

Novell Access Manager LDAP Group Data Element Readme

Policy extension examples include:

- LDAPGroupDataElement.java
- LDAPGroupDataElementFactory.java

Purpose

This example illustrates how a policy extension can use external data sources to obtain the information.

Functionality

This policy extension connects to the required LDAP repository, runs a search on it, and returns the results. An Identity Injection policy is created in this example that uses this policy extension.

Implementation Details

- The policy extension used in this example is of type *Data* and contains the following classes:
 - LDAPGroupDataElementFactory.java:** Implements the `com.novell.nxpe.NxpeContextDataElementFactory` interface. This is a factory that creates `LDAPGroupDataElement` objects.
 - LDAPGroupDataElement.java:** Implements the `com.novell.nxpe.NxpeContextDataElement` interface.
- This class needs the following configuration parameters:
 - Parameters required for LDAP connections (Provider_URL, Security principal, Security credentials)
 - Parameters required for LDAP search (Search context and Search filter (LDAP_User_DN))
 - Parameter for enabling Debug statements (debug)
- 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 `NxpeInformationContext` object is used to retrieve the value of configuration parameter in `LDAPGroupDataElement` as in the `getSecurityPrincipal`, `getLDAPUserDN`, `getSearchContext` methods.
- For a secure LDAP connection, you need to import the certificate of the external LDAP repository in a keystore, which is present at the Access Manager component that evaluates the policy. For this example, certificate can be imported in the “cacert” keystore present under the following location in the Access Gateway (because Identity Injection policies are evaluated at Access Gateway).

Location: /opt/novell/jdk1.6.0_30/jre/lib/security

If you want to use any other keystore, then point that keystore in the LDAP connection code for authentication.

For further information on connecting to external data sources, see [Retrieving Information from the Identity Server User Store](#) in the [NetIQ Access Manager 3.2 Developer Kit](#).

- For more details on the basics of policy extension and its implementation, see [NetIQ Access Manager 3.2 Developer Kit](#).

Installation

Before you start the installation, you need the following:

- The following Java classes and nxpe.jar:

LDAPGroupDataElement.java
LDAPGroupDataElementFactory.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.
- Knowledge of Java programming.

Setup

1. Extract the SDK tar (novell-nacm3_2.tar.gz). Go to nacm-3_2_0 > samples > PolicyExtension > LDAPGroupDataElement_Example and run build.xml.

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

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.

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 steps, 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: PolicyDataExtLDAP
- Policy Type: Access Gateway: Identity Injection
- Class Name: com.novell.nam.custom.policy.data.LDAPGroupDataElementFactory
- 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_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 LDAPGroupDataElement.java:

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

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

This parameter is mapped to LDAP User DN of the user present in the credential profile. This parameter is used to build a filter for LDAP search in the code.

- b) Name: Security_Principal

ID: 51 (This must match the parameter ID as mentioned in the policy extension implementation in the code)

Corresponding extract from LDAPGroupDataElement.java:

```
private static final String SECURITY_PRINCIPAL_NAME = "Security Principal"

private static final int EV_SECURITY_PRINCIPAL = 51;
```

This parameter is mapped to the user's LDAP User Name present in the credential profile. This parameter is used in the code for setting the "SECURITY_PRINCIPAL" attribute (identity of the user) required for establishing the LDAP context.

c) Name: Security_Credentials

ID: 52 (This must match the parameter ID as mentioned in the policy extension implementation in the code)

Corresponding extract from LDAPGroupDataElement.java:

```
private static final String SECURITY_CREDENTIALS_NAME = "Security Credentials";

private static final int EV_SECURITY_CREDENTIALS = 52;
```

This parameter is mapped to the user's LDAP password present in the credential profile. This will be used in the code for setting the "SECURITY_CREDENTIALS" attribute for establishing the LDAP context.

d) Name: Debug

ID: 91 (This must match the parameter ID as mentioned in the policy extension implementation in the code)

Corresponding extract from LDAPGroupDataElement.java:

```
private static final String DEBUG_NAME = "Debug";

This parameter is set to "True" to enable debug statements.
```

e) Name: Search_Context

ID: 61 (This must match the parameter ID as mentioned in the policy extension implementation in the code)

Corresponding extract from LDAPGroupDataElement.java:

```
private static final String SEARCH_CONTEXT_NAME = "Search Context";

private static final int EV_SEARCH_CONTEXT = 61;
```

This parameter is used to set the Search context (name of the context or object to search) in the LDAP search.

f) Name: Provider_URL

ID: 31 (This must match the parameter ID as mentioned in the policy extension implementation in the code)

Corresponding extract from LDAPGroupDataElement.java:

```
private static final String PROVIDER_URL_NAME = "User Store Replica";
private static final int EV_PROVIDER_URL = 31;
```

This parameter is used to set the Provider URL for establishing the LDAP context.

The screenshot shows the NetIQ Access Manager console with the 'Policies' tab selected. The 'Policy Extension: PolicyDataExtLDAP' configuration window is open. The 'Type' is 'Data' and the 'Policy Type' is 'Access Gateway: Identity Injection'. The 'Description' and 'Class Name' fields both contain 'com.novell.nam.custom.policy.data.LDAPGroupDataElementFactory'. The 'File Name' is set to 'ExtensionPolicyJar'. Below these fields is a 'Configuration Parameters' table with 6 items.

<input type="checkbox"/> Name	ID	Mapping	
<input type="checkbox"/> LDAP_User_DN	41	Credential Profile:LDAP Credentials:LDAP User DN ▼	
<input type="checkbox"/> Security_Principal	51	Credential Profile:LDAP Credentials:LDAP User Name ▼	
<input type="checkbox"/> Security_credential	52	Credential Profile:LDAP Credentials:LDAP Password ▼	
<input type="checkbox"/> Debug	91	String Constant ▼	: True
<input type="checkbox"/> Search_Context	61	String Constant ▼	: o=novell
<input type="checkbox"/> Provider_URL	31	String Constant ▼	: ldaps://164.99.86.54

At the bottom of the window are 'OK' and 'Cancel' buttons.

3. Distribute the policy extension.

For steps, 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 "PolicyDataLDAP" 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 will inject the output of "PolicyDataExtLDAP" in the HTTP header of the request sent to the protected resource.

New "Access Gateway : Identity Injection " policy

The 'New' dialog box has a title bar with a close button. It contains two input fields: 'Name' with the value 'PolicyDataLDAP' and 'Type' with a dropdown menu showing 'Access Gateway: Identity Injection'. At the bottom right are 'OK' and 'Cancel' buttons.

Rules defined for this policy

“Inject into Custom Header” Action:

- Custom Header Name: LDAP Search Result (Name can be anything)
- Value of the Custom Header: This points to the PolicyDataExtLDAP, which is a policy data extension created in step 2.

The configuration panel shows the breadcrumb 'Policies > Edit Policy'. The title is 'Edit Rule: PolicyDataLDAP - Rule 1'. It includes fields for 'Type' (Access Gateway: identity Injection), 'Description' (An example policy that uses policy extension to obtain information from external LDAP r), and 'Priority' (1). Below is an 'Actions' section with a 'New' dropdown. The action is 'Inject into Custom Header' with configuration: 'Custom Header Name' (LDAP Search Result), 'Value' (Data Extension:PolicyDataExtLDAP), 'Multi-Value Separator' (,), and 'DN Format' (LDAP (ex, cn=jsmith,ou=Sales,o=Novell)). A note at the bottom states: 'Changes made on this panel must be applied from the Policies panel.' At the bottom are 'OK' and 'Cancel' buttons.

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.