API Gateway

Version 7.6.2

14 July 2020

Authentication and Authorization Integration Guide

This documentation describes the following Axway software:

Axway API Gateway 7.6.2

Contents

Preface

API Gateway contains a set of message filters that directly or indirectly restrict access to resources or web services.

Filters that directly control access include XML-signature verification, CA certificate chain verification, and SAML assertion verification. With these filters, policy decisions are made and enforced within API Gateway itself.

Filters that *indirectly* control access offload the policy decision to an external access management system. With these filters, the policy decision is made by the external system but then enforced by API Gateway.

API Gateway can leverage your existing Identity Management infrastructure, thus avoiding the need to maintain separate silos of user information. For example, if you already have a database full of user credentials, API Gateway can authenticate requests against this database rather than using its own internal user store. Similarly, the API Gateway can authorize users, look up user attributes, and validate certificates against third-party Identity Management servers.

This guide describes how to configure API Gateway to integrate with the following products:

- LDAP servers — see *LDAP identity manager integration* on page 10.
- CA SiteMinder — see *CA SiteMinder integration* on page 23.
- RSA Access Manager — see *RSA Access Manager integration* on page 35
- Oracle Access Manager integration - see *Oracle Access Manager 11gR2 integration* on page 40
- Oracle Entitlements Server integration - see *Oracle Entitlements Server 11g and 11gR2 integration* on page 57

Who should read this guide

The intended audience for this guide is personnel in charge of the technical integration of an API Gateway solution.

Related documentation

The AMPLIFY API Management solution enables you to create, publish, promote, and manage Application Programming Interfaces (APIs) in a secure and scalable environment. For more information, see the *AMPLIFY API Management Getting Started Guide*.

The following reference documents are also available on the Axway Documentation portal at https://docs.axway.com:

- *Supported Platforms*

  Lists the different operating systems, databases, browsers, and thick client platforms supported by each Axway product.

- *Interoperability Matrix*

  Provides product version and interoperability information for Axway products.

Support services

The Axway Global Support team provides worldwide 24 x 7 support for customers with active support agreements.

Email support@axway.com or visit Axway Support at https://support.axway.com.

See "Get help with API Gateway" in the *API Gateway Administrator Guide* for the information that you should be prepared to provide when you contact Axway Support.

Training services

Axway offers training across the globe, including on-site instructor-led classes and self-paced online learning. For details, go to: http://www.axway.com/support-services/training

Accessibility

Axway strives to create accessible products and documentation for users.

This documentation provides the following accessibility features:

Screen reader support

- Alternative text is provided for images whenever necessary.
- The PDF documents are tagged to provide a logical reading order.

Support for high contrast and accessible use of colors

- The documentation can be used in high-contrast mode.
- There is sufficient contrast between the text and the background color.
- The graphics have the right level of contrast and take into account the way color-blind people perceive colors.

Updates and revisions

This guide includes the following documentation changes.

Changes in version 7.6.2

- Added sections on Oracle Access Manager (OAM) integration and Oracle Entitlements Server (OES) integration

Changes in version 7.6.1

No changes.

Changes in version 7.6.0

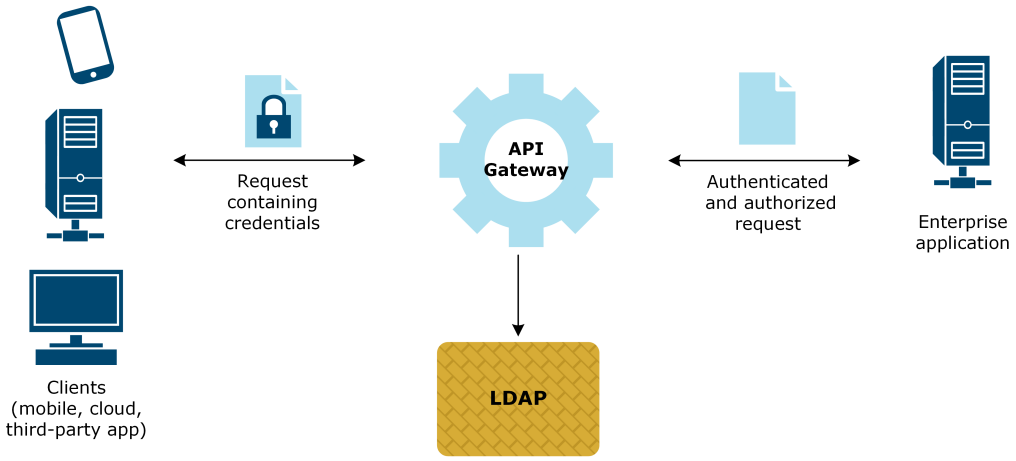
- Removed section on IBM Tivoli Access Manager integration because API Gateway 7.6.2 does not support Windows.

LDAP identity manager integration 1

API Gateway interacts with the following directory servers using the Lightweight Directory Access Protocol (LDAP):

- Apache Directory Server 2.0.0-M7
- IBM Security Directory Server 6.4.0
- Microsoft Active Directory 2012
- Open LDAP Directory Server 2.4.11
- Oracle Directory Server Enterprise Edition 11g
- Oracle Internet Directory 10.1.4.0.1
- Oracle Virtual Directory 11g

Flow description



The integration flow is as follows:

- API Gateway authenticates requests by searching the LDAP server for the user. API Gateway sends a bind request to the LDAP server to authenticate the user's credentials. For example, API Gateway extracts the user name and password from an HTTP Basic request and binds to LDAP using these credentials to check if the user name and password are valid.
- API Gateway retrieves roles or attributes from LDAP by searching the LDAP server for entries and storing retrieved values on the whiteboard to be used by other filters.

When a request has been authenticated, API Gateway can insert a SAML token into the message to show that authentication has occurred. This SAML token can then be consumed by downstream applications to extract information about the original client that sent the request.

Prerequisites

Before you start, you must have API Gateway and your chosen directory server installed and configured.

Configuration process

The configuration process has the following steps:

1. *Check the details for the directory server* on page 11
    - *Check the connection details* on page 11
    - *Check the user search conditions* on page 12
2. *Configure an LDAP connection* on page 14
3. *Configure an LDAP authentication repository* on page 14
4. *Configure API Gateway policy* on page 15
    - *Create an authentication policy* on page 16
    - *Test the policy* on page 16
    - *Retrieve attributes from the directory server* on page 17
    - *Validate a retrieved attribute* on page 18
    - *Insert a SAML token* on page 19
5. *Secure the connection to the directory server* on page 20

Check the details for the directory server

Before you can configure API Gateway to connect to your directory server, you must have the connection details and the user search conditions for the directory server.

- *Check the connection details* on page 11
- *Check the user search conditions* on page 12

Check the connection details

Before you can start with the configuration, you must have the following connection details:

Setting	Examples	Description
LDAP URL	• `ldap://192.168.0.129:10389` • `ldaps://192.168.0.129:10636`	The LDAP URL containing the host name and port that your directory server is listening on.
User name	• `uid=admin,ou=system` • `cn=root` • `CN=Administrator,CN=users,DC=axway,DC=com` • `cn=admin,o=Axway,I=Dublin4,st=Dublin,C=IE`	The distinguished name (DN) of the user that API Gateway uses when connecting to the directory server. The format may vary depending on your directory server.
Password	`secret`	The password of the user API Gateway uses.

Ensure you have these details at hand when you start configuring the connection between API Gateway and the directory server.

Check the user search conditions

API Gateway searches the directory server based on the details you define when configuring the LDAP authentication repository for API Gateway.

1. Connect and log in to the directory server using an LDAP browser.
2. Decide how you want to search the repository and note down the following details:

Setting	Examples	Description
Base Criteria	• `ou=system` • `CN=LOCALHOST` • `CN=users,DC=axwayqa,DC=com` • `ou=R&D,o=Axway,I=Dublin4,st=Dublin,C=IE`	The root DN to use when running queries against the directory server.
User Class	• `inetOrgPerson` • `User` • `Person`	The object class searched in the directory server. Each object in an LDAP directory has at least one object class associated with it.
User Search Attribute	• `uid` • `cn`	The attribute that contains the user name.

The format of the setting values may vary depending on your directory server.

When searching the directory server, API Gateway generates a search based on the User Class and User Search Attribute values:

```
(&(objectclass=<User Class>)(<User Search Attribute>=<value>))
```

For example:

```
(&(objectclass=inetOrgPerson)(uid=admin))
```

This example searches the repository as follows:

- Search for an object of type `inetOrgPerson` where the attribute `uid` has the value `admin`. Start from under the value entered for Base Criteria.
- If the user is found, return the DN.

API Gateway binds to the directory server using the returned DN and the password extracted from the request. A successful bind indicates that the user name and password are valid and the user has been authenticated.

Configure an LDAP connection

API Gateway binds to the directory server using the connection details and user credentials specified in the LDAP connection. Usually, the connection details include the user name and password of an administrator user who has read access to all users in the LDAP server you want to auhtenticate or retrieve attributes for.

This section describes the steps required to configure the connection between API Gateway and your directory server in Policy Studio. For more information on working in Policy Studio, see the *API Gateway Policy Developer Guide*.

1. In the node tree, click **Environment Configuration > External Connections**.
2. Right-click **LDAP Connections**, and click **Add a LDAP Connection**.
3. Enter a name for your connection (for example, `LDAP Connection`), and set the following values:
   - **URL**: `<LDAP url>`
   - **Type**: `Simple`
   - **User Name**: `<user name for API Gateway>`
   - **Password**: `<password for API Gateway>`

   For more information on configuring LDAP connections, see "Configure LDAP directories" in the *API Gateway Policy Developer Guide*.
4. Click **Test Connection** to verify that the connection to the directory server is configured successfully.
5. Click **OK** to save the entry in **LDAP Connections**.

Configure an LDAP authentication repository

API Gateway requires an authentication repository to authenticate a user using the user name and password. API Gateway compares the credentials user presents to those stored in the authentication repository. If API Gateway can retrieve a user's profile and bind to the directory server as that user, the user is authenticated.

You can leverage your existing directory server and configure API Gateway to query it for user profile data. This section describes how to configure integration between API Gateway and your directory server in Policy Studio. For more information on working in Policy Studio, see the *API Gateway Policy Developer Guide*.

1. In the node tree, click **Environment Configuration > External Connections > Authentication Repositories**.
2. Right-click **LDAP Repositories**, and click **Add a new Repository**.
3. Enter a name for your repository (for example, `LDAP Repository`), and set the following values:
   - **LDAP Directory**: The LDAP connection you created (`LDAP Connection`).
   - **Base Criteria**: `<Base Criteria of your directory server>`.
   - **User Class**: `<User Class of your directory server>`.
   - **User Search Attribute**: `<User Search Attribute of your directory server>`.
   - **Login Authentication Attribute**: This setting is optional. If left blank, API Gateway uses a default name as the authentication attribute. If not blank, the specified attribute is retrieved in the initial search for the user.
   - **Authorization Attribute**: Enter the attribute stored LDAP for the user you want to use for authorization.
   - **Authorization Attribute Format**: Depending on the format of your selected authorization attribute, select either `User Name` or `X.509 Distinguished Name`.

   For more information on configuring an LDAP repository, see "LDAP repositories" in the *API Gateway Policy Developer Guide*.
4. Click **OK** to save the configuration.

Configure API Gateway policy

This section describes the steps required to configure integration between API Gateway and your directory server in Policy Studio. For more information on working in Policy Studio, see the *API Gateway Policy Developer Guide*.

Create an authentication policy

You must configure an authentication policy to set API Gateway to authenticate against your directory server. This example uses the **HTTP Basic** authentication filter with a user name and password combination, but you can configure a different authentication mechanism as required.

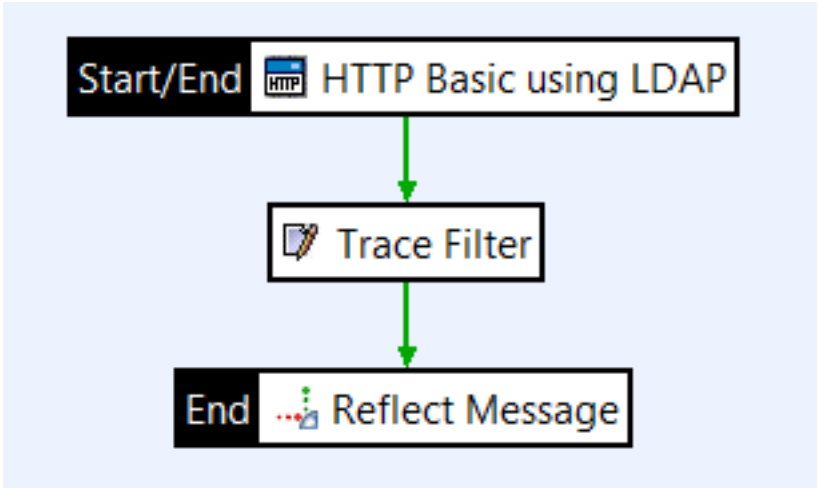
1. Add a new policy named, for example, `LDAP Authentication`.
2. Open the **Authentication** category in the filter palette, and drag an **HTTP Basic** filter onto the policy canvas.
3. Set the following, and click **Finish**:
   - **Credential Format**: `User Name`
   - **Repository Name**: The LDAP repository you configured (`LDAP Repository`)

   For more details on the fields and options in this configuration window, see "HTTP basic authentication" in the *API Gateway Policy Developer Filter Reference*.
4. Click on the **Add Relative Path** icon to create a new relative path (for example, `/ldap`) that links to this policy, and deploy the policy to API Gateway.

Test the policy

To test that the policy works and trace the operation in the log files, add a **Reflect** and a **Trace** filter to the policy. These filters are not required in the production environment.

1. Open the **Utility** category in the palette, drag a **Reflect** filter onto the policy canvas, and click **Finish**.
2. Drag a **Trace** filter onto the policy canvas, and click **Finish**.
3. Connect the filters with success paths.



4. Deploy the updated configuration to API Gateway.
5. Test the policy (for example, using the `sr` command). See the *API Gateway Policy Developer Guide* for more information on testing tools.

Retrieve attributes from the directory server

You can enhance your LDAP policy to retrieve attributes for the user after a successful authentication. API Gateway stores the credentials of the user in the `authentication.subject.id` message attribute. Typically, this contains the Distinguished Name (DN) or user name of the authenticated user. API Gateway extracts the name from the message attribute and uses the name to query the directory server.

1. Open the **Attributes** category in the filter palette, and drag a **Retrieve from Directory Server** filter onto the policy canvas.
2. In **LDAP Directory**, select the LDAP connection you configured (`LDAP connection`).
3. Select **From Selector Expression**, and enter `${authentication.subject.id}` to obtain the value of this message attribute at runtime.
4. In **Base Criteria**, enter the Base Criteria of your directory server.
5. In **Search Filter**, enter the following:

   ```
   (&(objectclass=<User Class>)(<User Search Attribute>=<value>))
   ```

   For example:

   ```
   (&(objectclass=inetOrgPerson)(CN=Administrator))
   ```

   The format may vary depending on your directory server.

