

# NetIQ Access Manager Data Extension Example for the External Attribute Source Policy Readme

Data extension examples include:

- NameAttributeFromMailIDFactory.java
- NameAttributeFromMailID.java

## Purpose

This example illustrates how an External Attribute Source policy retrieves information from external sources.

It provides details about:

- How to configure and install the External Attribute Source Data policy extension in the Administration Console.
- Implementation details of the Data policy extension factory and extension classes.
- How to use the information retrieved from the External Attribute Source policies as shared secret. It also explains how to use that shared secret to configure other policies or use them in the Identity servers to retrieve attributes from external sources.

## Functionality

This External Attribute Source policy extension example extracts the name/ ID from the email ID of a user. For example, if [myname@novell.com](mailto:myname@novell.com) is provided as input, then *myname* will be returned. This example retrieves the email ID of the user from the user store. In any practical scenario, implementation logic can be changed to retrieve attributes from any external sources such as external database, legacy systems, and Web services.

## Implementation Details

- The *Data Policy Extension* example consists of two Java classes:

**NameAttributeFromMailIDFactory.java:** Implements the `com.novell.nxpe.NxpeContextDataElementFactory` interface. This factory creates the `NameAttributeFromMailID` objects.

**NameAttributeFromMailID.java:** Implements the `com.novell.nxpe.NxpeContextDataElement` interface. This interface contains the methods required to create a context data element that will be used to retrieve data from external sources.

- This is an example of External Attribute Source policy extension. These policies extensions are always *Data* policy extensions. (As of Access Manager 3.2)

- This class needs a configuration parameter: email ID of the user.  
Configuration parameters are set for a policy extension through the Administration Console. In the policy extension, the values for these configuration parameters are retrieved at the time of evaluation from the NxpelInformationContext object that is sent by the policy engine.
- The `getValue` method in the `NxpeContextDataElement` interface is called by the policy engine when a request triggers a policy evaluation. It executes the logic present in the policy extension and returns the results.
- The `com.novell.nxpe.CustomParameter` class is used in the external attribute source policy extension class to retrieve the name of the external attribute from the policy Engine.

For more information on the implementation, see comments in the `NameAttributeFromMailIDFactory` and `NameAttributeFromMailID` classes and [NetIQ Access Manager 3.2 Developer Kit](#).

## Installation

Before you start the installation, you need the following:

- The following Java classes and `nxpe.jar`:  
`NameAttributeFromMailID.java`  
`NameAttributeFromMailIDFactory.java`

For information on how to obtain `nxpe.jar`, see section 4.1.1 Prerequisites in the NetIQ Access Manager Developer Kit 3.2.

Download `novell-nacm3_2.tar.gz` (SDK tar) from [http://www.novell.com/developer/ndk/novell\\_access\\_manager\\_developer\\_tools\\_and\\_examples.html](http://www.novell.com/developer/ndk/novell_access_manager_developer_tools_and_examples.html).

- Java SDK (jdk1.6) to compile the Java sources.
- Apache Ant.
- An Access Manager setup with the Identity Server, Access Gateway, and policies.
- The user must have an email ID set in the user store.
- A basic knowledge of Java programming.

## Setup

1. Extract the SDK tar (`novell-nacm3_2.tar.gz`). Go to `nacm-3_2_0 > samples-jar` folder. The `PolicyExtensions.jar` will be present at this location. This jar contains all the policy extension example's compiled classes. This jar can be used for this example. If you want to customize this example before installation then modify its java source, compile, and generate a Jar file.

Steps :

Go to `nacm-3_2_0 > samples > PolicyExtension > ExternalAttributeSource_Example` and run the `build.xml`.

This will generate the `PolicyExtensions.jar` in the same location where `build.xml` is available.

To run `build.xml`:

You need to copy nxpe.jar in the nacm-3\_2 > samples > PolicyExtension > nxpe folder before running build.xml.

Note the location where Java and Ant are installed on your system. Set them in the Path environment variable and run the *ant dist* command.

Run the following commands in the Linux environment:

- a) export ANT\_HOME=/usr/apache-ant-1.8.1<set it as per your environment>
- b) export JAVA\_HOME=/usr/java/jdk1.6.0\_27<set it as per your environment>
- c) export PATH=\$ANT\_HOME/bin:\$JAVA\_HOME/bin:\$PATH
- d) ant all

2. Create a policy extension of the type *Identity Server: External Attribute Source*.

For details, see [Installing the Extension on the Administration Console](#) in the [NetIQ Access Manager 3.2 Policy Guide](#).

