  |  |
| --- | --- | --- |
| Property | Meaning | Sample value |
| name | Plugin name. Must not be changed. | UserReferentialPlugin |
| jar-path | The path of the UserReferentialPlugin.jar. | [NextLabs]/PolicyController/jservice/jar/ UserReferentialPlugin.jar |
| description | Description of the plugin. Can leave as default. | User Referential Plugin |
| friendly\_name | Friendly name of the plugin. Can leave as default. | User Referential Plugin |

### 3.3.2 Cache information

|  |  |  |
| --- | --- | --- |
| Property | Meaning | Sample value |
| cache\_heap\_in\_mb | Cache size used to store user information, adjust accordingly to user data size. Size is MB unit. If it is set to too low, cache missed will happened more frequent. Old entry will be remove if cache is full based on first in first in first out. | 2048 |
| cache\_max\_object | Maximum objects in a cache element, if the maximum objects in element is more than setting, the old entry will be discarded based on first in first in first out. | 50000 |
| number\_of\_retries | Number of retries when an exception occurs during a refresh. Default value is 3 | 3 |
| interval\_between\_retries | Time in second between retries when exception occurs. Default value is 30 | 30 |
| time\_to\_live | The maximum period that user cache entry (a user) stays in the cache since last entry to the cache. The format of the property should be **<period>\_<unit>.** Supported units are **SECS, MINS, HRS, DAYS**. When this period has passed for a user, the user will be removed from the cache. The plugin will query the AD for the user and put him back to the cache when the user is asked next time. Default value is 1\_HRS | 1\_HRS |
| expired\_mode | Cache expired mode for user cache, live or purge | purge |
| purge\_time | This setting take effect only when expired\_mode is purge. The hour of the day, when purge triggered. When this is triggered, the user cache will be flushed. The format is hh:MM | 01:00 |
| refresh | This setting take effect when expired\_mode is purge. Plugin will query all the user data from database and put it into the cache after the purge completed. The value can be true/false | true |
| null\_string | The string to return when the result is a NULL object. By commenting out this property, a NULL object will be returned | NO\_DATA |

### 3.3.3 DB information

|  |  |  |
| --- | --- | --- |
| Property | Meaning | Sample value |
| DB\_1\_database\_url | Database URL string, contain hostname, port number and database name | jdbc:sqlserver://GENSQL02W12R2:1433;databaseName=DSF\_TEST |
| DB\_1\_database\_driver\_name | Name of the database driver, for future usage if switch database type | com.microsoft.sqlserver.jdbc.SQLServerDriver |
| DB\_1\_database\_username | Username for database connection, user should be the default schema for database | airbus |
| DB\_1\_database\_password | Password the for database user, enter in plain text and will be encrypted once the PDP started | 123next! |
| DB\_1\_database\_password\_encrypted | Indication for database password is encrypted or not, set to false if you change plain text password | false |
| DB\_1\_database\_schema\_name | Schema of the database, for future usage since MSSQL server does not support this. | sch\_DSF |

|  |  |  |
| --- | --- | --- |
| Property | Meaning | Sample value |
| DB\_1\_key\_attributes | A list of primary and foreign keys used to query the database for additional attributes. Each entry must be separated by a coma (,) and must follow the following format:  case**:**shortname**:**dbColumn   * **case** – either “ci” for case-insensitive or “cs” for case sensitive. Defines the case sensitivity of queries against dbColumn. * **shortname** – an attribute with this name will be created in the plugin cache to store the resource id submitted in the authorization request * **dbColumn –** the database column name that contains the value we want to compare the shortnamewith. **Case sensitive.** | ci:id:obid |
| DB\_1\_attributes\_to\_pull | A list of all user attributes that will be pulled by the plugin **from the USRPD table**. Each entry must be separated by a coma (,) and must follow the following format:  cardinality**:**shortname**:**dbColumn   * **cardinality** – either “single” or “multi” depending if the attribute is single value or multi-value * **shortname** – the attribute name that will be used by policy controller to refer to this attribute. Not case sensitive. * **dbColumn –** the database column name that contains the value we want to pull. **Case sensitive.** | single:service:department,single:company:company,single:country:CountryCode,single:userid:LogonID,single:nationality:Nationality,single:countrybirth:CountryBirth,single:companycountry:employmentlocation  *note: list can be expanded by adding entries for other columns of the* ***USRPD table.*** |
| DB\_1\_link\_attributes\_to\_pull | A list of all user attributes that will be pulled by the plugin **from the USRAT\_PROGT table**. Each entry must be separated by a coma (,) and must follow the following format:  cardinality**:**shortname**:**dbColumn  Refer to DB\_1\_attributes\_to\_pull for definition of cardinality, shortname and dbColumn. | multi:userhcp:progk,multi:userversion:provs  *note: sample value shows full list of valid entries. Additional entries cannot be added without a need for code change.* |