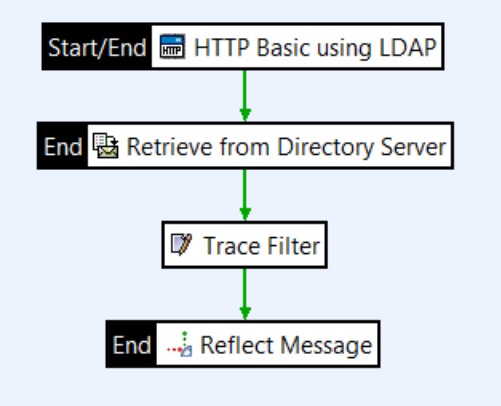
6. In **Attribute Name** table, list the attributes you want to retrieve from the user profile. If no attributes are explicitly listed, API Gateway extracts all user attributes. The retrieved attributes are set to `attribute.lookup.list` for that user.

   For example, a user `CN=Administrator` could have an attribute `memberOf` with the value `CN=Group Policy Creator Owners`. If you choose to retrieve the Attribute Name `memberOf`, API Gateway returns the value `CN=Group Policy Creator Owners`.

7. Click **Finish**.

   For more details on the fields and options in this configuration window, see "Retrieve attribute from directory server" in the *API Gateway Policy Developer Filter Reference*.

8. Connect the filters with success paths.



Validate a retrieved attribute

After retrieving user attributes, API Gateway can validate a retrieved attribute value using an **Evaluate Selector** filter. Based on whether the filter fails or passes, API Gateway can then make a decision in the policy, such as allow the user to access a particular resource.

In this example, API Gateway checks if the user `CN=Administrator` belongs to the group "Policy Creator Owners".

1. Open the **Utility** category in the palette, and drag an **Evaluate Selector** filter onto the policy canvas.
2. In the **Expression** field, enter the selector expression to evaluate, and click **Finish**:

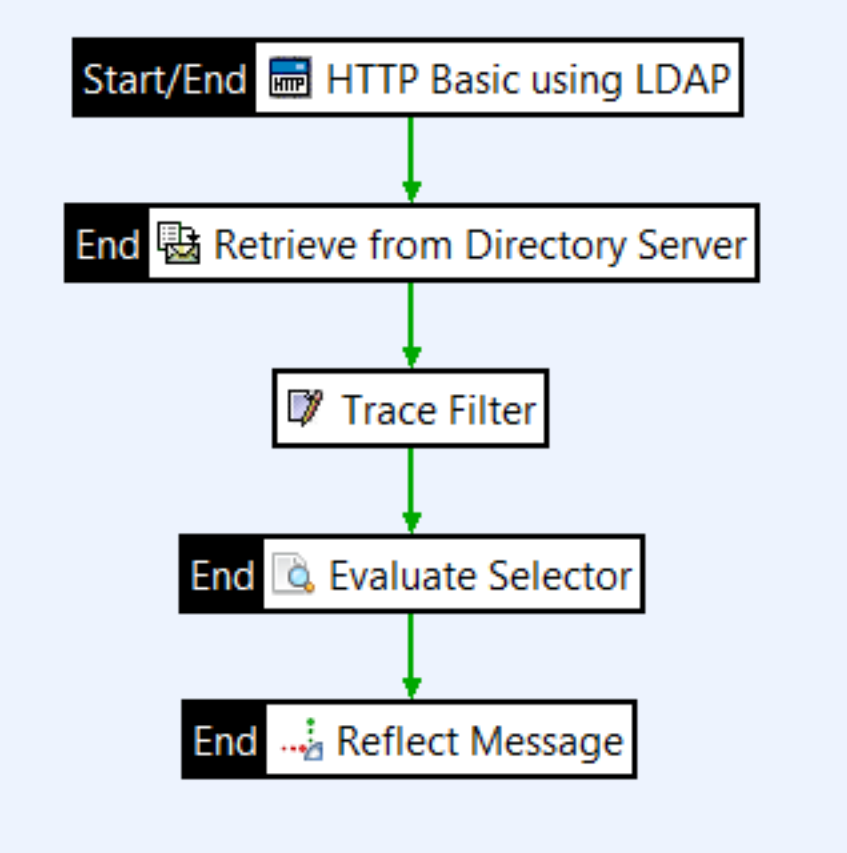
   ```
   ${<user>[<place in attribute.lookup.list>].<Attribute Name>.contains("<attribute value>")}
   ```

   For example:

   ```
   ${Administrator[0].memberOf.contains("CN=Group Policy Creator Owners,CN=Users,DC=axwayqa,DC=com")}
   ```

   The selector expression reads as follows:

   - Check if the retrieved attribute `memberOf` in the `attribute.lookup.list` for the user `Administrator` contains the value `CN=Group Policy Creator Owners,CN=Users,DC=axwayqa,DC=com`.
   - If this matches, the filter passes.
3. Connect the filters with success paths. For testing, connect the **Evaluate Selector** filter between the **Trace** and **Reflect Message** filters.



This example prints the following in the trace file:

```
DEBUG   14/02/2013 16:54:21.205 }
DEBUG  14/02/2013 16:54:21.205  user {
DEBUG  14/02/2013 16:54:21.205  Value: [CN=Administrator: null:null:
{memberof=memberOf: CN=Group Policy Creator Owners,CN=Users,DC=axwayqa,DC=com,
CN=Domain Admins,CN=Users,DC=axwayqa,DC=com, CN=Enterprise
Admins,CN=Users,DC=axwayqa,DC=com,
CN=Schema Admins,CN=Users,DC=axway,DC=com,
CN=Administrators,CN=Builtin,DC=axwayqa,DC=com}]
```

For more information on configuring selectors, see the "Select configuration values at runtime" in the *API Gateway Policy Developer Guide*.

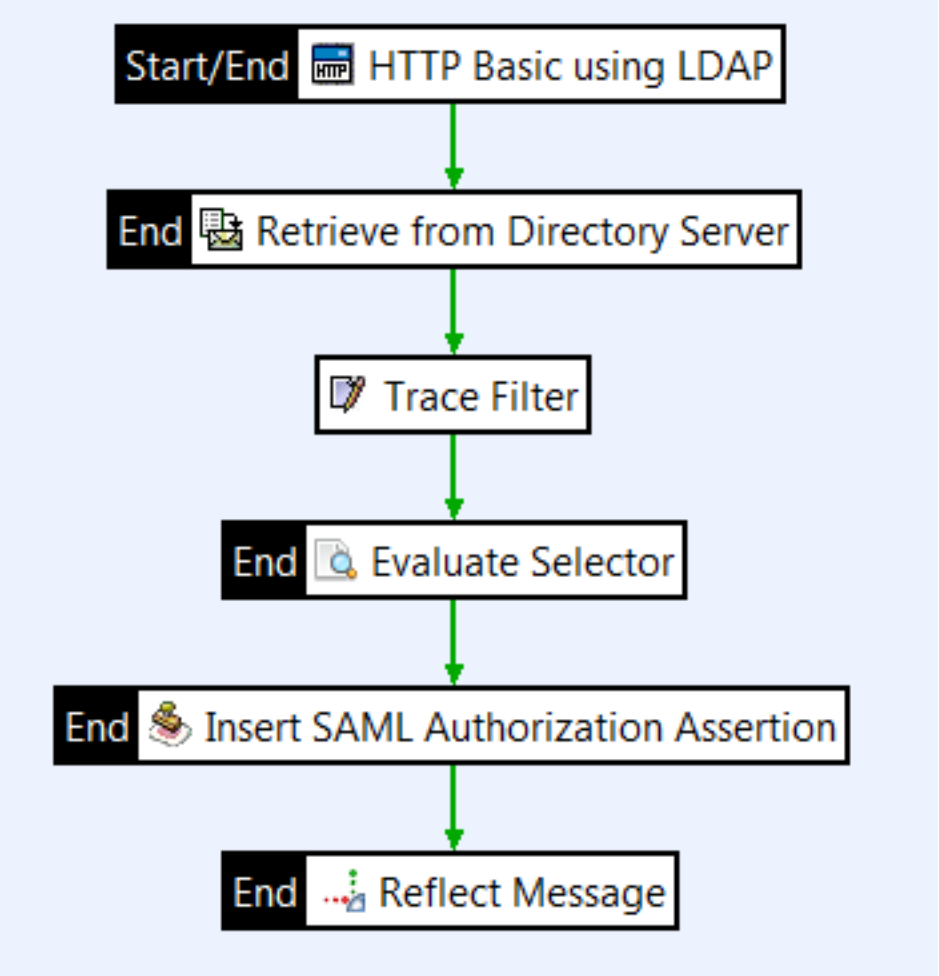
Insert a SAML token

You can extend the authentication to downstream web services, if required. After the end user is successfully authenticated and the attibutes retrieved, API Gateway adds a SAML authentication assertion to the response message for the web services to consume.

1. Open the **Authentication** category in the palette, and drag an **Insert SAML Authentication Assertion** filter onto the policy canvas.

2. On the **Assertion details** tab, select any issuer on the **Issuer Name** list, and set the expiry date for the SAML authentication assertion.
3. On the **Assertion Location** tab, make sure that **Add to WS-Security Block with SOAP Actor/Role** is selected and **SOAP Actor/Role** set to **Current Actor/Role Only**.
4. On the **Advanced** tab, select **Insert SAML Attribute Statement** and **Indent**, then click **Finish**.
5. Connect the filters with success paths.

   For more details on how to configure an **Insert SAML Authentication Assertion** filter, see "Insert SAML authorization assertion" in the *API Gateway Policy Developer Filter Reference*.



Secure the connection to the directory server

For security, you can use an SSL connection between API Gateway and your directory server. This section describes how to configure this in Policy Studio. For more information on working in Policy Studio, see the *API Gateway Policy Developer Guide*.

API Gateway and Policy Studio require the CA certificate of your directory server. You must import the CA certificate into the API Gateway and Policy Studio Java keystores.

Add the LDAP server certificate to the API Gateway certificate store

1. In the node tree, click **Environment Configuration > Certificates and Keys > Certificates**.
2. Click **Create/Import > Import Certificate**, and select the CA certificate of your directory server.
3. In **Alias Name**, give the certificate a name or click **Use Subject** to use the subject name , then click **OK**.

Add the LDAP server certificate to the API Gateway Java keystore

1. In the node tree, click **Environment Configuration > Certificates and Keys > Certificates**.
2. Click **Keystore**, click the browse button next to the **Keystore** field, and browse to the keystore file:

   ```
   INSTALL_DIR/apigateway/posix/jre/lib/security/cacerts
   ```

3. Click **Open**, and enter the keystore password.
4. Click **Add to keystore**.
5. Select the CA certificate of your directory server, and click **OK**.
6. Give a name to the certificate, or use the default name, and click **OK**.
7. Click **OK** to save the configuration, and deploy the updated configuration to API Gateway.

Add the LDAP server certificate to the Policy Studio Java keystore

1. In the node tree, click **Environment Configuration > Certificates and Keys > Certificates**.
2. Click **Keystore**, click the browse button next to the **Keystore** field, and browse to the keystore file:

   ```
   INSTALL_
   DIR/policystudio/posix/jre/lib/security/cacerts
   ```

3. Click **Open**, and enter the keystore password.
4. Click **Add to keystore**.
5. Select the CA certificate of your directory server, and click **OK**.
6. Give a name to the certificate, or use the default name, and click **OK**.
7. Click **OK** to save the configuration, and restart Policy Studio.

Configure the LDAP connection over SSL

1. In the node tree, click **Environment Configuration > External Connections > LDAP Connections**.
2. Right-click the appropriate directory server connection, and click **Edit**.
3. In the **URL** field, enter the LDAPS host name and port. For example:

```
ldaps://ldap_host:636
```

4. Select the **SSL Enabled** check box, and click **Test Connection**.
5. After a successful connection, click **OK**, and deploy the deploy the updated configuration to API Gateway.

CA SiteMinder integration 2

CA SiteMinder is a centralized web access management system that provides user authentication and single sign-on, policy-based authorization, identity federation, and auditing of access to web applications and portals. This section describes how to configure API Gateway 7.6.2 to authenticate and authorize end users using CA SiteMinder 12.52.

CA SiteMinder authenticates end users and authorizes them to access protected web resources. API Gateway can request SiteMinder to authenticate end users using the user profiles stored in the SiteMinder server. SiteMinder decides whether the user should be authenticated, and API Gateway then enforces this decision. API Gateway can also request SiteMinder to make authorization decisions on behalf of end users that have successfully authenticated to API Gateway.

For more information on CA SiteMinder, go to the CA Technologies website.

Flow description

SiteMinder authentication flow starts when an end user uses browser to attempt to access a resource protected with CA SiteMinder on API Gateway. First, API Gateway checks if the request from the user contains a valid session cookie for SiteMinder. If API Gateway does not find a valid session cookie the flow is as follows:

1. End user is prompted to provide the login credentials.
2. API Gateway forwards the credentials to SiteMinder.
3. SiteMinder decides whether the user is authenticated and authorized for the requested resource. If the authentication is successful, SiteMinder returns a session cookie and a response to API Gateway.
4. API Gateway stores the session cookie from SiteMinder in the `siteminder.session` message attribute and creates a custom cookie.
5. API Gateway returns a message with the cookie and a success response to the end user, and authorizes the user to access the requested resource.

On subsequent requests to access the protected resource, the end user sends the cookie to API Gateway in the request message. API Gateway retrieves the cookie and validates it against SiteMinder. The end user stays authenticated for the entire lifetime of the session cookie. As long as the session cookie is valid, API Gateway does not need to re-authenticate the end user against SiteMinder for every request. This increases throughput and performance considerably.

Prerequisites

Before you start, you must have the following:

- API Gateway installed
- CA SiteMinder installed and configured

For more details on the installation and the initial configuration of CA SiteMinder, see the CA SiteMinder documention. It is recommended you familiarize yourself with this documentation before you start integrating API Gateway with CA SiteMinder.

Configuration process

The example policy is build in stages, starting with a simple authentication and authorization policy for SiteMinder that is then refined by adding further filters.

The example policy uses HTTP Basic to authenticate the end user, but you can replace it with another authentication mechanism, if required.

