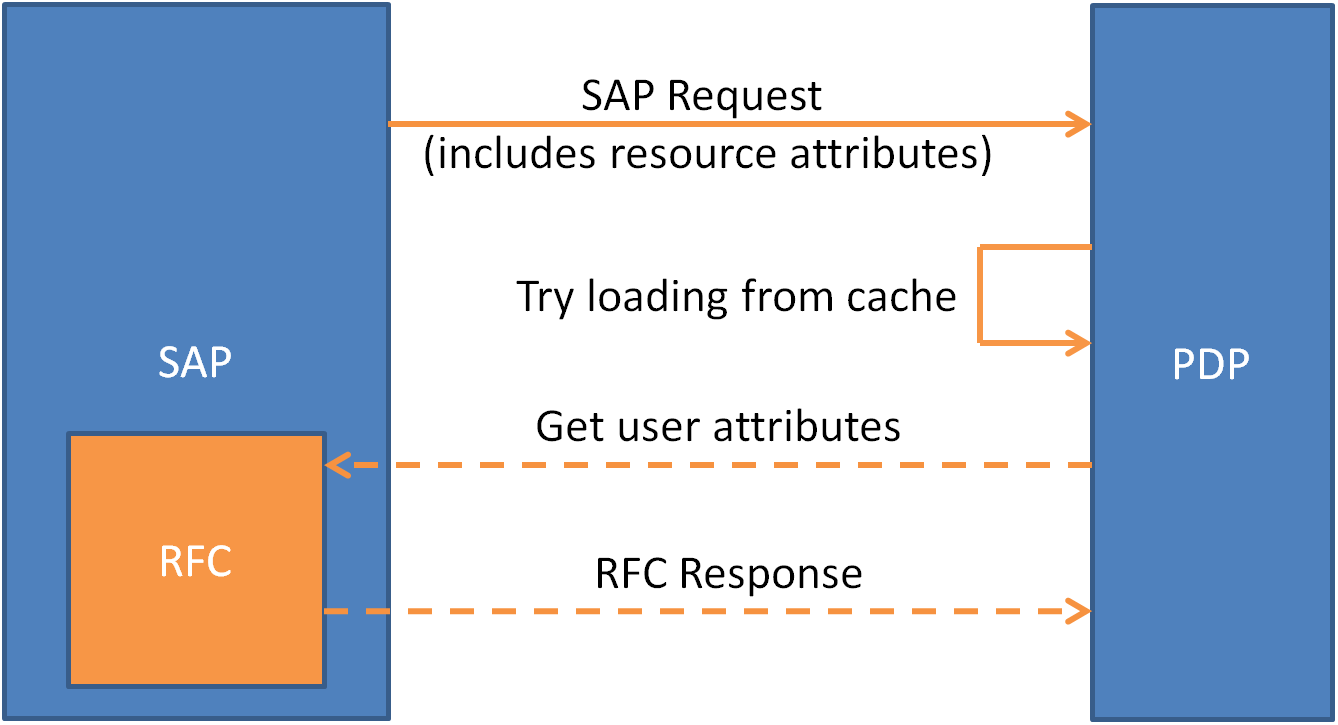
Cache SAP User Attribute Provider Plugin

Developers Guide

# Overview

## User Attribute Provider with Cache



## Invoking Sequence

1. Initial SAP request does not include user.mrparea.
2. Policy Controller requires value of user.mrparea for policy evaluation.
3. Policy Controller invokes user attribute plugin to obtain user.mrparea value.
4. User Attribute Provider plugin checks cache for user.mrparea.
5. If available, cached value of user.mrparea is returned.
6. Otherwise, the User Attribute Provider plugin makes Remote Function Call to SAP system to retrieve user.mrparea.

## Behavior

This User Attribute Provider plugin, as with the normal attribute provide plugins, is triggered by the Policy Controller when a required attribute value is unknown. Triggering the plugin is not in the control of the plugin itself.

However, this plugin will only process request for attribute of mrparea, in which proper values will be returned to the Policy Controller. Otherwise, the plugin will return an empty Multivalue to the Policy Controller.

Also, by default, the plugin will return an empty Multivalue to the Policy Controller when encountering Errors, Exceptions and Failures from calling the SAP Remote Function Calls, retrieving the values from the SAP system or parsing the retrieved values.

Policies should be designed and written in a way that takes care of the condition of an empty Multivalue to cater for unexpected errors and exceptions.

The flow of the User Attribute Provider plugin can be summarized in the diagram below.