2.1. Creating a new policy extension:

- a) Name: ExternalAttrSource\_EmailExample
- b) Type: Identity Server: External Attribute Source
- c) Class Name: com.novell.nam.custom.policy.data.NameAttributeFromMailIDFactory
- d) File Name: name of the jar file created in step 1

The screenshot shows a 'New' dialog box with the following fields and values:

- Name: ExternalAttrSource\_EmailExample
- Description: Fetches name or ID present in the I
- Policy Type: Identity Server: External Attribu (dropdown)
- Type: Data (dropdown)
- Class Name: novell.nam.custom.policy.data.NameA (e.g. com.test.MyClass)
- File Name: PolicyExtensions (dropdown)

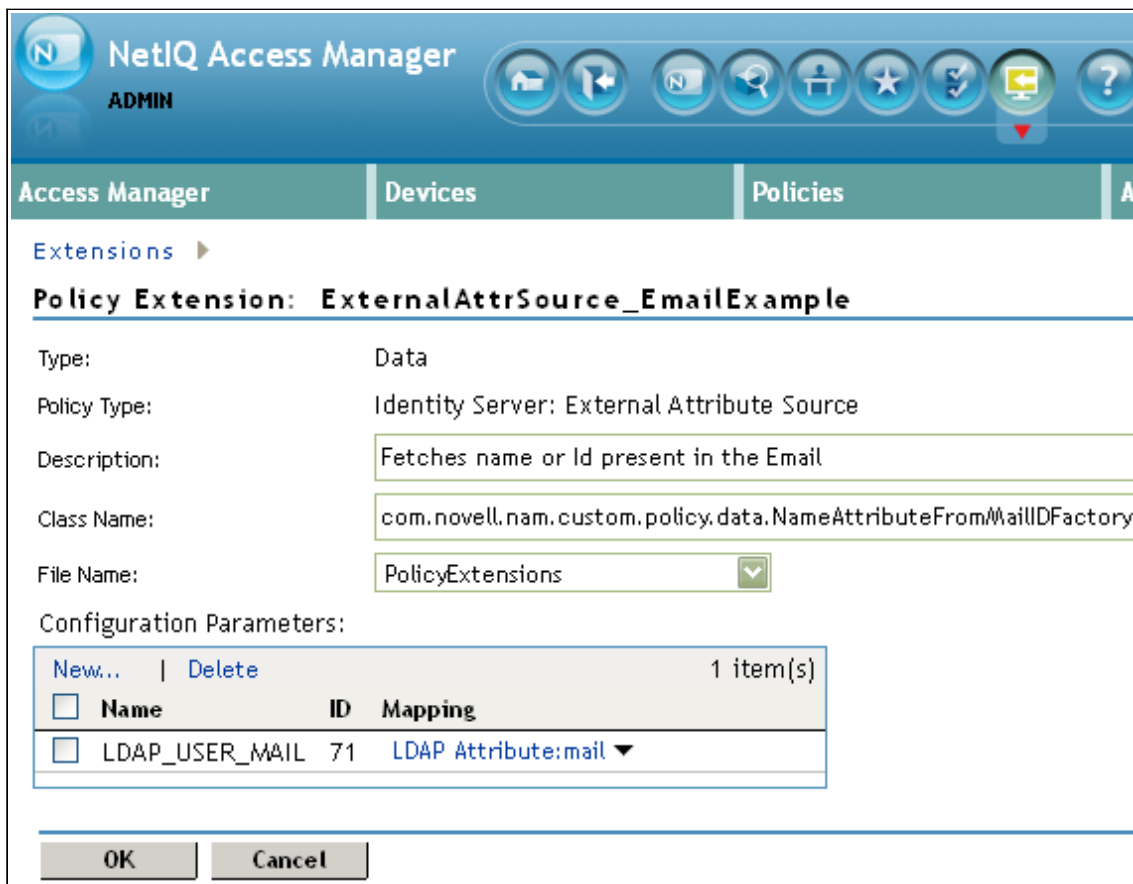
Note: If the .jar file containing the class is not listed, you must upload it. To upload a .jar file, select Upload.

2.2. Policy extensions configuration parameters: Configuration parameters are sent to the policy extension for execution.

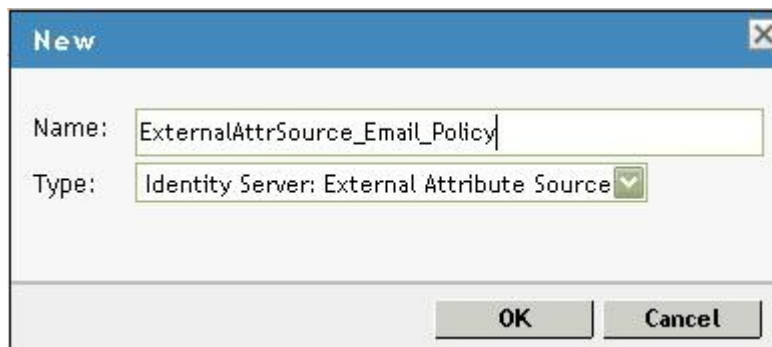
- a) Name: LDAP\_USER\_MAIL (Name can be anything)
- b) ID: 71 (This must match the parameter ID mentioned in the policy extension implementation in code)
- c) Corresponding extract from NameAttributeFromMailID.java :

```
private static final String LDAP_USER_MAIL = "LDAP User email";  
  
private static final int EV_LDAP_USER_MAIL = 71;
```

This parameter has to be mapped to email ID of the user present in the user store.