## Configure Resource Referential Plugin

Open **ResourceReferentialPlugin.properties** to configure the plugin

### 3.4.1 General information

|  |  |  |
| --- | --- | --- |
| Property | Meaning | Sample value |
| name | Plugin name. Must not be changed. | ResourceReferentialPlugin |
| jar-path | The path of the ResourceReferentialPlugin.jar. | C:/ProgramFiles/ApacheSoftware Foundation/Tomcat 8.5/nextlabs/dpc/jservice/jar/ ResourceReferentialPlugin.jar |
| description | Description of the plugin. Can leave as default. | Resource Referential Plugin |
| friendly\_name | Friendly name of the plugin. Can leave as default. | Resource Referential Plugin |

### 3.4.2 Cache information

|  |  |  |
| --- | --- | --- |
| Property | Meaning | Sample value |
| cache\_heap\_in\_mb | Cache size used to store user information, adjust accordingly to user data size. Size is MB unit. If it is set to too low, cache missed will happened more frequent. Old entry will be remove if cache is full based on first in first in first out. | 2048 |
| cache\_max\_object | Maximum objects in a cache element, if the maximum objects in element is more than setting, the old entry will be discarded based on first in first in first out. | 50000 |
| number\_of\_retries | Number of retries when an exception occurs during a refresh. Default value is 3 | 3 |
| interval\_between\_retries | Time in second between retries when exception occurs. Default value is 30 | 30 |
| time\_to\_live | The maximum period that user cache entry (a user) stays in the cache since last entry to the cache. The format of the property should be **<period>\_<unit>.** Supported units are **SECS, MINS, HRS, DAYS**. When this period has passed for a user, the user will be removed from the cache. The plugin will query the AD for the user and put him back to the cache when the user is asked next time. Default value is 1\_HRS | 1\_HRS |
| expired\_mode | Cache expired mode for user cache, live or purge | purge |
| purge\_time | This setting take effect when expired\_mode is purge. The hour of the day, when purge triggered. When this triggered, the user cache will be flush. The format is hh:MM | 01:00 |
| refresh | This setting take effect when expired\_mode is purge. Plugin will query all the user data from database and put it into the cache after the purge completed. The value can be true/false | true |
| null\_string | The string to return when the result is a NULL object. By commenting out this property, a NULL object will be returned | NO\_DATA |

### 3.3.3 DB information

|  |  |  |
| --- | --- | --- |
| Property | Meaning | Sample value |
| DB\_1\_database\_url | Database URL string, contain hostname, port number and database name | jdbc:sqlserver://GENSQL02W12R2:1433;databaseName=DSF\_TEST |
| DB\_1\_database\_driver\_name | Name of the database driver, for future usage if switch database type | com.microsoft.sqlserver.jdbc.SQLServerDriver |
| DB\_1\_database\_username | Username for database connection, user should be the default schema for database | airbus |
| DB\_1\_database\_password | Password the for database user, enter in plain text and will be encrypted once the PDP started | 123next! |
| DB\_1\_database\_password\_encrypted | Indication for database password is encrypted or not, set to false if you change plain text password | false |
| DB\_1\_database\_schema\_name | Schema of the database, for future usage since MSSQL server does not support this. | sch\_DSF |

