

NetIQ Access Manager Template Action Policy (Permit) Readme

The Policy extension examples include:

- PolicyActionExtnPermitFactoryTemplate.java
- PolicyActionExtnPermitTemplate.java

Purpose

This example can be used as a template to implement an action policy extension of type Permit. This example provides a basic framework that can be used as a starting point for creating action policy (com.novell.nxpe.NxpeContextActionElement.) permit extensions.

This example covers the following topics:

- How to configure and install the Action policy extension of type "Permit" in the Administration Console
- Implementation details of the Action policy extension factory and extension classes

Functionality

This policy action extension can be of type "Deny", "Permit" or "Obligation". An action defined as "Permit" extension in UI will Permit access to the protected resource in both case of success and failure.

Implementation Details

- This Action Policy Extension of type Permit example consists of the following two Java classes:
 - **PolicyActionExtnPermitFactoryTemplate.java:** Implements the com.novell.nxpe.NxpeActionFactory interface. This is a factory that creates PolicyActionExtnTemplate objects
 - **PolicyActionExtnPermitTemplate.java:** Implements the com.novell.nxpe.NxpeActionFactory interface. It contains the methods required to create a context action element that can be used for authorization, 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 doAction method in the NxpeAction 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 PolicyActionExtPermitTemplate 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 PolicyActionExtPermitTemplate

- Permit Action Extension will return either Success (NxpeResult.Success) or Failure (NxpeResult.GeneralFailure, NxpeResult.ErrorDataUnavailable etc.). In case of success any other action as per the extension logic (eg. updating the external database) will be performed and simultaneously logged. In case of failure the same would be skipped but the access to the protected resource will be granted in both the cases
- The Action policy extension can be used in the Access Gateway Authorization policies

How to Customize This Example to Meet Your Needs?

- Factory: Modify the PolicyActionExtPermitFactoryTemplate class to control the creation of PolicyActionExtPermitTemplate as needed. Synchronize the methods of PolicyActionExtPermitTemplate accordingly to avoid problems due to concurrency
- Configuration parameters: Identify and set the configuration parameters required by your action extension and provide their mappings in the Administration Console
- Implement getter methods for all of your configuration parameters in the PolicyActionExtPermitTemplate class, as it has been done for retrieving LDAP User DN in this example
- Modify doAction method to contain your own policy evaluation logic in PolicyActionExtPermitTemplate
- For creating the Action policy extension for other policies, create a policy extension from your customized extension classes of the type of Authorization policy and then create a policy of the same type that uses your policy extension. For more information, see Section 6.2.3, Creating Roles, Section 6.4.2, Configuring an Identity Injection Policy, Section 6.3.2, Creating Access Gateway Authorization Policies, and Section 6.6, Creating External Attribute Policies, Section 6.1.6, Adding Policy Extensions in the [NetIQ Access Manager 4.2 Administration Guide](#).
- For more details about the implementation, see comments in the PolicyActionExtPermitFactoryTemplate and PolicyActionExtPermitTemplate classes or [NetIQ Access Manager 4.2 Developer Guide](#).

Installation

Before starting installation, ensure that the following prerequisites are met:

The following Java classes and jar files nxpe.jar & nidp.jar:

- PolicyActionExtPermitFactoryTemplate.java
- PolicyActionExtPermitTemplate.java

Download netiq-access-manager-sdk-4_2-extensions.tar.gz (SDK tar) from

https://www.netiq.com/documentation/access-manager-developer-documentation/resources/netiq-access-manager-sdk-4_2-extensions.tar.gz.

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

- 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

Extract the SDK tar (netiq-access-manager-sdk-4_2-extensions.tar.gz). Go to `nam_4_2 > samples > PolicyExtension > TemplateActionPermitExtension_Example`. 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, modify its java source, compile, and generate a Jar file.

Steps:

1. Go to `nam_4_2 > samples > PolicyExtension > TemplateActionPermitExtension_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 `nam_4_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: Authorization. For steps, see [Installing the Extension on the Administration Console](#) in the [NetIQ Access Manager 4.2 Administration Guide](#).

