Server User Attribute Provider

v1.2

Contents

[Overview 1](#_Toc500248841)

[Server User Attribute Provider 1](#_Toc500248842)

[Functionalities 1](#_Toc500248843)

[Query user and group attributes from AD 1](#_Toc500248844)

[Maintain an in-memory cache of user attributes. 1](#_Toc500248845)

[Purge user from cache on demand 1](#_Toc500248846)

[Component 1](#_Toc500248847)

[Package 1](#_Toc500248848)

[Deploy 1](#_Toc500248849)

[Configure 1](#_Toc500248850)

[General information 1](#_Toc500248851)

[Cache information 1](#_Toc500248852)

[LDAP information 1](#_Toc500248853)

[Troubleshooting 1](#_Toc500248854)

[Appendix 1](#_Toc500248855)

[Server User Attribute Provider, Enforcer and Enrollment 1](#_Toc500248856)

[Key store 1](#_Toc500248857)

[Trust store 1](#_Toc500248858)

[Group 1](#_Toc500248859)

[Password Encryption 1](#_Toc500248860)

# Overview

## Server User Attribute Provider

Server User Attribute Provider is a Dynamic User Attribute Provider plugin designed for CE Policy Controller/Java Policy Controller to query and cache user attributes from Active Directory (AD) and LDIF Files. The plugin can provide user attribute data on-demand for Policy Controller evaluation when the attribute is not available in the enforcer request, by hitting its own cache or querying AD directly.

# Functionalities

## Query user and group attributes from AD or LDIF

* One or more domain can be configured.
* Specify the domain type as LDAP for AD and LDIF for LDIF.
* A user or a group 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 AD for ID can work with one PDP.
* **All IDs must be unique across all domains configured.**
* More than one user attributes or group attributes 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.
* A LDAP user search filter and LDAP user search base can be configured in the plugin’s properties file to limit the user returned. Search base is not applicable for LDIF and only simple filters like (objectclass=user) is applicable.
* A LDAP group search filter and LDAP group search base can be configured in the plugin’s properties file to limit the group returned. Search base is not applicable for LDIF and only simple filters like (objectclass=group) is applicable.
* (Only applicable for LDAP domain type) Customer can choose whether to retrieve disabled accounts or not.
* The plugin would return EvalValue.NULL if user or group 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 AD using connection pool. The connection pool settings are configurable.
* The plugin loads all user information from the LDIF file in to the cache and also reloads the cache based on the cache\_refresh\_period property value. If domain type is LDIF, make sure that the cache\_refresh\_period value is not set to 0.
* The plugin supports connecting to AD using SSL. In order to configure SSL, a trust store containing the AD certificate needs to be specified in the plugin’s properties.
* The plugin works in 2 way SSL communication, user will need to create the keystore to store the private key and distribute the certificate to AD server manually for 2 way SSL communication.

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

* On PDP start/restart, the plugin would query for all users and all groups from AD and store their attributes in this cache. If user configure cache\_refresh\_period =0, the plugin will not pull the users and groups information at startup. **If domain type is LDIF, do not set to the value of cache\_refresh\_period to 0 because querying user attributes dynamically from an LDIF file is not supported.**
* 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 AD and put back the user or group to cache.
* 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
* 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

## Purge user from cache on demand

The plugin comes with a C# console application called PurgeUser. This application can be run using command tool. By supplying an identifier, the application will purge such user or group from the cache. When supplying **all** instead of the id, the whole cache will be refreshed. The command to use is:

PurgeUser.exe <id>

# Component

## Package

* ServerUserAttributeProvider.jar
* ServerUserAttributeProvider.properties
* suap-truststore.jks

## Deploy

To deploy the plugin in the Policy Controller

* Copy **ServerUserAttributeProvider.properties and suap-truststore.jks** to **[PolicyController]/jservice/config (dpc/jservice/config in Java Policy Controller)**
* Copy **ServerUserAttributeProvider.jar** to a convenient folder. It’s recommended to copy the jar to **[PolicyController]/jservice/jar (dpc/jservice/jar in Java Policy Controller)**
* **For CE Policy Controller, copy the xlib/ldap.jar file to [PolicyController]/jre/lib/ext and for Java Policy Controller, copy the xlib/ldap.jar file to [Tomcat]/Nextlabs/Shared\_lib in Java Policy Controller.**