|  |  |  |
| --- | --- | --- |
| Property | Meaning | Sample value |
| DB\_1\_key\_attributes | A list of primary and foreign keys used to query the database for additional attributes. Each entry must be separated by a coma (,) and must follow the following format:  case**:**shortname**:**dbColumn   * **case** – either “ci” for case-insensitive or “cs” for case sensitive. Defines the case sensitivity of queries against dbColumn. * **shortname** – an attribute with this name will be created in the plugin cache to store the resource id submitted in the authorization request * **dbColumn –** the database column name that contains the value we want to compare the shortnamewith. **Case sensitive.** | ci:id:obid  *note: sample value shows full list of valid entries. Additional entries cannot be added without a need for code change.* |
| DB\_1\_attributes\_to\_pull | A list of all user attributes that will be pulled by the plugin **from the DARIT table**. Each entry must be separated by a coma (,) and must follow the following format:  cardinality**:**shortname**:**dbColumn   * **cardinality** – either “single” or “multi” depending if the attribute is single value or multi-value * **shortname** – the attribute name that will be used by policy controller to refer to this attribute. Not case sensitive. * **dbColumn –** the database column name that contains the value we want to pull. **Case sensitive.** | single:excc1:id\_excc1t,single:indcl:id\_indct,single:milcl:id\_milct  *note: sample value shows full list of valid entries. Additional entries cannot be added without a need for code change.* |
| DB\_1\_proghd\_attributes\_to\_pull | The string listing the item attributes which need to be pulled by the plugin from PROGHD table. Attributes are separated by comma and prefixed by cardinality. Suffix will be database column name. Valid cardinality are “single:” and “multi:”. The cardinality of the attributes should follow on the cardinality of such attributes in database design | multi:hcp:progk,multi:hcpversion:provs  *note: sample value shows full list of valid entries. Additional entries cannot be added without a need for code change.* |
| DB\_1\_progit\_attributes\_to\_pull | The string listing the item attributes which need to be pulled by the plugin from PROGIT table. Attributes are separated by comma and prefixed by cardinality. Suffix will be database column name. Valid cardinality are “single:” and “multi:”. The cardinality of the attributes should follow on the cardinality of such attributes in database design | multi:hcp\_indct:id\_indct,multi:hcp\_milct:id\_milct,multi:hcp\_excc:id\_excc1t  *note: sample value shows full list of valid entries. Additional entries cannot be added without a need for code change.* |
| DB\_1\_excc2t\_attributes\_to\_pull | The string listing the item attributes which need to be pulled by the plugin from EXCC2T table. Attributes are separated by comma and prefixed by cardinality. Suffix will be database column name. Valid cardinality are “single:” and “multi:”. The cardinality of the attributes should follow on the cardinality of such attributes in database design | multi:excc2:excc2  *note: sample value shows full list of valid entries. Additional entries cannot be added without a need for code change.* |
| DB\_1\_maex\_attributes\_to\_pull | The string listing the item attributes which need to be pulled by the plugin from MAEX table. Attributes are separated by comma and prefixed by cardinality. Suffix will be database column name. Valid cardinality are “single:” and “multi:”. The cardinality of the attributes should follow on the cardinality of such attributes in database design | multi:excc2\_1:embgr  *note: sample value shows full list of valid entries. Additional entries cannot be added without a need for code change.* |
| DB\_1\_prog\_key\_attributes | The string listing the ID attribute(s) of program in database. Multiple attributes can be listed, separated by comma, prefixed by case sensitivity. Suffix with database column name. If the attribute is prefixed by ‘cs:’, the plugin will do a case-sensitive query for the IDs of that attribute. If the attribute is prefixed by ‘ci:’. the plugin will do a case-insensitive query for the IDs of that attribute. | ci:id:id\_hd  *note: sample value shows full list of valid entries. Additional entries cannot be added without a need for code change.* |
| DB\_1\_prog\_attributes\_to\_pull | The string listing the program attributes which need to be pulled by the plugin from PROGIT table. Attributes are separated by comma and prefixed by cardinality. Suffix will be database column name. Valid cardinality are “single:” and “multi:”. The cardinality of the attributes should follow on the cardinality of such attributes in database design | multi:hcp\_indct:id\_indct,multi:hcp\_milct:id\_milct,multi:hcp\_excc:id\_excc1t  *note: sample value shows full list of valid entries. Additional entries cannot be added without a need for code change.* |

# Troubleshooting

Perform the following steps in the Policy Controllers in order to enable verbose logging.

1. Open logging.properties in [PolicyController]/config
2. Find a line containing com.bluejungle.level and add com.nextlabs.level = FINEST in the bellow line (change the level to FINEST if the com.nextlabs.level is already present)
3. Restart the Policy Controller.