- Distribute the policy extension.  
For steps, refer "[Distributing Policy Extension](#)" in the [NetIQ Access Manager 3.2 Policy Guide](#). Create an External Attribute policy from the data extension, assign the policy to the Identity Server, and distribute the policy extension created in step 2.



#### Rules defined for this policy

#### Fetch Attributes Action:

- Name of the External Attribute: ExternalAttrSource\_Email\_Policy Result
- Value of the External Attribute: This points to the ExternalAttrSource\_EmailExample, which is a policy data extension created in step 2.

The screenshot shows the NetIQ Access Manager Admin interface. The top navigation bar includes 'Access Manager', 'Devices', 'Policies', and 'Auditing'. The current view is 'Policies > Edit Policy > Edit Rule: ExternalAttrSource\_Email\_Policy - Rule 1'. The configuration details are as follows:

- Type: Identity Server: External Attribute Source
- Description: This is a simple external attribute source policy that returns the name or id present in .
- Priority: 1

The 'Actions' section shows a 'New' dropdown menu with the following configuration:

- Do: Fetch Attributes
- External Attribute Name: ExternalAttrSource\_Email\_Policy Result
- Value: Data Extension:ExternalAttrSource\_EmailExample

At the bottom, there is a note: 'Changes made on this panel must be applied from the Policies Panel.' and two buttons: 'OK' and 'Cancel'.

CustomParameter: The name of this External Attribute (that is ExternalAttrSource\_Email\_Policy Result) can be obtained in the extension class (ExternalAttrSource\_EmailExample) by using com.novell.nxpe.CustomParameter as shown in the following code snippet:

```
NxpeParameter customParam = new
CustomParameter(CustomParameter.ATTRIBUTE_VALUE);
System.out.println("FADE: Attribute that we are looking at is " +
informationContext.getData(customParam));
```

Thus, if the same extension class is used in more than one External Attribute Source policies, External Attribute Name can be kept different in different policies. Then, *CustomParameter* can be used in the policy extension class to identify the External Attribute Source policy that is invoking the policy extension class, based on the External Attribute Name.

4. Define a shared secret for the Name field of the email ID.

For more information, see [Creating Shared Secret Names](#) in the [NetIQ Access Manager 3.2 Identity Server Guide](#).

Shared secret's settings:

- a) *Secret Name* should match the policy name (created in step 3- ExternalAttrSource\_Email\_Policy)

- b) *Secret Entry Name* should match the attribute name configured while creating the policy (see step 3, External attribute Name -ExternalAttrSource\_Email\_Policy Result)
5. After the shared secret is created, it can be used for configuring other policies or for being set in the attribute sets of the Identity Servers.
- a) Attribute sets: The shared secret is used to create attribute sets that the Identity Server sends in the authentication response to the service providers. For more information, see [Configuring the Attributes Sent with Authentication](#) in the [NetIQ Access Manager 3.2 Identity Server Guide](#).
  - b) Use in other policies: The shared secret can be used in other policies. For example, an identity injection policy can be configured to obtain the value of a shared secret and inject it into the custom headers. For more information, see [Creating Identity Injection Policies](#) and [Configuring a Custom Header Policy](#) in the [NetIQ Access Manager 3.2 Policy Guide](#).

Copyright © 2012 Novell, Inc. All Rights Reserved.

Novell grants permission, free of charge, to any person obtaining copies of this software and its associated documentation files (the "Software"), to deal in the Software without restriction, including to use, copy, adapt, publish, distribute, display, perform, sublicense, and sell copies of the Software, subject to the following condition: You must include the above copyright notice and this permission notice in all full or partial copies of the Software.

NOVELL PROVIDES THE SOFTWARE "AS IS," WITHOUT ANY EXPRESS OR IMPLIED WARRANTY, INCLUDING WITHOUT THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, AND NON-INFRINGEMENT. NOVELL, THE AUTHORS OF THE SOFTWARE, AND THE OWNERS OF COPYRIGHT IN THE SOFTWARE ARE NOT LIABLE FOR ANY CLAIM, DAMAGES, OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT, OR OTHERWISE, ARISING FROM, OUT OF, OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.