## Configure

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

### General information

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

### Cache information

|  |  |  |
| --- | --- | --- |
| Property | Meaning | Sample value |
| time\_to\_live | The maximum period that a 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\_DAYS | 1\_DAYS |
| cache\_refresh\_start\_time | The timestamp indicates when the refresh process starts. If this timestamp has passed when the Policy Controller starts, the cache refresh process will start immediately. | 22/06/2017 11:04:00 |
| cache\_refresh\_start\_time\_format | The format of the cache\_refresh\_start\_time timestamp | dd/MM/yyyy HH:mm:SS |
| cache\_refresh\_period | The period between refreshes, starts from the cache\_refresh\_start timestamp. Supported units are SECS, MINS, HRS and DAYS. Default value is 0 | 1\_DAYS |
| 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. | 128 |
| cache\_max\_object | Maximum objects in a cache element, if the maximum objects in element is more than setting, it will not get cache. | 5000 |
| 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 |

### LDAP information

|  |  |  |
| --- | --- | --- |
| Property | Meaning | Sample value |
| pool\_max\_size | The maximum number of connections per connection identity that can be maintained concurrently in the pool. Default value is 20. | 20 |
| pool\_pref\_size | The preferred number of connections per connection identity that should be maintained concurrently in the pool. Default value is 10. | 10 |
| pool\_init\_size | The number of connections per connection identity to create when initially creating a connection for the identity. Default value is 1. | 1 |
| pool\_time\_out | The number of milliseconds that an idle connection may remain in the pool without being closed and removed from the pool. Default value is 30000. | 360000 |
| pool\_debug | The level of debug output to produce. Valid values are “none”, "fine" (trace connection creation and removal) and "all"  (all debugging information). | false |
| key\_store | The location of the keystore containing the private key of the plugin, which would be used for SSL connection. This is needed for 2 way SSL communication. The file will need to be created manually by user. The format should be in jks format. Optional for one way SSL communication | C:/Program Files/NextLabs/Policy Controller/jservice/config/suap-keystore.jks |
| key\_store\_pass | The password of the key store, in encrypted form. The password should be in encrypted format using NextLabs crypto tool which can be from in control center. | sa549f6ba05c840e5f43ef63e06a8ae1a |
| trust\_store | The location of the trust store containing the certificate of the Active Directory, which would be used for SSL connection. The **suap-trustore.jks** in the package is an empty trust store that can be used. | C:/Program Files/NextLabs/Policy Controller/jservice/config/suap-truststore.jks |
| trust\_store\_pass | The password of the trust store, in encrypted form. The password for the suap-truststore.jks is **sa549f6ba05c840e5f43ef63e06a8ae1a**, which is the encrypted string for **123blue!** |  |
| paging\_size | The size of each batch of users returned in a LDAP search. By default LDAP restricts this number to be 1000 maximum. The number in this properties file cannot be larger than the setting in the AD. Default value is 1000. | 1000 |
| profile\_names | The string listing the unique names of different domains. These names can be self-identified but they must be unique | DOMAIN\_1,DOMAIN\_2 |

Each domain should have the following properties, where the property names should be prefixed by the “<domain name>\_”, e.g. DOMAIN\_1\_host, DOMAIN\_1\_port, DOMAIN\_1\_user\_search\_base