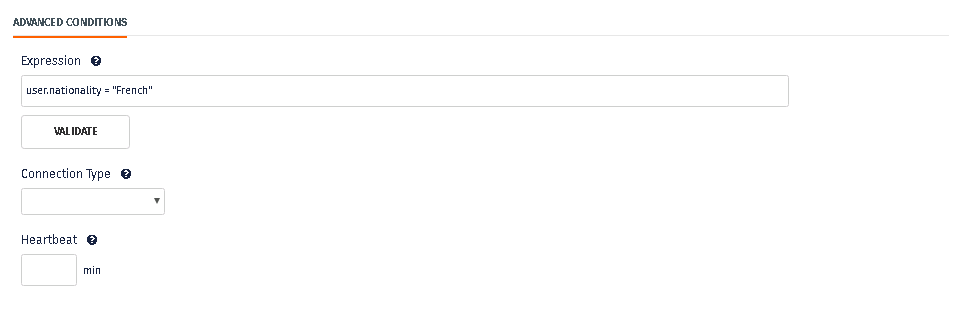
# Appendix

## Creating Policies with the Referential Plugin

The referential plugin can be used in policy evaluation using components and/or advanced conditions.

* + 1. Using Referential Plugin in Advanced Conditions

In order to use the referential plugin using advanced conditions, simply type the conditional expression as shown below.



The syntax for attributes returned by the **user referential plugin** is user.<shortname>

Example:

* user.nationality
* user.country

The syntax for attributes returned by the **resource referential plugin** is:

resource.<resource shortname>.<shortname>

Example

* resource.AHresource.PROGK
* resource.fso.filename

For a complete list of supported shortnames, please refer to Appendix 5.2

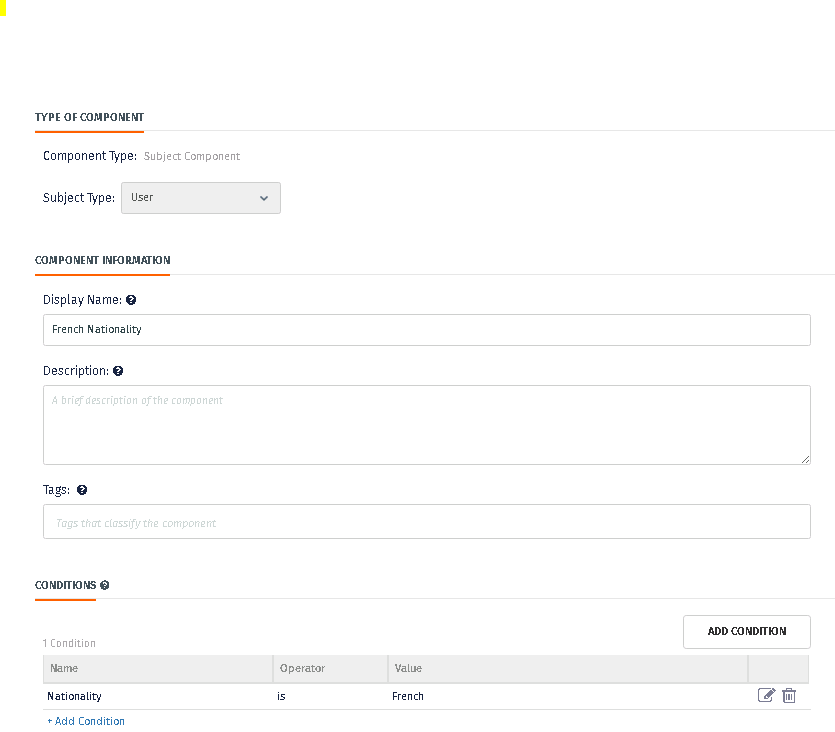
* + 1. Using Referential Plugin in Components

1. Modify the applicable policy models (e.g. user policy model, resource policy model) and add the corresponding attributes. The attribute name provided in the first column will be the name used when defining the criteria in the components. The short name in the second column must match the short name used by the referential plugin.