2.1. Create a new policy extension:

- Name: TemplatePolicyActionExtension
- Type: Access Gateway: Authorization
- ClassName: com.novell.nam.custom.policy.action.PolicyActionExtPermitFactoryTemplate
- File name: name of the jar file created in step 1

NOTE: There should be no empty spaces anywhere in the Class Name.

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:

Dashboard > Devices > Policies > Security > Extensions

Policy Extension: TemplatePolicyActionExtension

Type: Action: Permit
 Policy Type: Access Gateway: Authorization
 Description: Allow Action Extension
 Class Name: com.novell.nam.custom.policy.action.PolicyActionExtAllowFactoryTemplate
 File Name: PolicyExtensions

Configuration Parameters:

New... Delete			1 item(s)
<input type="checkbox"/> Name	ID	Mapping	
<input type="checkbox"/> LDAP_User_DN	41	Authentication Contract:[Current]	▼

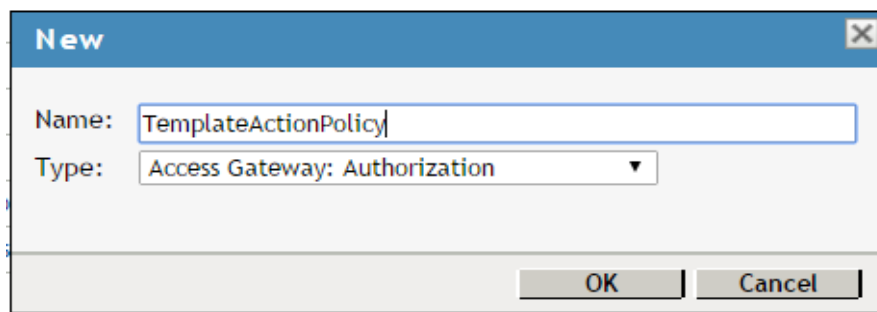
OK Cancel

3. Distribute the policy extension.

For the steps, see “[Distributing Policy Extension](#)”. Create an authorization policy from the data extension, assign the policy to the Access Gateway, and distribute the policy extension created in step 3.

The TemplateActionPolicy authorization policy created in this step will be assigned to a protected resource. For more information, see [Assigning an Authorization Policy to a Protected Resource](#). This policy will evaluate the TemplatePolicyActionExtension – Policy “Action” Extension created in step 2 and will decide whether to provide access to the protected resource. Since the action extension created is of type “Permit”, it would Permit the access to the protected resource.

New Access Gateway: Authorization policy



The image shows a 'New' dialog box with a blue header and a close button (X) in the top right corner. Below the header, there are two input fields: 'Name:' with the text 'TemplateActionPolicy' and 'Type:' with a dropdown menu showing 'Access Gateway: Authorization'. At the bottom of the dialog, there are two buttons: 'OK' and 'Cancel'.

Rules defined in this policy:

- Condition Group: Authentication Contract
- Action: Permit: ActionExtension(Permit) : TemplatePolicyActionExtension

This points to the TemplatePolicyActionExtension, which is a policy action extension created in step 2. Since the action defined as "Permit" extension in UI, it will always permit access. The protected resource can be accessed in case of both success and failure. Any action present in the extension logic will take place in case if it returns success while in case of failure no action would be performed. If any error is returned, then it will be logged in the logs.

Edit Rule: TemplateActionPolicy - Rule 1

Type: Access Gateway: Authorization

Description:

Priority:

Conditions

Condition Group 1

New ▾

Authentication Contract: ⓘ

Comparison:

Mode:

Value:

Result on Condition Error:

Actions

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

For information about NetIQ legal notices, disclaimers, warranties, export and other use restrictions, U.S. Government restricted rights, patent policy, and FIPS compliance, see <https://www.netiq.com/company/legal/>.

© 2016 NetIQ Corporation. All Rights Reserved.

For information about NetIQ trademarks, see <https://www.netiq.com/company/legal/>. All third-party trademarks are the property of their respective owners.