The following steps are required to integrate API Gateway with CA SiteMinder:

1. *Configure API Gateway as the SiteMinder agent* on page 24.
    - *Configure API Gateway as the SiteMinder agent* on page 24.
    - *Register API Gateway as the SiteMinder agent* on page 25.
2. *Configure SiteMinder connection* on page 26.
3. *Configure SiteMinder authentication policy* on page 27.
    - *Create an authentication repository* on page 27.
    - *Configure routing to the protected resource* on page 27.
    - *Configure the SiteMinder authentication and authorization policy* on page 27.
4. *Configure single sign-on* on page 29.
    - *Configure custom cookie creation* on page 29.
    - *Configure the cookie check* on page 30.
    - *Configure SiteMinder session validation* on page 31.

Configure API Gateway as the SiteMinder agent

This section describes how to configure API Gateway to act as an agent for CA SiteMinder.

- *Add CA binaries to API Gateway* on page 25
- *Register API Gateway as the SiteMinder agent* on page 25

Add CA binaries to API Gateway

Integration with CA SiteMinder requires CA SiteMinder SDK version 12.52-sp02 or later. You must add the required third-party binaries to your API Gateway installation.

1. Ensure that any SiteMinder binaries you may have previously added to API Gateway have been deleted.
2. Install CA SiteMinder SDK.
3. Copy the following `jar` files from the `java` directory of the CA SDK :
   - `cryptoj.jar`
   - `smagentapi.jar`
   - `smjavasdk2.jar`
4. Add the files to the following directory on API Gateway:

   ```
   INSTALL_DIR/apigateway/<platform>/lib
   ```

5. Restart API Gateway.

Register API Gateway as the SiteMinder agent

Before API Gateway can act as a Policy Enforcement Point (PEP) for SiteMinder, you must register API Gateway as an agent for CA SiteMinder Policy Server. With the `smreghost` tool, you can register the agent from the command line, and create the `SmHost.conf` file.

The agent needs to be registered on the machine running the API Gateway instance. You must run the `smreghost` tool separately on each individual API Gateway instance. The `SmHost.conf` file is generated in the same directory as the `smreghost` tool.

Before you start, you need the following information on your CA SiteMinder Policy Server:

- IP address
- Login credentials (user name and password)
- Host Configuration Object name

To obtain this information, contact your SiteMinder administrator.

1. Go to `INSTALL_DIR/apigateway/<platform>/lib`.
2. Run the following:

   ```
   export LD_LIBRARY_PATH=INSTALL_
   DIR/apigateway/<platform>/lib
   ```

3. Run the `smreghost` tool as follows:

```
smreghost -i <CA SiteMinder IP address> -u <user name>
-p <password> -hc <Host Configuration Object name> -hn
<hostname
```

For example:

```
smreghost -i 192.168.0.99 -u GatewayAgent -p XXXXXX -
hc V6HostConfObject -hn apigateway.axway.int
```

The hostname must be the fully qualified machine name of the host machine running API Gateway.

4. To enable debug output from the Siteminder agent, add the following to `jvm.xml` if needed:

```
<ConfigurationFragment>
<VMArg name="-DSMJAVASDK_LOG_INFO=true"/>
</ConfigurationFragment>
```

Configure SiteMinder connection

This section describes how to configure API Gateway to connect to CA SiteMinder using Policy Studio. For more details on working in Policy Studio, see *API Gateway Policy Developer Guide*.

Before you start, you need the following information for your CA SiteMinder Policy Server:

- Web Agent name
- Agent Configuration Object name

To obtain this information, contact your SiteMinder administrator.

1. In the node tree, click **Environment Configuration > External Connections > SiteMinder/SOA Security Manager Connections**.
2. Select **Add a SiteMinder connection**.
3. Enter your agent name (`apigateway.axway.int`) and agent configuration object name (`V6HostConfObject`) you created in *Register API Gateway as the SiteMinder agent* on page 25.
4. Click **Browse**, select the `SmHost.conf` file for your agent, and click **OK**.

For more details on the fields and options in this configuration window, see "Configure SiteMinder/SOA Security Manager connections" in the *API Gateway Policy Developer Guide*.

Configure SiteMinder authentication policy

This section describes how to configure an authentication policy in Policy Studio for API Gateway to authenticate to CA SiteMinder. For more details on working in Policy Studio, see *API Gateway Policy Developer Guide*.

Create an authentication repository

1. In the node tree, click **Environment Configuration > Authentication Repositories> SiteMinder Repositories**.
2. Select **Add a new Repository**, and enter a name for your repository (for example, `SiteMinder repository`).
3. Set **Agent Name** to the agent you registered (`GatewayAgent`), and click **OK**.

For more details on the fields and options in this configuration window, see "CA SiteMinder repositories" in the *API Gateway Policy Developer Guide*.

Configure routing to the protected resource

1. In the node tree, click **Policies**.
2. Add a new policy , and enter a name for it (`Route to protected resource`).
3. Open the **Routing** category in the palette, drag a **Connect to URL** filter onto the policy canvas, and enter a name for your filter (`Route to protected resource`).
4. Enter the URL for your protected resource, and click **Finish**.
5. Right click the filter, and select **Set as Start**.

The SiteMinder authentication and authorization policy calls to this protected routing policy using a **Policy Shortcut** filter.

For more details on the fields and options in this configuration window, see "Connect to URL" in the *API Gateway Policy Developer Filter Reference*.

Configure the SiteMinder authentication and authorization policy

This sections describes how to configure a policy to authenticate and authorize an end user existing in a CA SiteMinder repository.

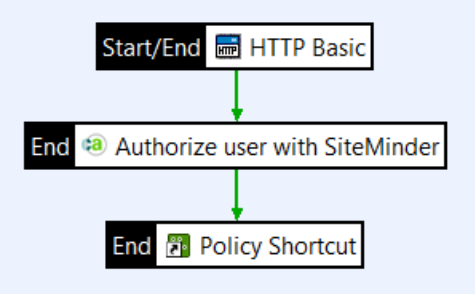
To start, add a new policy named, for example, `SiteMinder`.

1. Open the **Authentication** category, and drag a **HTTP Basic** filter onto the policy canvas.
2. Set the following, and click **Finish**:
   - **Credential Format**: `User name`
   - **Repository name**: `<your SiteMinder repository>` (`SiteMinder repository`)

     For more details on the fields and options in this configuration window, see "HTTP basic authentication" in the *API Gateway Policy Developer Filter Reference*.
3. Open the **CA SiteMinder** category, and drag a **Authorization** filter onto the policy canvas.
4. Enter a name for the filter (for example, `Authorize user with SiteMinder`), and click **Finish**.

   For more details on the fields and options in this configuration window, see "SiteMinder authorization" in the *API Gateway Policy Developer Filter Reference*.
5. Open the **Utility** category, drag a **Policy Shortcut** filter onto the policy canvas.
6. Set **Policy Shortcut** to the routing policy you created (`Route to protected resource`), and click **Finish**.
7. Connect the filters with a success path.
8. Click on the **Add Relative Path** icon to create a new relative path (for example, `/siteminder`) that links to this policy.
9. Deploy the new configuration to API Gateway.

The policy looks like this:



The policy has the following flow:

- API Gateway authenticates the end user using HTTP Basic.
- API Gateway passes the end user's credentials to SiteMinder.
- SiteMinder authenticates the end user, authorizes the end user for the particular resource, and sends a response to API Gateway.
- API Gateway routes the request to a policy shortcut calling the protected resource.

Configure single sign-on

In a production deployment, it is not practical to authenticate and authorize each user for each request. Instead, you can configure API Gateway to store the SiteMinder session cookie for single sign-on (SSO).

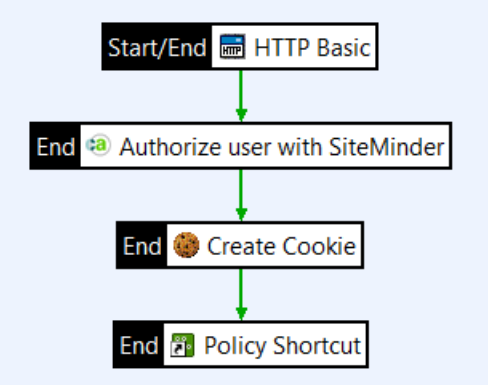
After an end user is successfully authenticated, API Gateway can create a custom cookie and place the SiteMinder session cookie validated for that end user as the cookie value. On later request, instead of re-authenticating the user every time, API Gateway can check if the request contains a valid session cookie. If a valid session cookie is found, the end user does not have to be authenticate again. API Gateway can also insert a SAML authorization assertion for downstream web services to consume.

This section expands the previously configured SiteMinder authentication policy. To start, copy your SiteMinder authentication and authorization policy (`SiteMinder`), and rename it (for example, `SiteMinder Single Sign-On`.

Configure custom cookie creation

1. Open the **Conversion** category, and drag a **Create Cookie** filter onto the policy canvas.
2. Set the following:
    - **Cookie Name**: `smcookie`
    - **Cookie Value**: `${siteminder.session}`
    - **Path**: /
3. Set **Max-Age** to how long you want the cookie to remain valid, and click **Finish**.
4. Connect the **Authorization** filter (`Authorize user with SiteMinder`) to the

Create Cookie filter with a success path.

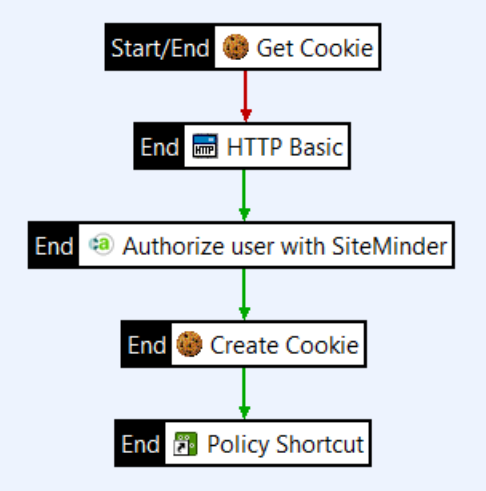


The message attribute `${siteminder.session}` was specified in *Create an authentication repository* on page 27. After the end user is successfully authenticated to SiteMinder, this is the message attribute that contains the end user's SiteMinder session cookie.

For more details on the fields and options in this configuration window, see "Create cookie" in the *API Gateway Policy Developer Filter Reference*.

Configure the cookie check

1. Open the **Attributes** category, and drag a **Get Cookie** filter onto the policy canvas.
2. Set the **Cookie Name** to `smcookie`.
3. Select **Fail if cookie not found in the message**, and click **Finish**.
4. Right-click the **Get Cookie** filter, and select **Set as Start**.
5. Connect the **Get Cookie** filter to the **HTTP Basic** filter with a failure path.



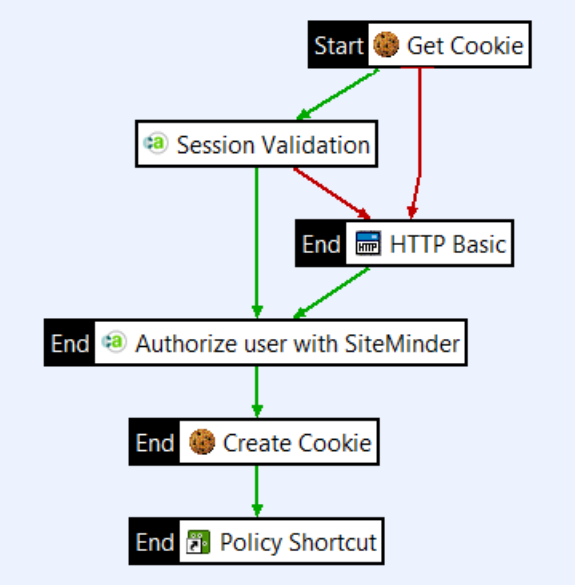
If the request from then end user does not contain a valid SiteMinder cookie, the end user is prompted to provide login credentials.

For more details on the fields and options in this configuration window, see "Get cookie" in the *API Gateway Policy Developer Filter Reference*.

Configure SiteMinder session validation

If API Gateway finds a valid session cookie from an incoming request, it uses the cookie value to validate the end user's SiteMinder session.

1. Open the **CA SiteMinder** category, and drag a **Session Validation** filter onto the policy canvas.
2. Set **Agent Name** to your registered SiteMinder agent (`GatewayAgent`).
3. Set **Selector Expression to retrieve session** to `${cookie.smcookie.value}`, and click **Finish**.
4. Connect the **Get Cookie** filter to the **Session Validation** filter with a success path.
5. Connect the **Session Validation** filter to the **Authorization** filter (`Authorize user with SiteMinder`) with a success path.
6. Connect the **Session Validation** filter to the **HTTP Basic** filter with a failure path.



If the SiteMinder session cannot be validated, the end user is prompted to provide login credentials.

For more details on the fields and options in this configuration window, see "SiteMinder session validation" in the *API Gateway Policy Developer Filter Reference*

Deploy the policy

1. Click on the **Add Relative Path** icon to create a new relative path (for example, `/siteminder_sso`) that links to this policy.

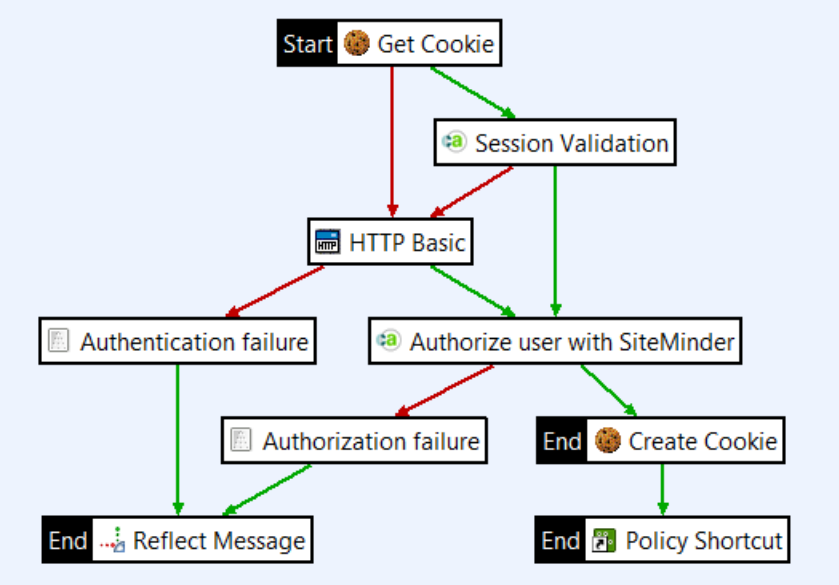
2. Deploy the new configuration to API Gateway.

You have now configured a simple policy for single sign-on in SiteMinder authentication.