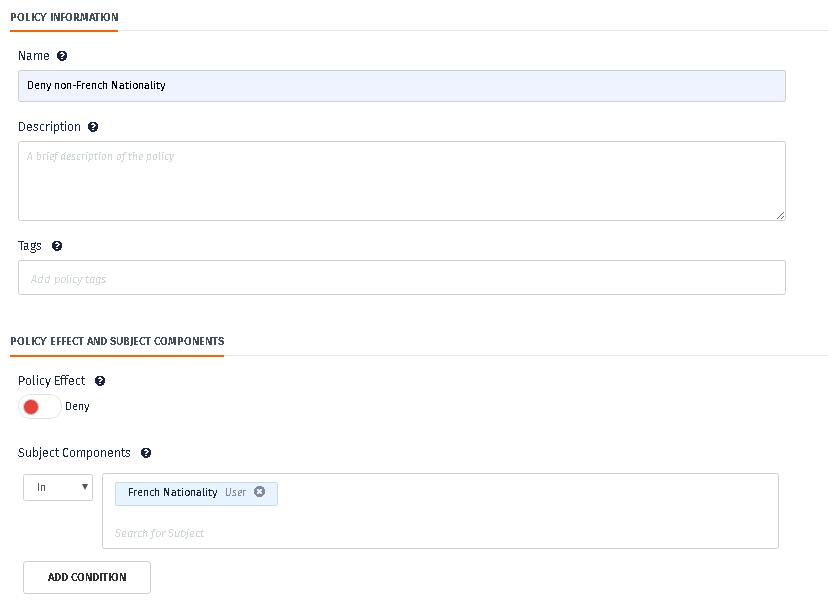


Figure 1 User Policy Model with the additional referential attribute plugins

1. Create a component and use the added attributes to define the component criteria.



1. Use the created component in the policy.



* + 1. Enrollment and Referential Plugin

Avoid using the same attribute names in enrollment that the referential plugin already uses and vice versa.

Enrolment is the process of pulling attributes from an external data source (e.g. Active Directory), storing it in the Control Center database and precompiling it in the bundle file. During the process of enrolment, an external attribute will be mapped to a Control Center attribute (also known as enrolled attribute. For example, the Active Directory variable “c” can be mapped to an enrolled attribute called “country”.

The referential plugin cannot be used in policy evaluation **using components,** if the component uses a referential plugin short name similar to an enrolled attribute. For example, using a component based on the “country” short name should be avoided because an enrolled attribute “country” exists. In such scenario, the policy evaluation will only consider the enrolled attribute and not the value returned by the referential plugin. In this scenario, in order to force the policy evaluation to use the referential plugin to retrieve “country”, an advanced condition must be used.

## Shortnames used by the Referential Plugins

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Policy model | Short name | Referential table | dbColumn | Cardinality |
| Resources | excc1 | DARIT | ID\_EXCC1T | Single value |
| Resources | indcl | DARIT | ID\_INDCT | Single value |
| Resources | milcl | DARIT | ID\_MILCT | Single value |
| Resources | hcp | PROGHD | PROGK | Multiple value |
| Resources | hcpversion | PROGHD | PROVS | Multiple value |
| Resources | hcp\_indct | PROGIT | ID\_INDCT | Multiple value |
| Resources | hcp\_milct | PROGIT | ID\_MILCT | Multiple value |
| Resources | hcp\_excc | PROGIT | ID\_EXCC1T | Multiple value |
| Resources RTYPE PROGRAM | hcp\_indct | PROGIT | ID\_INDCT | Multiple value |
| Resources  RTYPE PROGRAM | hcp\_milct | PROGIT | ID\_MILCT | Multiple value |
| Resources  RTYPE PROGRAM | hcp\_excc | PROGIT | ID\_EXCC1T | Multiple value |
| Resources | excc2 | MAEX | EMBGR | Multiple value |
| EXCC2T | EXCC2 | Multiple value |
| Users | userid | USRPD | LogonID | Single value |
| Users | Service | USRPD | Department | Single value |
| Users | Company | USRPD | Company | Single value |
| Users | country | USRPD | CountryCode | Single value |
| Users | nationality | USRPD | Nationality | Single value |
| Users | countrybirth | USRPD | CountryBirth | Single value |
| Users | companycountry | USRPD | Employmentlocation | Single value |
| Users | userhcp | USRAT\_PROGT | PROGK | Multiple value |
| Users | userversion | USRAT\_PROGT | PROVS | Multiple value |

## Comparing NULL Strings

For single value attributes, the Referential plugin will return the value specified in the null\_string configuration. By default it is set to “NO\_DATA”. As such, in order to determine if the value of a single-value attribute is null, it must be compared with NO\_DATA. For example:

user.country = “NO\_DATA”

Please note that the user and resource null\_string attributes are configured separately. It is possible to have a different null\_string setting for the user and the resource plugin.

For multi-value attributes, *null*, without quotation marks can be used. For example:

user.userhcp = null