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 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 NxpeInformationContext 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.

- 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.

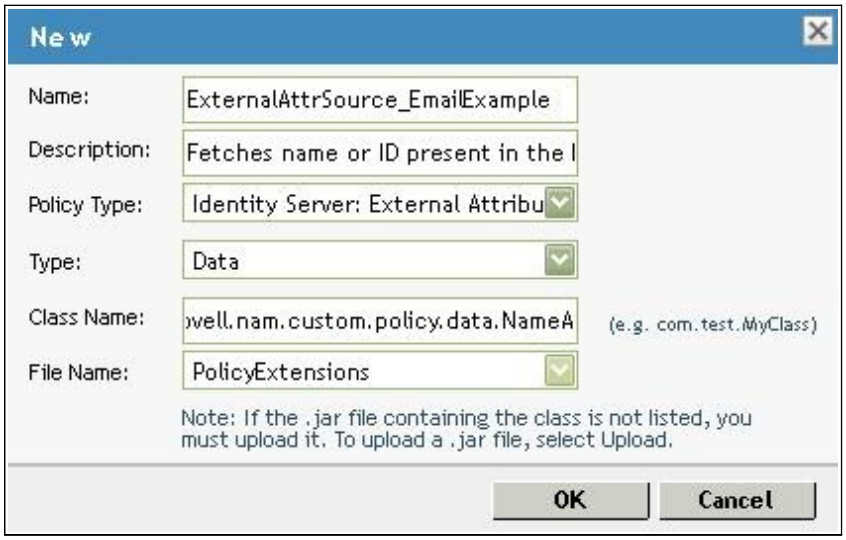
2.1. Creating a new policy extension:

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



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

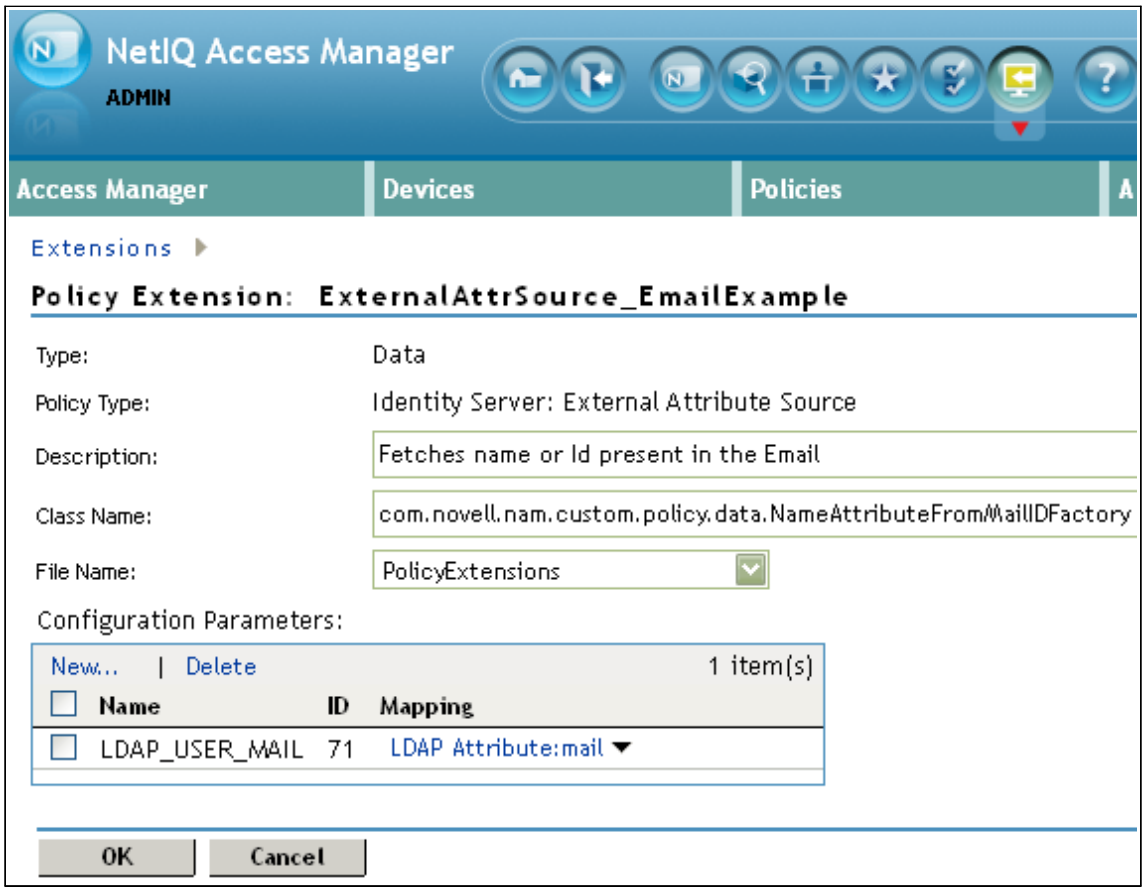
a) Name: LDAP_USER_MAIL (Name can be anything)

b) ID: 71 (This must match the parameter ID mentioned in the policy extension implementation in code)