The policy has the following flow:

- API Gateway checks if the request contains a valid custom cookie. If a valid custom cookie is not found, the end user is prompted to provide login credentials.
- If a valid custom cookie is found, API Gateway retrieves the value of the cookie and uses that to check if the end user's session cookie in SiteMinder is still valid. If the session cookie is no longer valid, the end user is prompted to provide login credentials.
- SiteMinder authorizes the user to access the requested resource and sends response with the SiteMinder cookie back to API Gateway.
- API Gateway creates a custom cookie to hold the end user's SiteMinder session cookie.
- API Gateway calls to the protected resource using the routing policy.
- API Gateway sends the response along with the custom cookie to the end user.

This policy can be part of a larger policy, including features such as XML threat detection and conditional routing (not described in this guide). You can also authenticate the user with some other authentication method instead of HTTP Basic. In addition, you can add additional filters, such as messages in case of failures:

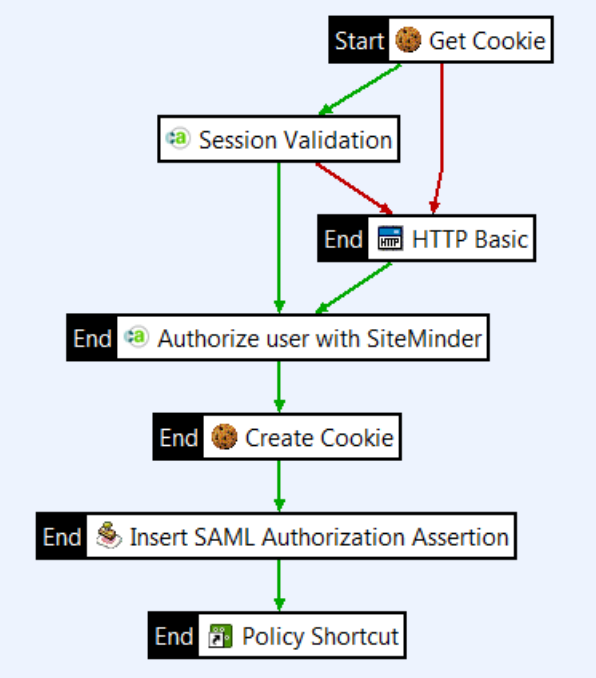


Configure inserting a SAML token

You can extend the single sign-on to downstream web services, if required. After the end user is successfully authenticated and authorized, API Gateway adds a SAML authentication assertion to the response message for the web services to consume.

1. Open the **Authorization** category, and drag a **Insert SAML Authentication Assertion** filter onto the policy canvas.

2. On the **Assertion details** tab, select any issuer on the **Issuer Name** list, and set the expiry date for the SAML authentication assertion.
3. On the **Assertion Location** tab, make sure that **Add to WS-Security Block with SOAP Actor/Role** is selected and **SOAP Actor/Role** set to **Current Actor/Role Only**.
4. On the **Advanced** tab, select **Insert SAML Attribute Statement** and **Indent**, then click **Finish**.
5. Connect the filter between the **Create Cookie** and the **Policy Shortcut** filters with success paths.



For more details on the fields and options in this configuration window, see "Insert SAML authorization assertion" in the *API Gateway Policy Developer Filter Reference*.

Test the policy

This section describe how to test your SiteMinder policy using a standard browser.

1. Open a web browser in private (incognito) mode to ensure no user is yet logged in.
2. Enter the URL to call your policy:

```
http://<ip address>:<port>/<path>
```

For example:

```
http://localhost:8080/siteminder_sso
```

3. Enter the login credentials. API Gateway authenticates the user against SiteMinder and returns the response along with the SiteMinder session cookie.
4. Refresh the browser to access the protected resource. Because the custom cookie (`smcookie`) is available this time, API Gateway does not prompt for credentials. Instead of re-authentication, API Gateway validates the cookie against the SiteMinder.

The **Traffic Monitor** tab on the API Gateway Manager (`https://localhost:8090`) is an excellent place to view and troubleshoot the message flows. For more details, see "Monitor services in API Gateway Manager" in the *API Gateway Administrator Guide*.

RSA Access Manager integration 3

RSA Access Manager (formerly RSA ClearTrust) provides Identity Management and access control services for web applications. It centrally manages access to web applications, ensuring that only authorized users are allowed access to resources. You can configure API Gateway to act as a client to the RSA Access Manager, and leverage the user information stored in RSA Access Manager for user authentication and authorization.

API Gateway uses the **Access Manager** filter to query authorization information for a particular user on a given resource from RSA Access Manager and to make the authorization decision. If the user has been authorized for the resource in question, the request is allowed through to the service. Otherwise, the request is rejected. In addition, you can also configure an authentication filter to authenticate users against a RSA Access Manager authentication repository.

Integration with RSA Access Manager requires RSA Access Manager SDK version 6.2 or server installation libraries.

For more details on the product, see the RSA Access Manager documentation.

RSA Access Manager server connections

API Gateway can connect to a number of Access Manager authorization servers or dispatcher servers using *connection sets* (connection groups). A connection set consists of a globally configured set of servers that API Gateway connects to. API Gateway round-robins between groups of external servers, providing a high degree of failover. If one of the servers becomes unavailable, API Gateway can use one of the other servers in the group.

You can deploy multiple Access Manager authorization servers for load-balancing purposes. In this case, API Gateway first connects to a dispatcher server that returns a list of active authorization servers. If the first dispatcher server in the connection group is not available, API Gateway attempts to connect to the dispatcher server with the next highest priority in the group. If a dispatcher server has not been deployed,API Gateway can connect directly to an authorization server. API Gateway then attempts to connect to one of the authorization servers.

API Gateway attempts to connect to the listed servers according to the priorities assigned to them. For example, a connection group could include two high-priority servers, one medium-priority server, and one low-priority server configured. If API Gateway can successfully connect to the two high-priority servers, it alternates requests only between these two servers, and the other servers in the group are not used. However, if *both* high-priority servers are unavailable, API Gateway attempts to connect to the medium-priority server. Only if the medium-priority server fails as well, API Gateway connects to the low-priority server.

Flow description

1. An end user attempts to access a resource protected with RSA Access Manager on API Gateway.
2. API Gateway authenticates the end user against the RSA Access Manager authentication repository.
3. API Gateway requests RSA Access Manager to make the security decision for the user.
4. RSA Access Manager makes the security decision based on the user information, and returns the decision to API Gateway.
5. API Gateway enforces the security decision.
    - On success, API Gateway routes the message on to a configured target system.
    - On failure, API Gateway blocks the message and returns an error to the end user.

Prerequisites

Before you start, you must have the following:

- API Gateway installed
- RSA Access Manager v6.2 installed and configured

Configuration process

This guide provides a simple policy to demonstrate how API Gateway integrates with RSA Access Manager to authenticate and authorize users to specific resources. The example policy uses HTTP Basic to authenticate the end user, but you can replace it with another authentication mechanism, if required.

The following steps are required to integrate API Gateway with RSA Access Manager:

1. *Add RSA Access Manager binaries to API Gateway* on page 36.
2. *Configure RSA Access Manager connection* on page 37.
3. *Configure an RSA Access Manager authentication repository* on page 38.
4. *Configure API Gateway policy* on page 38.

Add RSA Access Manager binaries to API Gateway

You must copy RSA Access Manager libraries to API Gateway, so you must have RSA Access Manager installed on a server.

1. Copy the following files from the `lib` directory on your RSA Access Manager installation:
   - `axm-core-6.2.jar`
   - `cryptojce-6.1.jar`
   - `cryptojcommon-6.1.jar`
   - `jcm-6.1.jar`
2. Add the files to the `INSTALL_DIR/apigateway/ext/lib` directory on API Gateway:
3. Restart API Gateway.

Configure RSA Access Manager connection

This section describes how to configure a RSA Access Manager connection in Policy Studio. For details on working in Policy Studio, see the *API Gateway Policy Developer Guide*.

The RSA Access Manager connection is configured as a connection set. A connection set consists of a globally configured set of servers that API Gateway connects to. You can reuse these global sets when configuring policies in Policy Studio.

Note All servers in the connection set must be of the same type (authorization servers or dispatcher servers).

1. In the node tree, click **Environment Configuration > External Connections > Connection Sets**.
2. Select **RSA Access Manager Connection Sets**, and click **Add a Connection Set**.
3. Enter a name for the connection set (for example, `Authorization` or `Dispatch`), and click **Add** to add a server.
4. Enter the host name and port the server is listening on.
5. Select the security type for the server connection.

   Note The security type (**Clear**, **SSL (Anonymous)**, or **SSL Authentication**) you select must match the security requirement of the server.
6. If you selected **SSL Authentication**, click **Signing Key:**, and select the certificate you want to use, then select **OK**.
7. To change the priority of a server in the set, select the server, and click **Up** or **Down**.
8. Repeat for all the servers you want to include in the connection set, and click **OK**.

For more details on the fields and options in this configuration window, see "Configure connection groups" in the *API Gateway Policy Developer Guide*.

To later view or edit your connection sets, click **Environment Configuration > External Connections > Connection Sets**, double-click **RSA Access Manager Connection Sets**, and select the connection you want.

Configure an RSA Access Manager authentication repository

This section describes how to configure an RSA Access Manager authentication repository in Policy Studio. For details on working in Policy Studio, see the *API Gateway Policy Developer Guide*.

1. In the node tree, click **Environment Configuration > External Connections > Authentication Repositories**.
2. Select **RSA Access Manager Repositories**, and click **Add a new Repository**.
3. Enter a name for the repository (for example, `RSA Access Manager Repository`).
4. Select which server type API Gateway connects to (**Authorization Server** or **Dispatch Server**).
5. Select the connection group that API Gateway connects to.
6. Select the authentication type you plan to use, and click **OK**.

For more details on the fields and options in this configuration window, see "RSA Access Manager repositories" in *API Gateway Policy Developer Guide*.

Configure API Gateway policy

This section describes how to configure a policy for RSA Access Manager integration in Policy Studio. For more information on working in Policy Studio, see the *API Gateway Policy Developer Guide*.

The RSA Access Manager authentication repository is available from all authentication filters. Here, the example policy uses the **HTTP Basic** authentication filter to authenticate a client against a RSA Access Manager repository using a user name and password combination. You can configure a different authentication mechanism as required.

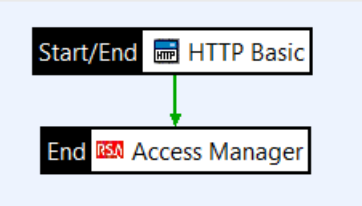
To start, add a new policy named, for example, `RSA Access Manager`.

1. Open the **Authentication** category in the palette, and drag a **HTTP Basic** filter onto the policy canvas.
2. Set the following, and click **Finish**:
   - **Credential Format**: `User Name`.
   - **Allow client challenge**: Select this.
   - **Repository Name**: The repository you configured (`RSA Access Manager Repository`).

   For more details on the fields and options in this configuration window, see "HTTP basic authentication" in the *API Gateway Policy Developer Filter Reference*.
3. Right click the filter, and select **Set as Start**.

4. Open the **Authorization** category in the palette, and drag an **Access Manager** filter onto the policy canvas.
5. Select which server type API Gateway connects to (**Authorization Server** or **Dispatch Server**), and select the connection group.
6. In **Server**, enter the name of the server that hosts the requested resource. The name entered must correspond to a preconfigured server name in RSA Access Manager.
7. In **Resource**, enter the name of the requested resource, and click **Finish**. The resource must be preconfigured in RSA Access Manager.
   For more details on the fields and options in this configuration window, see "RSA Access Manager authorization" in the *API Gateway Policy Developer Filter Reference*.
8. Connect the filters with a success path.
9. Click on the **Add Relative Path** icon to create a new relative path (for example, `/rsa`) that links to this policy.
10. Deploy the new configuration to API Gateway.

The policy looks like this:



The policy has the following flow:

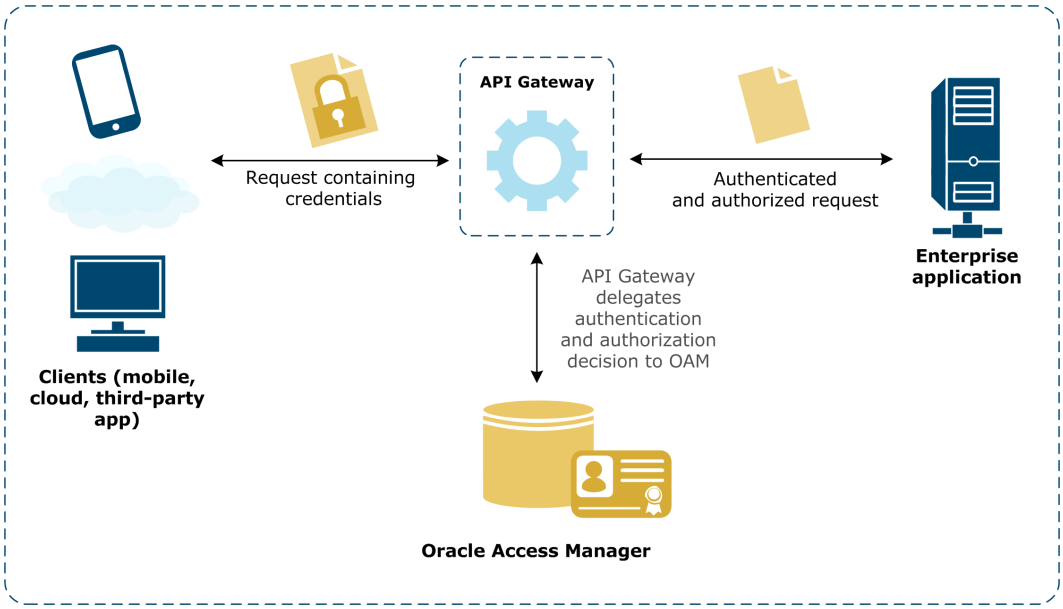
- API Gateway authenticates the end user using HTTP Basic.
- API Gateway passes the end user's credentials to RSA Access Manager.
- RSA Access Manager authenticates the end user, authorizes the end user for the particular resource, and sends a response to API Gateway.
- API Gateway relays the response to the end user.

Oracle Access Manager 11gR2 integration 4

This section describes how to configure API Gateway to authenticate and authorize user requests against Oracle Access Manager (OAM) 11gR2.

1. API Gateway is configured to authenticate a client against OAM using a user name and password.
2. Upon successful authentication, API Gateway authorizes the user against OAM.

The following overview diagram shows the message flow through API Gateway, which authenticates and authorizes a user for a particular resource against OAM before routing the message on to the web service.



Prerequisites

API Gateway

You must have installed API Gateway version 7.6.2 or higher and have received a valid license from Axway.

Access Server SDK

The Access Server SDK (ASDK) must be installed on the machine running API Gateway.

Oracle Access Manager

OAM 11gR2 must be installed and configured on the machine running API Gateway.

You should start it by using the following commands on UNIX-based systems (assuming a WebLogic domain of `idm_domain`, a server name of `oam_server1`, and a host name of `oam_host`).