|  |  |  |
| --- | --- | --- |
| Property | Meaning | Sample value |
| Type | The LDAP type for querying user information from AD or LDIF for querying user information from LDIF. | LDAP or LDIF |
| host | (Only for LDAP) The host name or IP address of the Active Directory. |  |
| port | (Only for LDAP) The port of the Active Directory. | 389 for non SSL, 636 for SSL |
| ssl | (Only for LDAP) Boolean flag to indicate whether to connect to the AD using SSL channel or not. When set to true, the certificate of the AD must be imported into the trust store used by the plugin. | true |
| authentication | (Only for LDAP) Authentication mode for the AD, it can be simple or none, for none the username and password will be optional. | simple |
| username | (Only for LDAP) The account user name used to query users This account should have read permission on all user objects in the Active Directory. |  |
| password | (Only for LDAP) The password of the account in encrypted format. |  |
| get\_disabled\_accounts | (Only for LDAP) The Boolean flag to decide if the plugin should retrieve disabled accounts in AD or not. | false |
| user\_search\_base | (Only for LDAP) The bases of the search for users in Active Directory. Multiple search bases can be specified, separated by semicolon. | OU=dir1,DC=company,DC=com;OU=dir2,DC=company,DC=com |
| user\_filter | The search filter which would be applied to all user searches executed by the plugin. | (&(objectClass=user)(objectClass=person)) |
| user\_key\_attributes | The string listing the ID attribute(s) of users in AD. Multiple attributes can be listed, separated by comma, prefixed by case sensitivity. 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. Case-insensitive query is useful when the user ID is case-converted by the enforcer or the enforced application, such as Exchange Server, which makes the ID value passed by the enforcer into the Policy Controller different from the value provided by the AD. | ci:mail,cs:windowsSID,ci:sAMAccountName |
| user\_attributes\_to\_pull | The string listing the user attributes which need to be pulled by the plugin. Attributes are separated by comma and prefixed by cardinality. Valid cardinality are “single:” and “multi:”. The cardinality of the attributes should follow on the cardinality of such attributes in AD. | multi:citizenship,single:department |
| group\_search\_base | (Only for LDAP) The base of the search for groups in Active Directory. Multiple search bases can be specified, separated by semicolon. | OU=dir1,DC=company,DC=com;OU=dir2,DC=company,DC=com |
| group\_filter | The search filter which would be applied to all group searches executed by the plugin | (&(objectClass=group)(proxyAddresses=\*)) |
| group\_key\_attributes | The string listing the ID attribute(s) of groups in AD. Multiple attributes can be listed, separated by comma, prefixed by cardinality. 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. Case-insensitive query is useful when the user ID is case-converted by the enforcer or the enforced application, such as Exchange Server, which makes the ID value passed into the Policy Controller different from the value provided by the AD. It is recommended to use **mail** attribute as the group’s key attribute | ci:mail |
| group\_attributes\_to\_pull | The string listing the group attributes which need to be pulled by the plugin. Attributes are separated by comma and prefixed by cardinality. Valid cardinality are “single:” and “multi:”. The cardinality of the attributes should follow on the cardinality of such attributes in AD. | single:department |
| file\_path | (Only for LDIF) This points to path of the LDIF file to read user information. |  |

# Troubleshooting

Open **logging.properties** in **[PolicyController]/config**

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)

Restart the Policy Controller.

# Appendix

## Server User Attribute Provider, Enforcer and Enrollment

A User Attribute Provider will only be triggered for an attribute if that attribute value is not provided by the enforcer.

With enrollment, any condition in components involving enrolled attributes will be precompiled by the Control Center. As such, policy is matched based on the ID of the user, not by any condition on any attribute. Hence, the Server User Attribute Provider will only be triggered in the advanced condition block, or when the policy asks for an unenrolled attribute in components.

Without enrollment, the Server User Attribute Provider will always be triggered in both components and advanced condition block when the policy asks for an attribute that is not provided by the enforcer.

## Key store

A key store of an application (A) is a container holding all private key of all applications/servers (A). From the private key in this store, a certificate can be extract and import to application (B) trust store so that application(B) can treated application(A) as trusted client.

The **suap-keystore.jks** will need to created manually by user to enable 2 way SSL communicaition. In order to create the key store, the **keytool** utility of Java or the application **KeyStore Explorer** (<http://keystore-explorer.org/>) can be used. It is not compulsory to use the suap-keystore.jks. It is needed when 2 way SSL communication is in place.

Comment out this portion if 2 way SSL is not in-use.

## Trust store

A trust store of an application (A) is a container holding all certificates of all applications/servers (B) that this application (A) should trust. By trusting an application or a server (B), this application (A) can send requests to (B) using one-way SSL channel.

The **suap-truststore.jks** is an empty trust store packaged with the plugin. To use this trust store, all certificates used for SSL of the Active Directory needs to be imported to this trust store. In order to import a certificate to a trust store, the **keytool** utility of Java or the application **KeyStore Explorer** (<http://keystore-explorer.org/>) can be used. The password for the suap-truststore.jks is **123blue!**

It is not compulsory to use the suap-truststore.jks. An existing trust store with the necessary certificates can be used as well.

## Group

Group is supported primarily for Exchange Enforcer. A group can be considered as a single recipient with its own attribute.

In Active Directory, an Exchange group can be a Security Group or a Distribution Group. By adding the condition **(!(groupType:1.2.840.113556.1.4.803:=2147483648))** in the group filter, we can exclude Security Group from the plugin’s query.

An AD group used as an Exchange group generally needs to have a default email address specified in the **mail** attribute in AD. As such, it is recommended that **mail** is used as the key attribute for group. In addition, Exchange Groups normally have one or more email addresses in the **proxyAddresses** attribute. Hence, it is a good idea to add **(proxyAddresses=\*)** in the group filter.

## Password Encryption

To create an encrypted password, a tool located at **[PolicyServer]/tools/crypt/mkpassword.bat** can be used. A sample command is: **mkpassword.bat –password password** with the result of **scc1d86db33184b96278786ce58a47957**