

NetIQ Access Manager Template Data Policy Readme

Policy extension examples include:

- PolicyDataExtnFactoryTemplate.java
- PolicyDataExtnTemplate.java

Purpose

This example can be used as a template to implement a policy extension of type *Data* that is `com.novell.nxpe.NxpeContextDataElement`. This example provides a basic framework that can be used as a starting point for creating data policy (`com.novell.nxpe.NxpeContextDataElement`.) extensions.

This example covers the following topics:

- How to configure and install the *Data* policy extension in the Administration Console.
- Implementation details of the Data policy extension factory and extension classes.

Functionality

This policy extension contains a simple logic that returns "A" if LDAP UserDN of the user is not empty or null otherwise it returns "B". In this example, this *Data* policy extension is used in an Identity Injection policy to inject the output of the policy extension in the custom header.

Implementation Details

- This *Data Policy Extension* example consists of the following two Java classes:

PolicyDataExtnFactoryTemplate.java: Implements the `com.novell.nxpe.NxpeContextDataElementFactory` interface. This is a factory that creates `PolicyDataExtnTemplate` objects.

PolicyDataExtnTemplate.java: Implements the `com.novell.nxpe.NxpeContextDataElement` interface. It contains the methods required to create a context data element that can be used for injection, for activating roles, or in a condition.

- This class needs a configuration parameter *LDAP User DN* 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 to it 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 `getLDAPUserDN` method in `PolicyConditionExtnTemplate` is used to retrieve the value of the LDAP User DN configuration parameter from the `NxpeInformationContext` object
- The `NxpeException` class is used for exceptions in `PolicyDataExtnTemplate`.

- The *Data* policy extension can be used in the Access Gateway Authorization policies, Access Gateway Identity Injection policies, Identity Server Role policies, and Identity Server External Attribute Source policies.

How to Customize This Example to Meet Your Needs?

- Factory: Modify the PolicyDataExtnFactoryTemplate class to control the creation of PolicyDataExtnTemplate as needed. Synchronize the methods of PolicyConditionExtnTemplate accordingly to avoid problems due to concurrency.
- Configuration parameters: Identify and set the configuration parameters required by your condition extension and provide their mappings in the Administration Console.
- Implement getter methods for all of your configuration parameters in the PolicyDataExtnTemplate class, as it has been done for retrieving LDAP User DN in this example.
- Modify getValue method to contain your own condition evaluation logic in PolicyDataExtnTemplate.
- For creating the *Data* policy extension for other policies, create a policy extension from your customized extension classes of the type of that policy (Authorization, Role or External Attribute Source respectively) and then create a policy of the same type (Authorization, Role or External Attribute Source respectively) that uses your policy extension. For more information, see [Section 2.2, Creating Roles](#), [Section 4.2, Configuring an Identity Injection Policy](#), [Section 3.2, Creating Access Gateway Authorization Policies](#), and [Section 6.0, Creating External Attribute Policies](#), [Section 1.7, Adding Policy Extensions](#) in the NetIQ Access Manager 3.2 Policy Guide.

For more details on the implementation, see comments in the PolicyDataExtnFactoryTemplate and PolicyDataExtnTemplate classes or [NetIQ Access Manager 3.2 Developer Kit](#).

Installation

Before you start the installation, you need the following:

- The following Java classes and nxpe.jar:

PolicyDataExtnTemplate.java
PolicyDataExtnFactoryTemplate.java

Download novell-nacm3_2.tar.gz (SDK tar) from http://www.novell.com/developer/ndk/novell_access_manager_developer_tools_and_examples.html.

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

- Java SDK (jdk1.6) to compile the Java sources.
- Apache Ant. You can download it from <http://ant.apache.org/bindownload.cgi>.
- An Access Manager setup with the Identity Server, Access Gateway, and policies.
- 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. 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 > TemplateDataExtension_Example and run 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. For example, run the following commands in the Linux environment:

- export ANT_HOME=/usr/apache-ant-1.8.1<set it as per your environment>
 - export JAVA_HOME=/usr/java/jdk1.6.0_27<set it as per your environment>
 - export PATH=\$ANT_HOME/bin:\$JAVA_HOME/bin:\$PATH
 - ant all
2. Create a policy extension of the type *Access Gateway: Identity Injection*. For more information, see [Installing the Extension on the Administration Console](#) in the [NetIQ Access Manager 3.2 Policy Guide](#).
 - 2.1. Creating a new policy extension:
 - Name: TemplatePolicyDataExtension
 - Policy Type: Access Gateway: Identity Injection
 - Class Name: com.novell.nam.custom.policy.data. PolicyDataExtnFactoryTemplate
 - File Name: name of the jar file created in step 1

New

Name:

Description:

Policy Type:

Type:

Class Name: (e.g. com.test.MyClass)

File Name:

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.

- Name: LDAP_User_DN (Name can be anything)
- ID: 41 (This must match the parameter ID as mentioned in the policy extension implementation in the code)
- Corresponding extract from PolicyDataExtnTemplate.java:

```
private static final String LDAP_USER_DN_NAME = "LDAP User DN";
```

```
private static final int EV_LDAP_USER_DN = 41;
```

This parameter has to be mapped to LDAP User DN of the user present in the credential profile as shown in the following image:

Policy Extension: TemplatePolicyDataExtension

Type: Data

Policy Type: Access Gateway: Identity Injection

Description:

Class Name:

File Name:

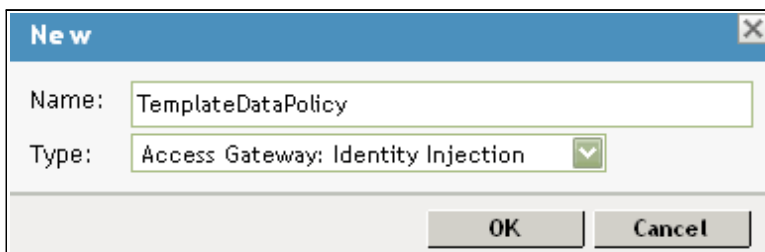
Configuration Parameters:

<input type="checkbox"/> Name	ID	Mapping
<input type="checkbox"/> LDAP_User_DN	41	Credential Profile:LDAP Credentials:LDAP User DN ▼

3. Distribute the policy extension.

For more information, see [“Distributing Policy Extension”](#). Create an identity injection policy from the data extension, assign the policy to the Access Gateway and distribute the policy extension created in step 2.

The “TemplateDataPolicy” Identity injection policy created in this step will be assigned to a protected resource. For more information, see [Assigning an Identity Injection Policy to a Protected Resource](#). It can be configured to inject the output of TemplatePolicyDataExtension in the HTTP header of the request sent to the protected resource.



New

Name:


Type:

OK Cancel

Rules defined for this policy

“Inject into Custom Header” Action:

- Custom Header Name: TemplatePolicyDataExtension Result (Name can be anything)
- Value of the Custom Header: This points to the TemplatePolicyDataExtension, which is a policy data extension created in step 3.



Policies ▶ Edit Policy ▶

Edit Rule: TemplateDataPolicy - Rule 1

Type: Access Gateway: Identity Injection

Description:

Priority:

Actions

New ▼

Do Inject into Custom Header

Custom Header Name:

Value:

Multi-Value Separator:

DN Format:

Changes made on this panel must be applied from the [Policies](#) Panel.

OK Cancel

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.