Start WebLogic using the following command:

```
# cd ~/middleware/user_projects/domains/idm_domain/bin
# ./startWebLogic.sh
```

You can then start managed WebLogic using the following:

```
# cd ~/middleware/user_projects/domains/idm_domain/bin
# ./startManagedWebLogic.sh oam_server1 t3://oam_host:7001
```

Enter the user name of your administrator user when prompted:

```
Please enter your username :weblogic
Please enter your password :
```

OAM user

Create an OAM user called `weblogic` with the password `weblogic` in OAM to test the procedure. Refer to the OAM documentation for instructions on how to add a user.

cURL testing utility

To test the integration steps, the cURL testing utility is used to POST requests to API Gateway. This utility is available from the following URL:

http://curl.haxx.se/download.html

Alternatively, you can use any client capable of sending HTTP POST requests with HTTP basic authentication.

Configure Oracle Access Manager

This section describes how to create a 11g WebGate and configure an authentication policy for it using the OAM Administration Console. For more detailed instructions on OAM configuration, refer to the OAM documentation for your OAM version.

Configure an 11g WebGate with OAM 11gR2

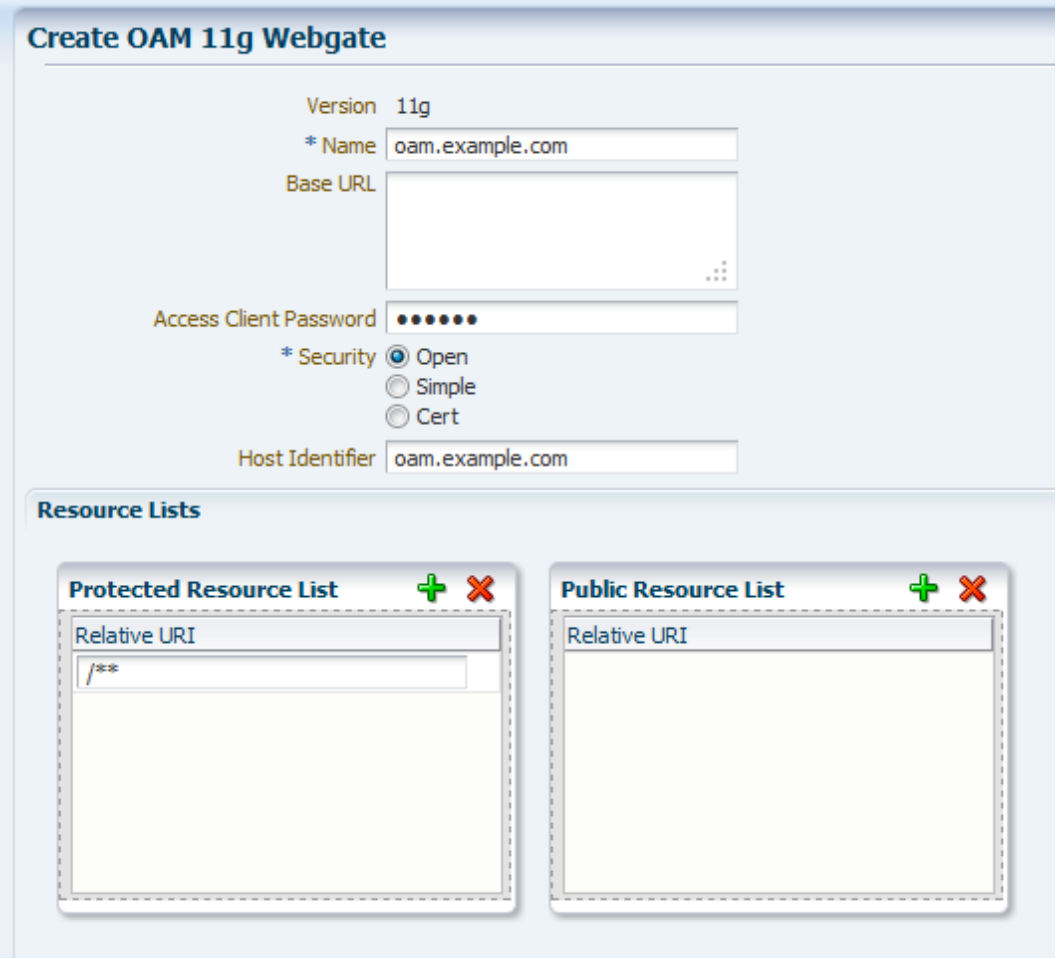
Use the web-based OAM Administration Console to create the new WebGate. The web interface is available at the following URL, where `OAM_HOST` refers to the IP or host name of the machine on which OAM is running:

```
http://OAM_HOST:7001/oamconsole
```

Log in using your WebLogic credentials and complete the following steps.

Step 1 - Create the 11g WebGate

1. On the **Welcome** page, click the **New OAM 11g Webgate** link.
2. Complete the following fields on the **Create OAM 11g Webgate** page:
    - **Name**: Enter a unique name for this OAM 11g WebGate, for example, `oam.example.com`.
    - **Access Client Password**: Enter a suitable password for this WebGate.
    - **Host Identifier**: Enter the host name of the machine on which your API Gateway and ASDK have been installed. In the following screenshot, the host name (that is to say **Host Identifier**) is used as the **Name** of the new WebGate.



3. Click **Apply** when you have completed the configuration.
4. Write down the location of the generated artifacts given in the confirmation message:

5. The complete configuration for the new WebGate is now displayed. You must enter a non-null value in the **Logout Target URL** field, for example, `end_url`.
6. Click **Apply** one more time to save the new **Logout Target URL**. You should see another confirmation message acknowledging the modification:

Step 2 - Configure the authentication policy

Update the authentication policy for the new WebGate:

1. Double-click the **Application Domains** node in the tree view under the **Policy Configuration** tab.
2. Click **Search** in the **Search** area. You do not need to enter anything in the **Search** field.

   You will be able to see your newly created 11g WebGate in the list of application domains.
3. Click the newly created `oam.example.com` WebGate link in the table.
4. Open the **Authentication Policies** tab to display the list of policies.
5. Click the `Protected Resource Policy` link in the table.
6. Change the **Authentication Scheme** field from `LDAPScheme` to `BasicScheme`.
7. Click **Apply**.

   You will see a confirmation message indicating that the update was successful.

Step 3 - Copy the WebGate artifacts to the API Gateway machine

Copy the auto-generated WebGate artifacts generated in *Step 1 - Create the 11g WebGate* on page 42.

As notified in the confirmation message, the artifacts were generated in the following location after creating the 11g WebGate:

```
/app/u01/middleware/user_projects/domains/idm_domain/output/oam.example.com
```

A directory listing on this location shows that two files were generated when the 11g WebGate was created: the `ObAccessClient.xml` and `cwallet.sso` files.

```
[oracle@oam oam.example.com]$ ls
cwallet.sso  ObAccessClient.xml
```

Both files must be copied from the OAM machine to the machine on which you have installed the 11g ASDK and are running API Gateway. They must be copied to the `ASDK_HOME/config` directory together with the `jps-config.xml` file.

```
$ ls -l /opt/oracle/AccessServerSDK11/config/
-rw-r--r--. 1 axway axway 3141 Feb 12 08:35 cwallet.sso
-rw-r--r--. 1 axway axway 1358 Feb 12 08:35 jps-config.xml
-rw-r--r--. 1 axway axway 3033 Feb 12 08:35 ObAccessClient.xml
```

Step 4 - Modify the API Gateway classpath

The API Gateway's classpath must be extended to include the ASDK 11 jars.

1. To achieve this, create a `jvm.xml` file at the following location:

```
INSTALL_DIR/apigateway/conf/jvm.xml
```

2. Edit this `jvm.xml` file so that its contents are as follows. Make sure to set the value of the `ASDK_HOME` environment variable to the location where you installed ASDK 11.

```
<ConfigurationFragment>

 <!-- OAM ASDK Settings -->
 <Environment name="ASDK_HOME" value="/opt/oracle/AccessServerSDK11" />
 <ClassDir name="$ASDK_HOME" />
 <VMArg  name="-Doracle.security.jps.config=$ASDK_HOME/config/jps-config.xml"/>

</ConfigurationFragment>
```

Configure API Gateway

This section describes how to configure API Gateway to work with Oracle Access Manager.

Start API Gateway

Refer to the *API Gateway Installation Guide* for instructions on how to start API Gateway.

Command example:

```
> startinstance -n "APIGateway1" -g "Group1"
```

Configure API Gateway to authenticate and authorize against OAM

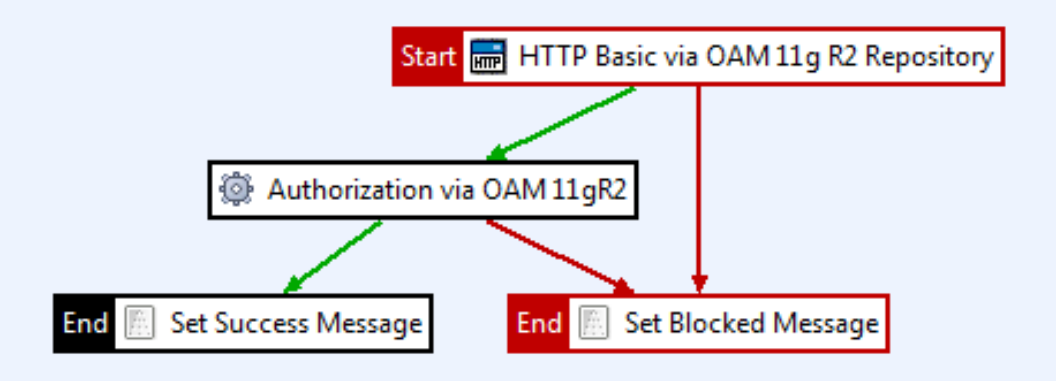
This section explains how to configure API Gateway to delegate authentication and authorization decisions to Oracle Access Manager.

The following steps are required:

- *Step 1 - Configure the OAM authentication repository* on page 46
- *Step 2 - Create a new policy* on page 48
- *Step 3 - Add the HTTP Basic Authentication filter* on page 49

- *Step 4 - Add the OAM Authorization filter* on page 49
- *Step 5 - Add the Success Message filter* on page 51
- *Step 6 - Add the Failure Message filter* on page 51
- *Step 7 - Add a relative path for the OAM authentication and authorization policy* on page 53
- *Step 8 - Deploy the policy* on page 54

The resulting policy created will appear as follows:



Step 1 - Configure the OAM authentication repository

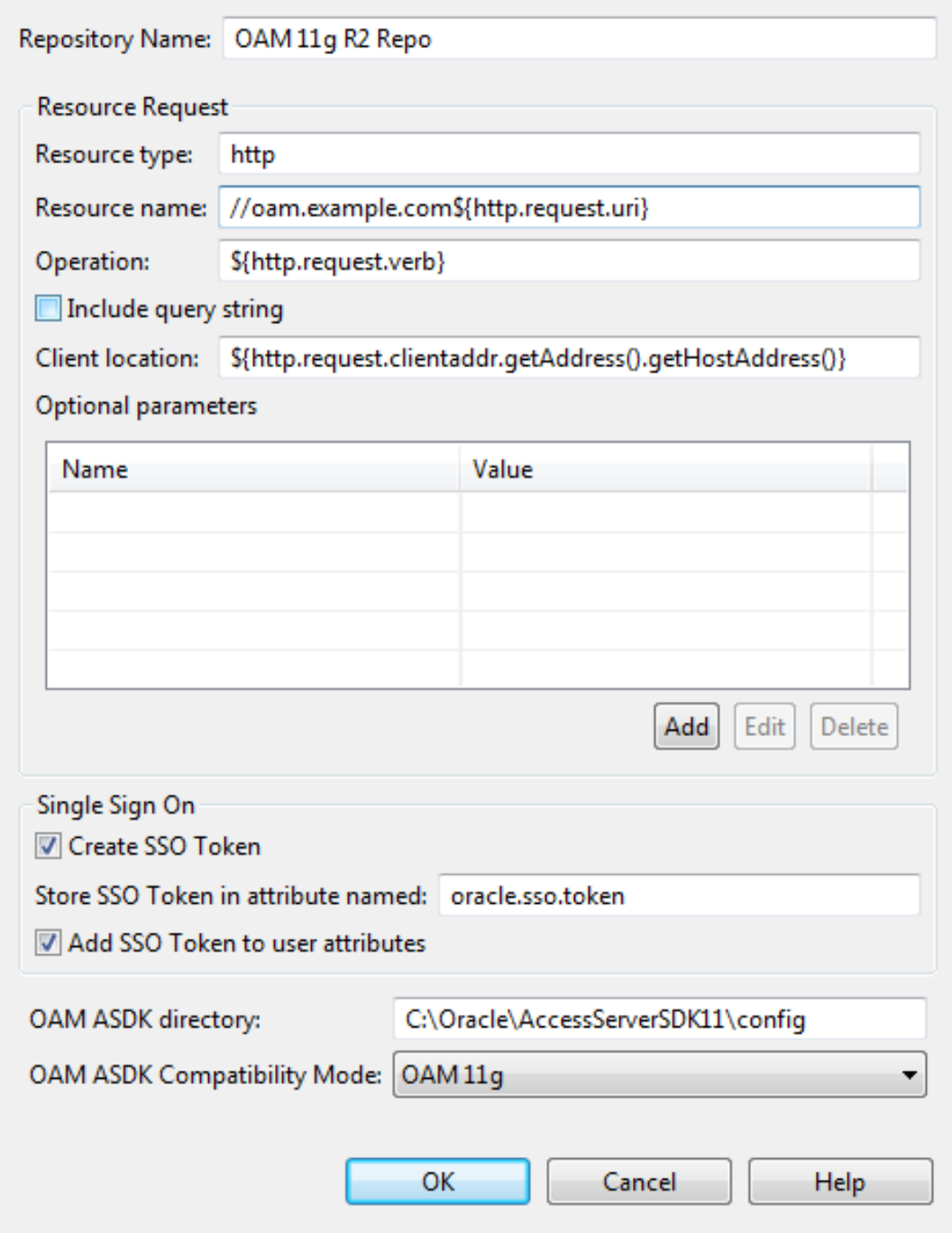
To configure the OAM authentication repository, perform the following steps:

1. In Policy Studio, expand the **Environment Configuration > External Connections** node in the tree.
2. Expand the **Authentication Repositories** node and click **Oracle Access Manager Repositories**.
3. Click the **Add a New Repository** link in the main window.
4. Configure the following fields in the **Authentication Repository** window. Refer to the *API Gateway Policy Developer Guide* for more details on the fields.
    - **Name**: Name of the authentication repository. For example: `OAM 11g R2 Repo`.
    - **Resource Type**: `http`
    - **Resource Name**:
      Enter the resource and host name. For example, `//oam.example.com${http.request.uri}`. Enter the exact same host name as you did for the **Host Identifier** field when creating the OAM 11g WebGate. The value is case-sensitive.
    - **Operation**: `${http.request.verb}`
    - **Client Location**: `${http.request.clientaddr.getAddress().getHostAddress()}`
    - **Create SSO Token**: Select the check box.
    - **Store SSO token in attribute named**: `oracle.sso.token`

- **Add SSO Token to user attributes**: This option is checked by default in order to store the OAM token for consumption by downstream OAM filters, for example, the OAM Authorization filter.
- **OAM ASDK Directory**:
  Enter: `C:\Oracle\AccessServerSDK11\config`
- **OAM ASDK Compatibility Mode**: Select **OAM 11g**.