Figure - Plugin Flow

# Setting Up

## Software Requirements

This plugin is compatible with the following NextLabs Products:

* NextLabs Control Center 7.7 and above
* NextLabs Policy Controller 7.7 and above

For development setup of the plugin, the following softwares are required:

* Java Development Kit (JDK) 1.7.0\_75
* Eclipse IDE
* Perforce P4V Client
* Apache Ant 1.9.4
* Cygwin Terminal
* NextLabs Main Externals (available on Perforce)

## Getting from Repository

The NextLabs Professional Services team uses the Bitbucket Repository for plugin development.

The path to the plugin on the repository is as follow:

<https://bitbucket.org/nxtlbs-devops/swaf_cachesapuserattributeprovider>

## Dependencies

The Cache SAP User Attribute Provider plugin requires several dependencies to function, as shown in the table below.

|  |  |  |
| --- | --- | --- |
| **Library** | **Version** | **Functionality** |
| sapjco | 3.0.11 (NTia32) | Provides JCo Connectivity to the SAP System. |
| ehcache | 1.1 | Provides local caching capabilities. |
| Mockito (optional) | 1.10.19 | Allows creation of mock objects for unit tests. |

These dependencies can be imported into the project automatically using build scripts from the NextLabs external library collection. More information about the build script is available in the later sections.

Before the dependencies can be imported, they have to be available in the local Perforce workspace. If the dependency libraries are no longer available on Perforce, do consult the team for the new location.

Perform the following to obtain the required dependencies:

1. Launch Perforce Client P4V.
2. Enter the appropriate credentials.  
   Contact the Professional Services Team Lead if you are not provided with the credentials.
3. Go to the //depot/main/external directory.
4. Right click on the relevant dependency name while holding the control key.
   1. apache-ant/apache-ant-1.9.4
   2. ehcache/ehcache-1.1
   3. j2sdk/jdk1.7.0\_75
   4. sapjco/3.0.11/NTia32
5. Click **Get Latest Revision.**
6. The dependencies are downloaded to your local Perforce workspace.

## Configuring Build Files

Prior to building the plugin, configurations have to be made to the build files to allow the build to match the local development environment. There are two files required to be modified as shown in the table below.

|  |  |
| --- | --- |
| **File** | **Modification** |
| configure | Change line 217 and 218 to match the local network’s release artifact. |
|  | |
| buildDev | Change line 27 to match the main external directory of your local Perforce workspace. |
| C:\Users\ypoh\AppData\Local\Microsoft\Windows\INetCache\Content.Word\dev builddev.png | |

## Build Project

To build the project, a Linux shell script is used. Therefore, a Linux based terminal or Cygwin (when running on Windows) is required.

Building the project is a two-step process. First, the dependencies library files have to be imported into the project. Next, the project can be compiled and packaged for deployment.

To build the project, follow the steps below:

1. Launch Cygwin.
2. Navigate to the project directory.
3. Run buildDev with makexlib  
   ./buildDev --makexlib
4. Dependencies library files are imported into the xlib folder in the project directory.
5. Run buildDev  
   ./buildDev
6. Project is built into the build folder in the project directory.
7. Project is packaged into a zip file for deployment.

## Checking In

To check in the project to Perforce, there are several files and folders that should not be checked in. The files and folders are listed as follow:

* build/
* lib/
* xlib/

## Installation

After building the plugin, the build artifacts will be packed in a zip file inside the build folder. The zip file contains the following files:

* Jar file (CacheSAPUserAttributeProvider.jar)
* Configuration file (CacheSAPUserAttributeProvider.properties)
* This readme document (ReadMe.docx)

Follow these steps to Install and Deploy the user attribute provider plugin into the Policy Controller.

1. Stop the Policy Controller.
2. Copy CacheSAPUserAttributeProvider.jar to [PC Home]\jservice\jar\
3. Copy CacheSAPUserAttributeProvider.properties to [PC Home]\jservice\config\
4. Update the configuration in the Configuration file to match the deployment system.
5. Start the Policy Controller.
6. The user attribute provider plugin is now installed and deployed.

## Uninstallation

Follow these steps to uninstall the user attribute provider plugin from the Policy Controller.

1. Stop the Policy Controller.
2. Remove CacheSAPUserAttributeProvider.jar from [PC Home]\jservice\jar\
3. Remove CacheSAPUserAttributeProvider.properties from [PC Home]\jservice\config\
4. Start the Policy Controller.
5. The user attribute provider plugin is now uninstalled.

# Architecture of Plugin



Figure - Plugin Architecture

The Cache SAP User Attribute Provider plugin features the capability to communicate with the SAP system to retrieve additional user attribute (MRP Area) that is not sent along with the initial authorization request to the PDP.