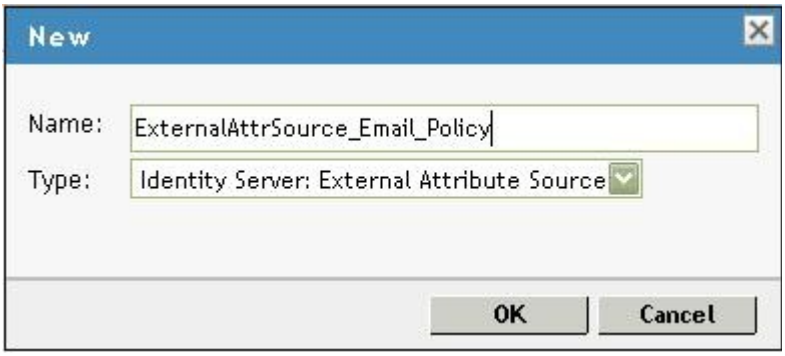
c) Corresponding extract from NameAttributeFromMailID.java :
private static final String LDAP_USER_MAIL = "LDAP User email";

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.



3. 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.

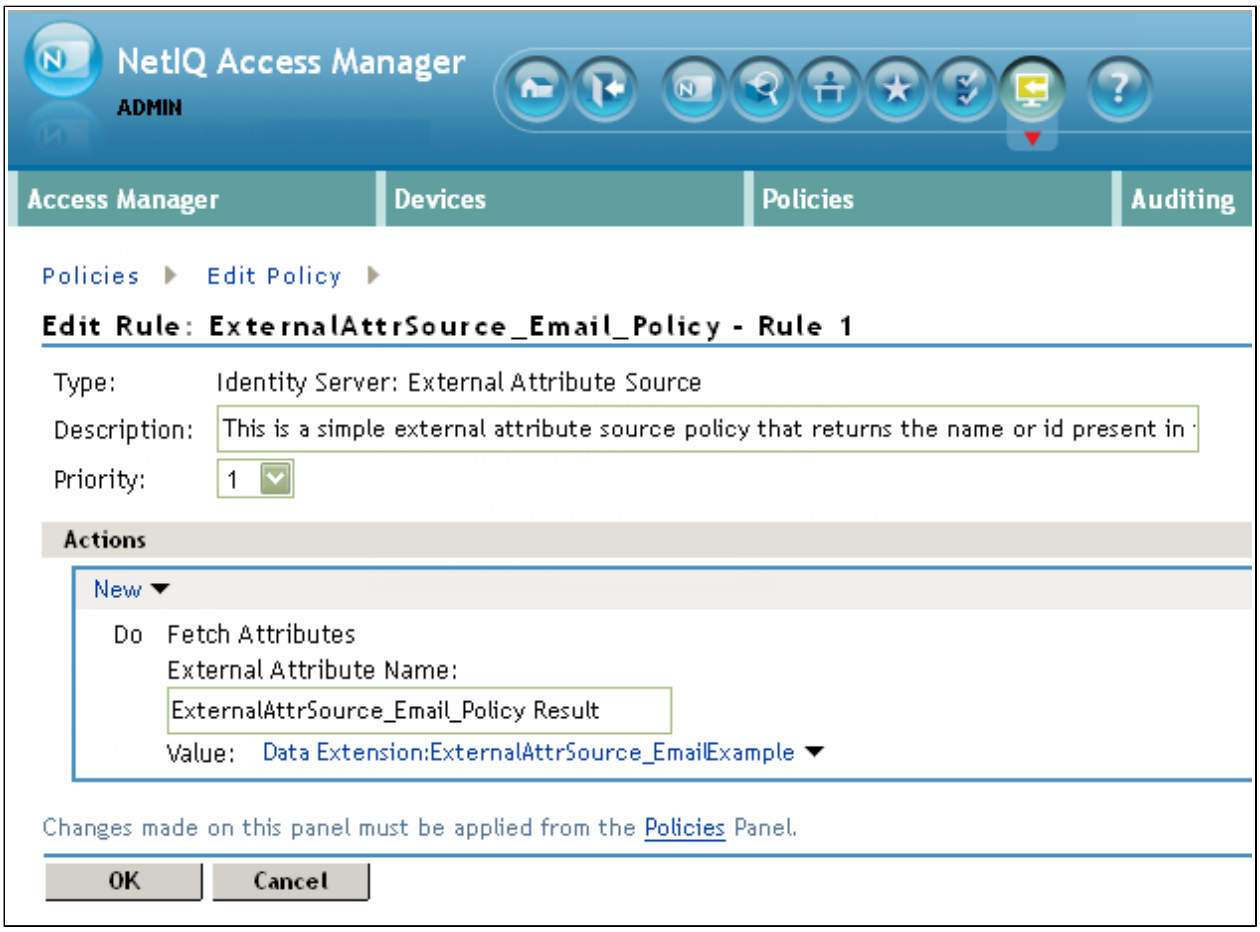


Rules defined for this policy

Fetch Attributes Action:

a) Name of the External Attribute: ExternalAttrSource_Email_Policy Result

b) Value of the External Attribute: This points to the ExternalAttrSource_EmailExample, which is a policy data extension created in step 2.



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 identity 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.