5. Click **OK** to complete the configuration.

The following figure shows the OAM authentication repository configured to talk to an 11g OAM server:



Step 2 - Create a new policy

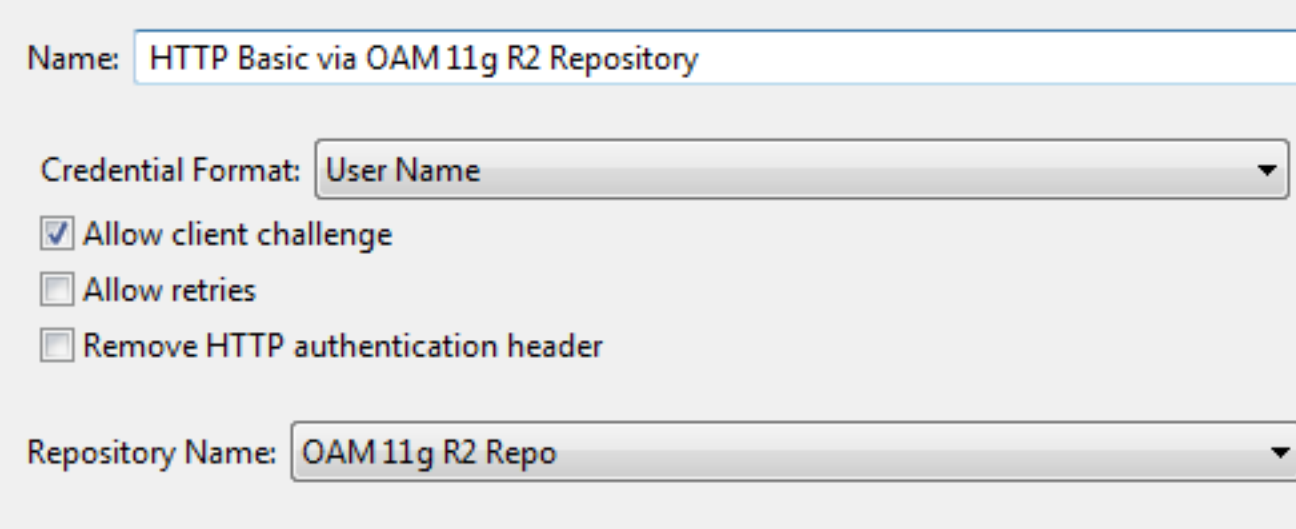
Create a new policy in Policy Studio called, for example, `OAM 11gR2 Authentication and Authorization`. The OAM authentication and authorization filters will be added here.

Step 3 - Add the HTTP Basic Authentication filter

Create an HTTP Basic Authentication filter and configure it to authenticate users against the OAM authentication repository created in *Step 1 - Configure the OAM authentication repository* on page 46.

1. Open the newly created `OAM 11gR2 Authentication and Authorization` policy.
2. Drag an **HTTP Basic** filter from the **Authentication** category in the palette and drop it onto the canvas and configure it as follows. Refer to the *API Gateway Policy Developer Guide* for more details on the fields.
   - **Name**: Name of the filter. For example: `HTTP Basic via OAM 11g R2 Repository`.
   - **Credential Format**: Select **User Name**.
   - **Allow Client Challenge**: Select the **Allow client challenge** check box.
   - **Repository Name**: Select **OAM 11gR2 Repo**.
3. Click **OK**.

   The completed configuration for the HTTP Basic Authentication filter is displayed as follows:



4. Right-click the filter and select the **Set as Start** menu option.

Step 4 - Add the OAM Authorization filter

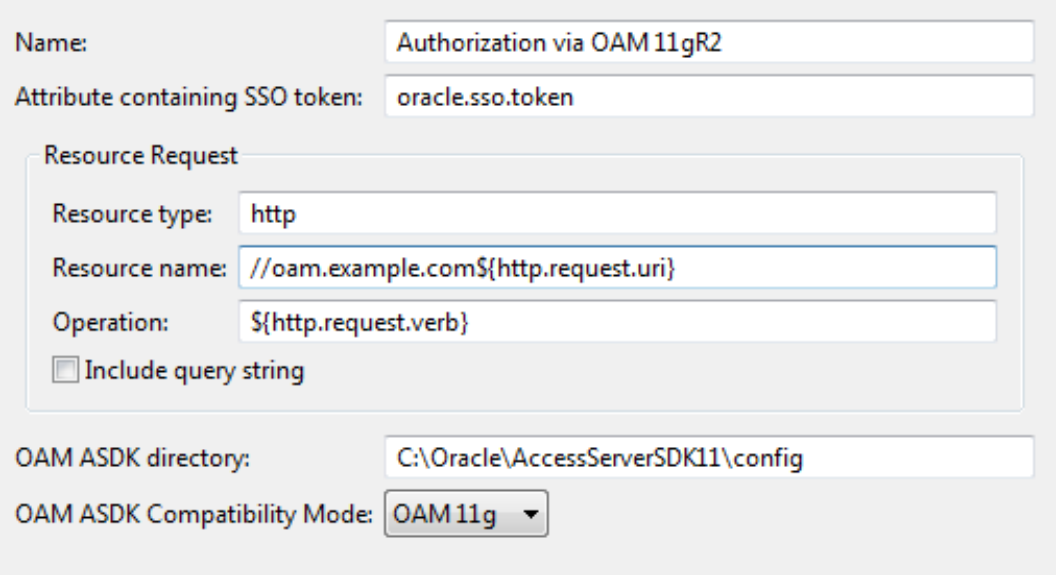
The next step is to add the OAM Authorization filter, which will authorize authenticated users against OAM.

1. From the **Oracle Access Manager category** in the palette of Policy Studio, drag the **Authorization** filter and drop it onto the **HTTP Basic Authentication** filter created in *Step 3 - Add the HTTP Basic Authentication filter* on page 49. By dropping a filter directly on to another filter, the new filter will be automatically connected to the first filter with a `success`

path.

2. Configure the fields on the filter as follows. Refer to the *API Gateway Policy Developer Guide* for more details on the fields.
   - **Name**: Enter a suitable name, such as `Authorization via OAM 11gR2`.
   - **Attribute Containing SSO Token**: Enter the name of the message attribute configured in the authentication repository earlier where the SSO token is stored, that is to say `oracle.sso.token`.
   - **Resource Type**: `http`
   - **Resource Name**:
     Enter the resource and host name. For example, `//oam.example.com${http.request.uri}`. Enter the exact same host name as you did for the **Host Identifier** field when creating the OAM 11g WebGate. The value is case-sensitive.
   - **Operation**: `${http.request.verb}`
   - **OAM ASDK Directory**:
     Enter: `C:\Oracle\AccessServerSDK11\config`
   - **OAM ASDK Compatibility Mode**: Select **OAM 11g**.
3. Click **OK**.

The following figure shows the OAM Authorization filter configured to talk to an 11g R2 OAM server:

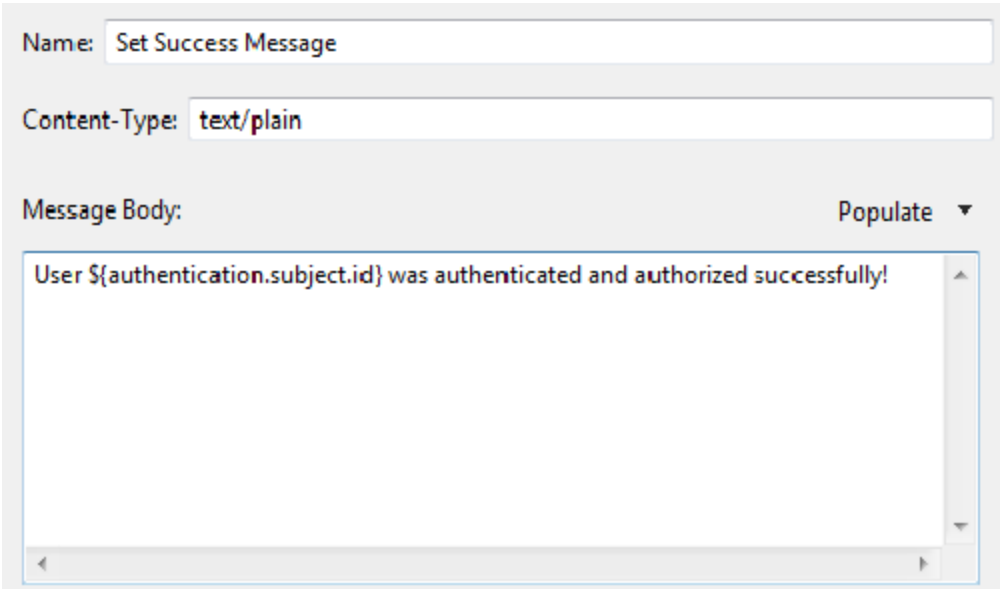


Step 5 - Add the Success Message filter

To display a success message to the user after successfully authorizing the user you can add a Set Message filter as follows. Refer to the *API Gateway Policy Developer Guide* for more details on the fields.

1. Drag a **Set Message** filter from the **Conversion** category in the palette and drop it onto the **OAM Authorization** filter created in *Step 4 - Add the OAM Authorization filter* on page 49.
2. Configure the following fields on this filter:
   - **Name**: Enter `Set Success Message`.
   - **Content-type**: `text/plain`
   - **Message Body**: `User '${authentication.subject.id}' was authenticated and authorized successfully!`
3. Click **OK**.

The configuration for the **Set Success Message** filter should now look like this:



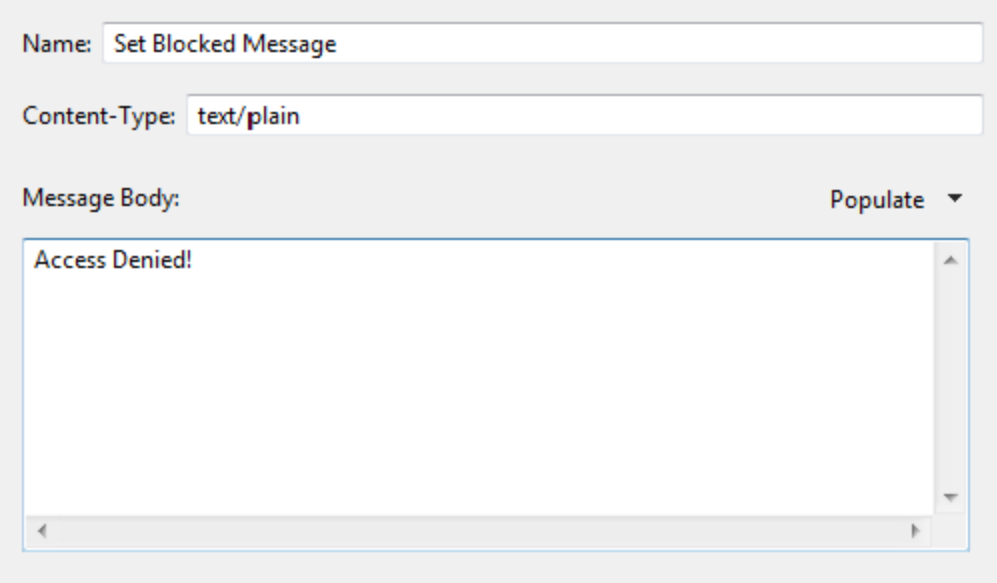
Step 6 - Add the Failure Message filter

If OAM fails to authenticate or authorize the user, an appropriate error message must be returned to the client. To display a failure message to the client after an unsuccessful authentication/authorization event you can add another **Set Message** filter as follows:

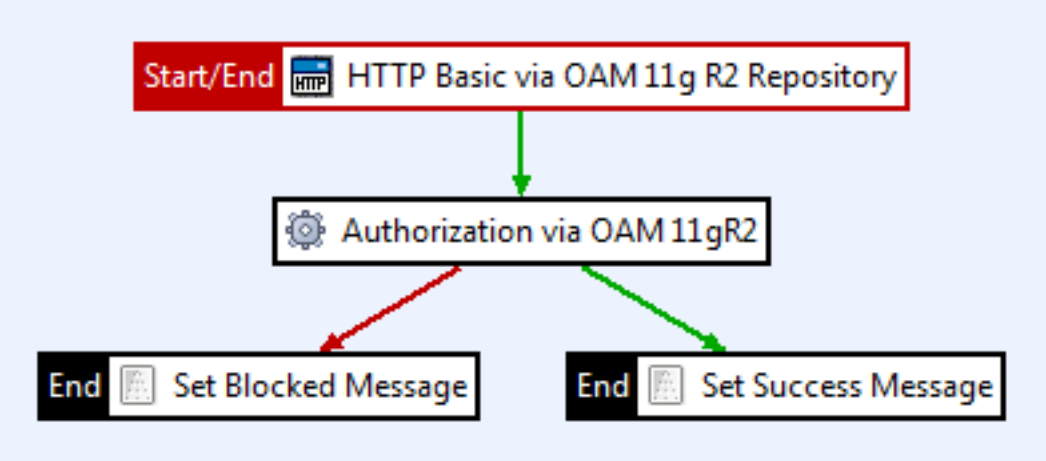
1. Drag a **Set Message** filter from the **Conversion** category in the palette and drop it onto the **OAM Authorization** filter. Because this filter already has the "Set Success Message" filter connected on its success path, the new **Set Message** filter will be automatically added on its *failure* path.

2. Configure the following fields on this filter. Refer to the *API Gateway Policy Developer Guide* for more details on the fields.
    - **Name**: Enter `Set Blocked Message`.
    - **Content-type**: `text/plain`
    - **Message Body**: `Access Denied!`
3. Click **OK**.