This communication is handled via the JCo function handler that is established on the SAP system. The retrieved information from the SAP system is then cached locally in the PDP to cater for subsequent use, thereby reducing the response time in the subsequent requests.

## JCoFunctionHandler

The JCoFunctionHandler is a wrapper around the sapjco3 API. This handler provides a simplified approach to perform the **remote function call (RFC)** to the **SAP JCo function**.

For the SAP JCo function, the function arguments are passed in through the import parameters. The import parameter can be populated at the point of making the function call.

By default, the SAP JCo function returns values in multiple ways. The following are the three different ways values may be returned to the plugin:

* RFC Export
* RFC Changing
* RFC Table

To simplify the experience of retrieving data from the JCo function, the return values are mapped into a Java HashMap object. This allows easy manipulation and consumption of the data without the need to understand the complex JCo Function library API.

All these complex operations are simplified into two simple steps:

1. Call Function and input arguments.
2. Get RFC output as Hashmap from Export, Changing or Table.

## LocalCacheManager

To reduce the average response time, the return values of the JCo Function are cached locally for the subsequent usage.

The Local Cache Manager serves as an interface for the cache implementation, keeping the options open for different implementation in the future.

Currently, under the hood of the Local Cache Manager Implementation is the ehcache library. Ehcache version 1.1 is selected to ensure consistency with the version used in the PDP.

Usage of the Local Cache Manager is simple with only two methods:

* getFromCache
* putToCache

The cache works as key value pairs with the username as the key to the list of associated MRP Area as the value.

# Understanding the Logs

To test and debug the User Attribute Provider plugin, the log serves as the primary tool to understand the condition, state and values returned by the plugin.

The following shows the list of important log statements provided by the User Attribute Provider plugin and their significant contribution in test and debug sessions.

To ensure a more thorough log, use the **FINEST** logging level.

## Cache Status

To check whether the cache is enabled and being used, the following logs can be referenced.

cache_enabled

Figure - Cache Enabled

cache_disabled

Figure - Cached Disabled

This should reflect the same configuration as configured in the CacheSAPUserAttributeProvider.properties file, under the use\_cache property.

## Cache Configuration

To check the cache configuration loading status, the following shows the values (in seconds) loaded into the plugin. It should match the value stated in the properties file.

cache_config

Figure - Cache Configuration

## Loading from SAP or Cache

To ensure the Cache is working correctly, the plugin can be invoked multiple times at carefully selected intervals to check the source of user attribute data. The time (highlighted in the figures) can be used to check against the expected expiration time of the cached item for proper behavior. The source of data may be either the SAP system (via Remote Function Call) or the Cache as shown in figures below.

time_sap

Figure - Loading from SAP

time_cache

Figure - Loading from Cache

If working correctly, the duration between two calls to the SAP system should be more than the duration set in the configuration file, under the cache\_duration property. Do note that this property is defined in **number of seconds**.

## Raw MRP Area List from SAP (XML)

For checking the MRP area list returned from the SAP system, the following log item can be referenced.

xml_data

Figure - Raw XML Data from SAP

This log item shows the raw XML data returned from the SAP system.  
The “MDLG\_OUT” field is expected to exist for the plugin to extract the list of MRP areas from this XML data.

## Parsed MRP Area List from SAP or Cache

After checking the raw XML data, the following figure shows the parsed MRP Area list. This can be used to check against the raw data for missing or incorrect values.

parsed_data

Figure - Parsed MRP Area

## Processed MRP Area Multivalue for Policy Controller Evaluation

Finally, to return the MRP Area list to the Policy Controller for evaluation, the data has to be converted to a Nextlabs proprietary Multivalue item.

The following figure shows the log item of the Multivalue content after the conversion from the generic Java ArrayList.

provider_data

Figure - Processed MRP Area Multivalue

This is the final value that will be returned to the Policy Controller for evaluation.

## Time Taken

To determine the time taken by the User Attribute Provider plugin, the total time taken in the plugin is logged. This is measured from the start of the plugin to the end of the plugin, right before returning the final value to the Policy Controller.

Also, the attribute name requested by the Policy Controller is also included to assist debugging and fine tuning as the User Attribute Provider plugin will only handle the mrparea attribute while returning an empty Multivalue otherwise.

C:\Users\ypoh\AppData\Local\Microsoft\Windows\INetCache\Content.Word\time_taken_mrp_area.png

Figure - TIme Taken (mrparea)

time_taken_others

Figure - Time Taken (other attributes)

## Attribute Handling

To further confirm that the attribute name requested is not mrparea, the following log item can be referenced. With this, the plugin will return an empty Multivalue to the Policy Controller.

not_mrp_area

Figure - Not MRP Area