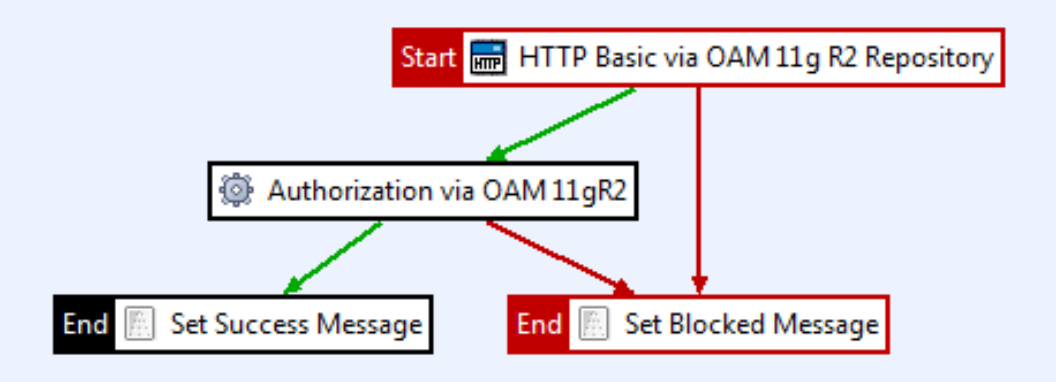
The configuration for the **Set Blocked Message** filter should now look like this:



The following figure shows the policy you have configured so far:



For completion, it would be useful to connect the Set Blocked Message filter to the HTTP Basic via OAM 11g R2 Repository filter along its failure path to get an appropriate failure message when authentication *and* authorization fail. Click the **Failure Path** item at the top of the palette to select it. Then simply click the **HTTP Basic via OAM 11g R2 Repository** filter and then click the **Set Blocked Message** filter to connect the failure path. The final policy is now displayed as follows:

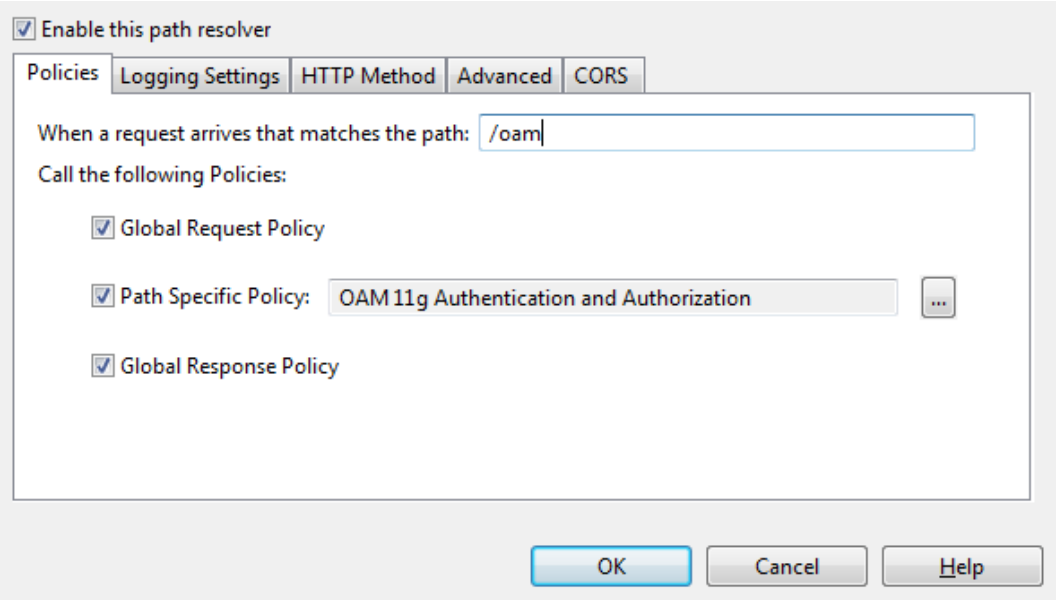


Step 7 - Add a relative path for the OAM authentication and authorization policy

In order for API Gateway to invoke the new policy for certain requests you must create a **Relative Path** and map it to the policy. All requests received on this path will be automatically forwarded to the policy for processing. For more information on relative paths, refer to the *API Gateway Policy Developer Guide*.

To add a **Relative Path** for this policy click **Add Relative Path** in the toolbar beneath the canvas.

Enter the path on which API Gateway will receive requests for this policy in the field provided in the **Resolve Path to Policies** dialog box:



Enter a relative path of `/oam` in the field provided. You can see that this path is automatically mapped to the **OAM 11g Authentication and Authorization** policy created earlier in this section.

Step 8 - Deploy the policy

To push the configuration changes to the live API Gateway instance you must deploy the new policy. You can do this by pressing the **F6** button.

Test the integration

Having completed the integration steps, you can now test the setup using the cURL testing utility. Assuming you are running API Gateway on a machine called `apigateway` on the default port of `8080`, you can send a POST request to the newly created policy on API Gateway using HTTP basic authentication with the following command:

```
> curl --user weblogic:weblogic --data "test=data" http://apiserver:8080/oam
User 'weblogic' was authenticated and authorized successfully!
```

You can see that the success message has been returned by API Gateway meaning that the `weblogic` user has been successfully authenticated and authorized by OAM to access the resource. A quick look at the API Gateway's trace output shows that the `weblogic` user has been authenticated and authorized to access the `//oam.example.com/oam` resource.

```
DEBUG 11/Dec/2012:13:07:44.090 [0794] run circuit "OAM 11g Authentication and
Authorization"...
DEBUG 11/Dec/2012:13:07:44.094 [0794] run filter [HTTP Basic via OAM 11g R2
Repository] {
DEBUG 11/Dec/2012:13:07:44.094 [0794]  Check user name via Oracle Access Manager
DEBUG 11/Dec/2012:13:07:44.338 [0794]  Creating ResourceRequest with resType:
'http', resName: '//oam.example.com/oam, operation: 'POST'.
DEBUG 11/Dec/2012:13:07:44.339 [0794]  Successfully created ResourceRequest
DEBUG 11/Dec/2012:13:07:44.543 [0794]  Login succeeded to OAM for user weblogic
DEBUG 11/Dec/2012:13:07:44.544 [0794] } = 1, filter [HTTP Basic via OAM 11g R2
Repository]
DEBUG 11/Dec/2012:13:07:44.544 [0794] Filter [HTTP Basic via OAM 11g R2 Repository]
completes in 450 milliseconds.
DEBUG 11/Dec/2012:13:07:44.545 [0794] run filter [Authorization via OAM 11gR2] {
DEBUG 11/Dec/2012:13:07:44.545 [0794]  Creating ResourceRequest with resType:
'http', resName: '//oam.example.com/oam, operation: 'POST'.
DEBUG 11/Dec/2012:13:07:44.545 [0794]  Successfully created ResourceRequest
DEBUG 11/Dec/2012:13:07:44.545 [0794]  Authz for resource:
oracle.security.am.asdk.ResourceRequest@33aa7b
DEBUG 11/Dec/2012:13:07:44.638 [0794]  User
'uid=weblogic,ou=people,ou=myrealm,dc=idm_domain' is authorized for resource:
//oam.example.com/oam
DEBUG 11/Dec/2012:13:07:44.638 [0794] } = 1, filter [Authorization via OAM 11g R2]
DEBUG 11/Dec/2012:13:07:44.639 [0794] Filter [Authorization via OAM 11gR2] completes
in 94 milliseconds.
DEBUG 11/Dec/2012:13:07:44.639 [0794] run filter [Set Success Message] {
DEBUG 11/Dec/2012:13:07:44.639 [0794]  The content type of the converted message is
```

```
text/plain
DEBUG 11/Dec/2012:13:07:44.640 [0794]  handle type text/plain with factory class
com.vordel.mime.Body$1
DEBUG 11/Dec/2012:13:07:44.640 [0794]  Added converted message is added to the
whiteboard
DEBUG 11/Dec/2012:13:07:44.640 [0794] } = 1, filter [Set Success Message]
DEBUG 11/Dec/2012:13:07:44.640 [0794] Filter [Set Success Message] completes in 1
milliseconds.
DEBUG 11/Dec/2012:13:07:44.641 [0794] ..."OAM 11g Authentication and Authorization"
complete.
```

Now, check what happens when authenticating with a user that has not been configured in OAM:

```
> curl --user admin:changeme --data "test=data" http://apiserver:8080/oam
Access Denied!
```

If you look at the API Gateway's trace output again, you can see that the filter has blocked the authorization request:

```
DEBUG   11/Dec/2012:14:35:42.331 [0ad4] run circuit "OAM 11g Authentication and
Authorization"...
DEBUG   11/Dec/2012:14:35:42.331 [0ad4] run filter [HTTP Basic via OAM 11g R2
Repository] {
DEBUG   11/Dec/2012:14:35:42.331 [0ad4] Check user name via Oracle Access Manager
DEBUG   11/Dec/2012:14:35:42.335 [0ad4] Creating ResourceRequest with resType:
'http',
resName: '//Tyson3-pc.vordel.com/oam, operation: 'POST'.
DEBUG   11/Dec/2012:14:35:42.336 [0ad4] Successfully created ResourceRequest
ERROR   11/Dec/2012:14:35:42.486 [0ad4] Login failed to Oracle Access Manager for
user admin
ERROR   11/Dec/2012:14:35:42.487 [0ad4] java exception:
com.vordel.circuit.authn.VordelAuthNException: Login failed

...

DEBUG   11/Dec/2012:14:35:42.533 [0ad4] } = 0, filter [HTTP Basic via OAM 11g R2
Repository]
DEBUG   11/Dec/2012:14:35:42.534 [0ad4] Filter [HTTP Basic via OAM 11g R2
Repository] completes in 203 milliseconds.
ERROR   11/Dec/2012:14:35:42.534 [0ad4] The message [Id-71222fbb50c744be03d40000]
logged Failure
at 12.11.2012 14:35:42,534 with log description: HTTP basic authentication failed
DEBUG   11/Dec/2012:14:35:42.535 [0ad4] run filter [Set Blocked Message] {
DEBUG   11/Dec/2012:14:35:42.535 [0ad4] The content type of the converted message is
text/plain
DEBUG   11/Dec/2012:14:35:42.535 [0ad4] handle type text/plain with factoryclass
com.vordel.mime.Body$1
DEBUG   11/Dec/2012:14:35:42.536 [0ad4] Added converted message is added to the
whiteboard
DEBUG   11/Dec/2012:14:35:42.536 [0ad4] } = 1, filter [Set Blocked Message]
```

```
DEBUG   11/Dec/2012:14:35:42.536 [0ad4] Filter [Set Blocked Message] completes in 1
milliseconds.
DEBUG   11/Dec/2012:14:35:42.536 [0ad4] ..."OAM 11g Authentication and
Authorization" complete.
```

Oracle Entitlements Server 11g and 11gR2 integration

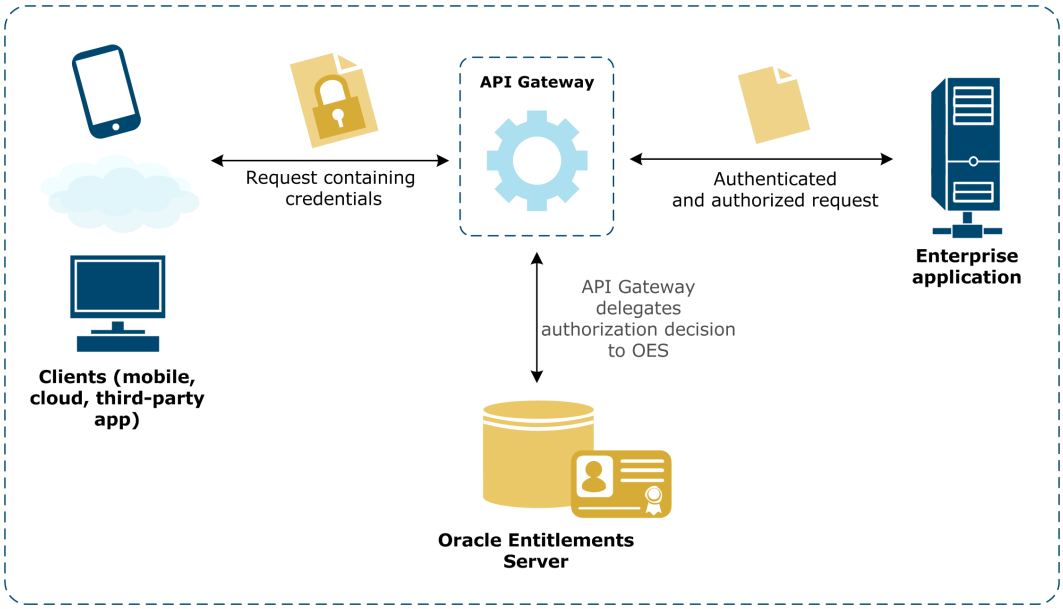
5

This section describes how to configure API Gateway to authorize an authenticated user against Oracle Entitlements Server (OES) 11g and 11gR2. This will be demonstrated by the following:

- API Gateway will authenticate a user against its local user repository
- API Gateway will then delegate the authorization decision for the specified resource to OES

The OES 11g Authorization filter is used to delegate the authorization decision to OES. This filter assumes that an authentication filter has been configured prior to it. Therefore, by the time the authorization filter executes, the `authentication.subject.id` message attribute is populated and its value is used as the subject in the authorization request to OES.

The following diagram shows the sequence of events that occurs when a client sends a message to API Gateway. The request sender is authenticated by API Gateway and is then authorized against Oracle Entitlements Server. If the user is permitted to access the requested resource, the request is routed to the Enterprise Application. Otherwise an appropriate fault message is returned to the client.



Prerequisites

API Gateway

You must have installed API Gateway version 7.6.2 or higher and have received a valid license from Axway.

This integration is also valid for the API Gateway Appliance (physical or virtual) version 7.6.2 or higher.

OES user

You must create an OES user called `weblogic`. Refer to the OES documentation for instructions on how to add a user.

API Gateway local user store

You must have added the `weblogic` user to the API Gateway local user store. The policy you will set up later requires an authenticated user's request to be authorized against OES. By adding the `weblogic` user to the local user store, the client can authenticate as this user. The user name will then be stored in the `authentication.subject.id` message attribute, which is then passed to the OES 11g Authorization filter and subsequently on to OES to make the authorization decision.

See the *API Gateway Administrator Guide* for more information on adding users.

OES client

You must have installed the OES client (security module) on the machine running API Gateway. The OES client has its own installer, which is available from www.oracle.com.

Note In the following integration steps, this version of the OES client was used: Oracle Entitlements Server Security Module 11g - 11.1.2.0.0.

The OES client installer requires that a JRE is available on the target machine. In the absence of a preferred JVM on the target machine, API Gateway ships with a JRE that can be used. On UNIX, the JRE is located in `INSTALL_DIR/apigateway/platform/jre`.

Start the OES client installer from the command line and pass the JRE location using the `jreLoc` argument as follows:

UNIX/Linux

```
./runInstaller -jreLoc INSTALL_DIR/apigateway/platform/jre
```

OES 11g or 11gR2

You must have installed, configured, and started OES 11g or 11gR2. For example, you can start it using the following commands on a UNIX-based system:

```
# cd ~/Middleware/user_projects/domains/oes_domain
# ./startWebLogic.sh
```

Note This command assumes a WebLogic domain of `oes_domain` has already been configured.

cURL testing utility

To test the integration steps, the cURL testing utility is used to POST requests to API Gateway. It is available from the following URL:

http://curl.haxx.se/download.html

Alternatively, you can use any client capable of sending HTTP POST requests with HTTP basic authentication.

Configure Oracle Entitlements Server

This section describes how to configure Oracle Entitlements Server to work with API Gateway.

Create a resource and authorization policy in OES

The following steps describe how to create a resource and an authorization policy for that resource in Oracle Entitlements Server. For more detailed instructions about each of these steps, refer to the OES documentation.

You can use the OES Authorization Policy Manager web-based administration interface to configure the resource. The web interface is available at the following URL, where `OES_HOST` refers to the IP or host name of the machine on which OES is running:

```
http://OES_HOST:7001/apm
```

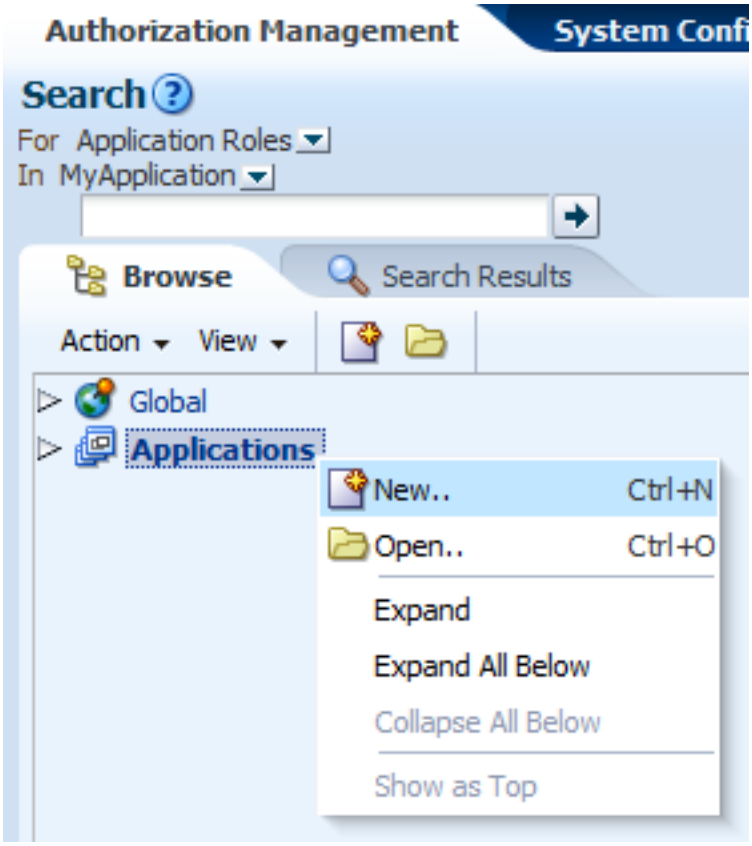
Log in using your WebLogic credentials and complete the following steps:

- *Step 1 - Create the application* on page 60
- *Step 2 - Create the security module* on page 60
- *Step 3 - Create the resource type* on page 62

Step 1 - Create the application

To add a new application in OES, perform these steps:

1. Open the **Authorization Management** tab.
2. Right-click on the **Applications** node in the tree and select **New**:



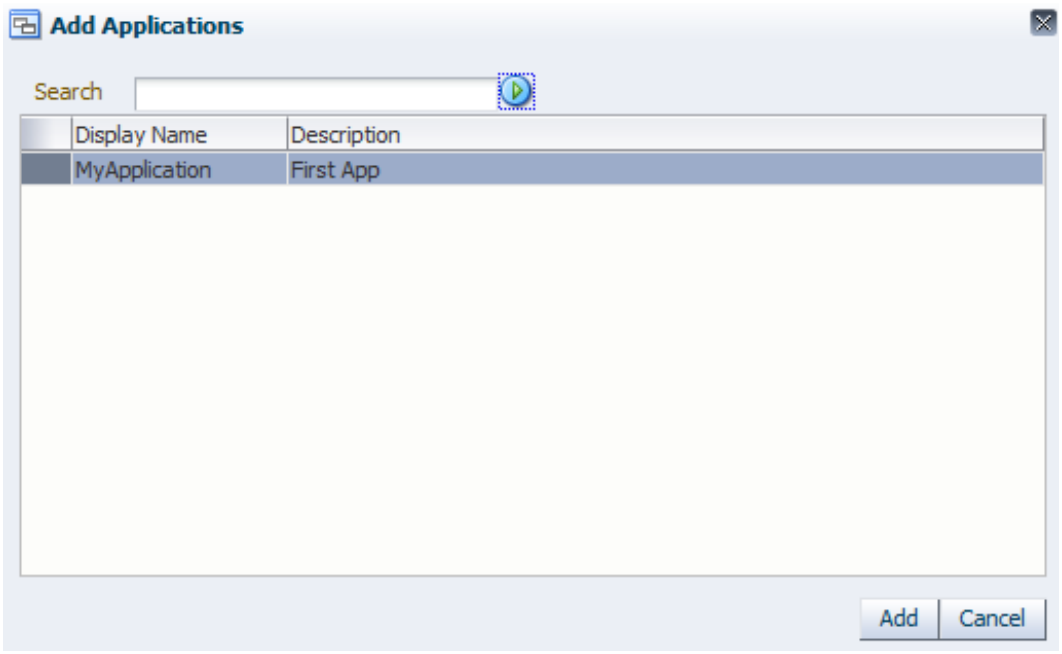
3. Enter a name and a description for the new application, for example, `MyApplication` and `First App`.
4. Click **Save** at the top right-hand corner of the page.

Step 2 - Create the security module

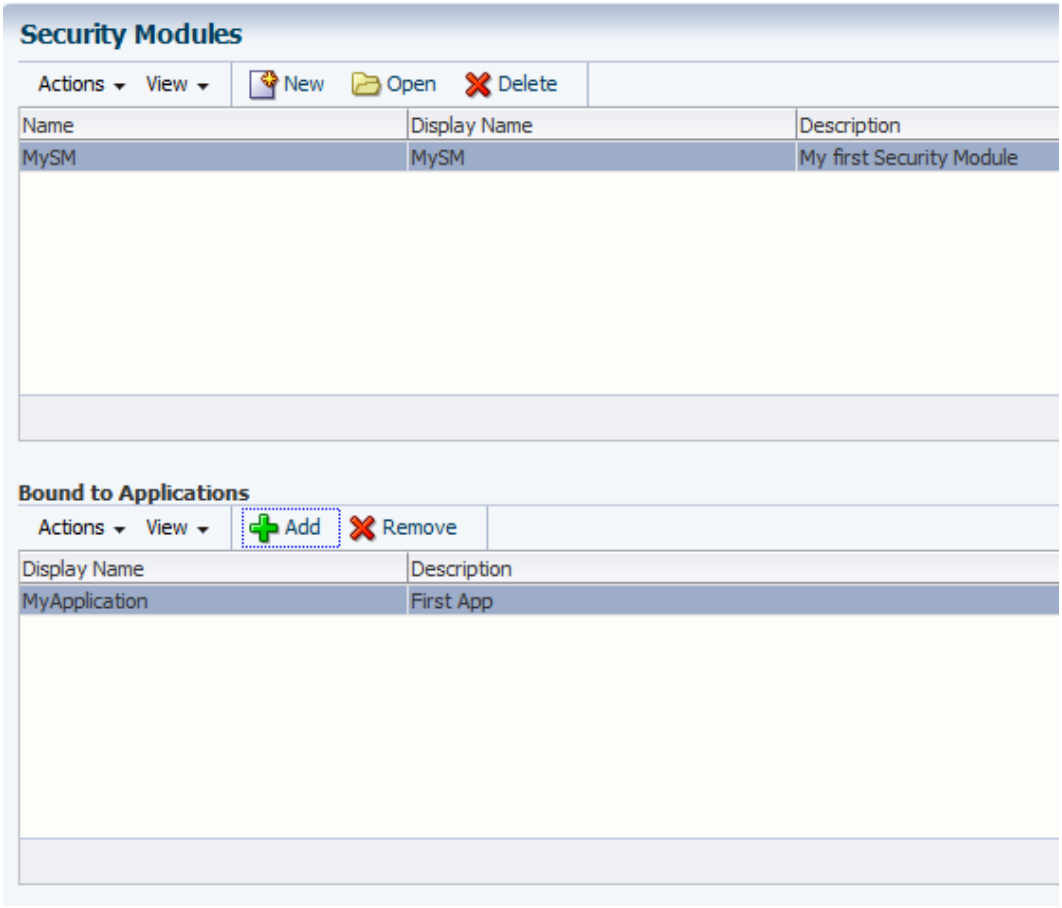
The next step is to create a new security module:

1. Open the **System Configuration** tab.
2. Double-click the **Security Modules** node in the tree.
3. Click **New** at the top of the **Security Modules** table.
4. Enter `MySM` as the name of the new security module.

5. Click **Add** at the top of the **Bound to Applications** table.
6. Leave the **Search** field blank and click the search button to the right of the field.
7. Select `MyApplication` from the search results and click **Add**.



8. The `MyApplication` should now be displayed in the **Bound to Applications** table as follows:

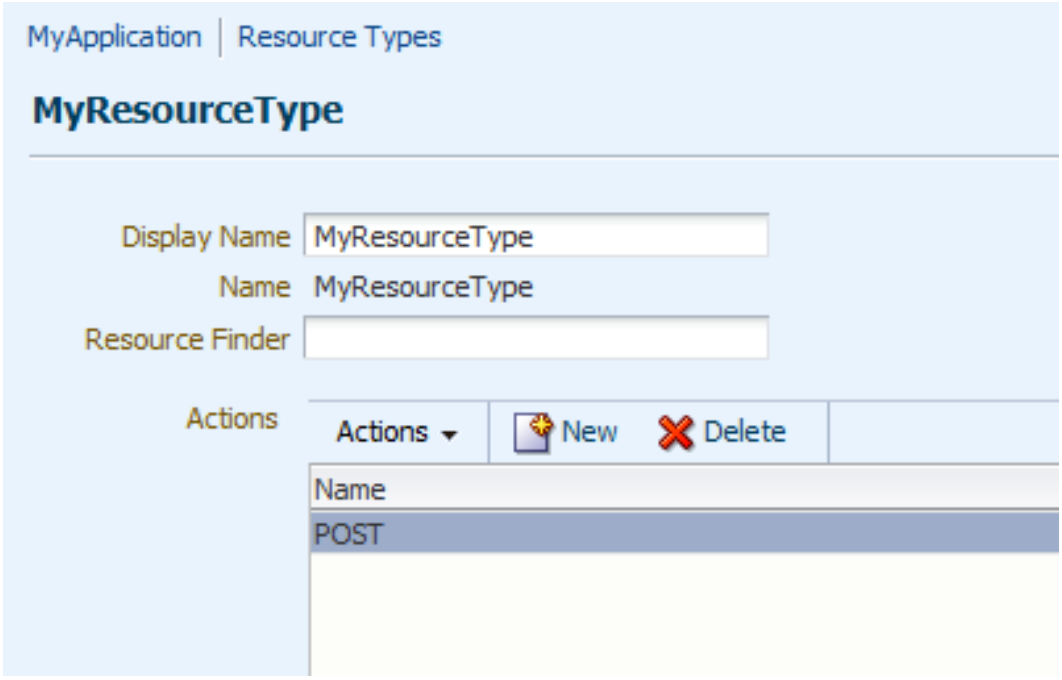


Step 3 - Create the resource type

You will now configure the resource type for the resource that users will be authorized for:

1. Open the **Authorization Management** tab again.
2. Expand the **Applications** node and then expand the newly created **MyApplication** node.
3. Double-click the **Resource Types** node.
4. Click **New** above the table showing the existing **Resource Types**.
5. Enter `MyResourceType` in the **Name** field.
6. Add an action for this resource by clicking **New** above the **Actions** table. Add `POST` as an action.
7. Click **Save** to save the new **Resource Type**, which should now be displayed as in the

following screenshot:

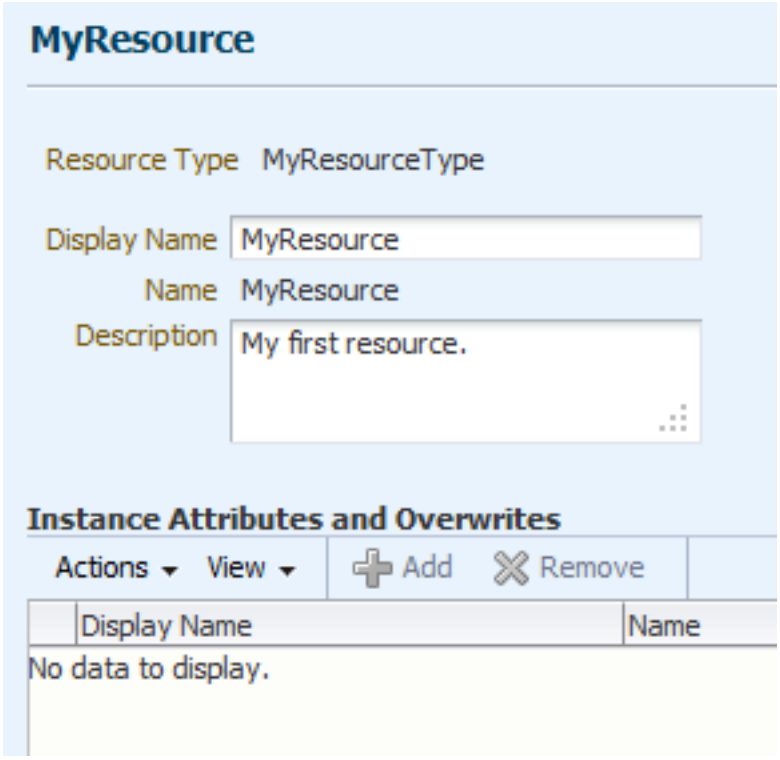


Step 4 - Create the resource

The next step is to add the **Resource**:

1. Expand the **Default Policy Domain** and then the **Resource Catalog** nodes.
2. Double-click the **Resources** node.
3. Click **New** above the table listing all existing **Resources**.
4. Select `MyResourceType` as the **Resource Type**.
5. Enter `MyResource` as the **Resource name**.
6. Click **Save** to save the **Resource**, which should now be displayed as in the following

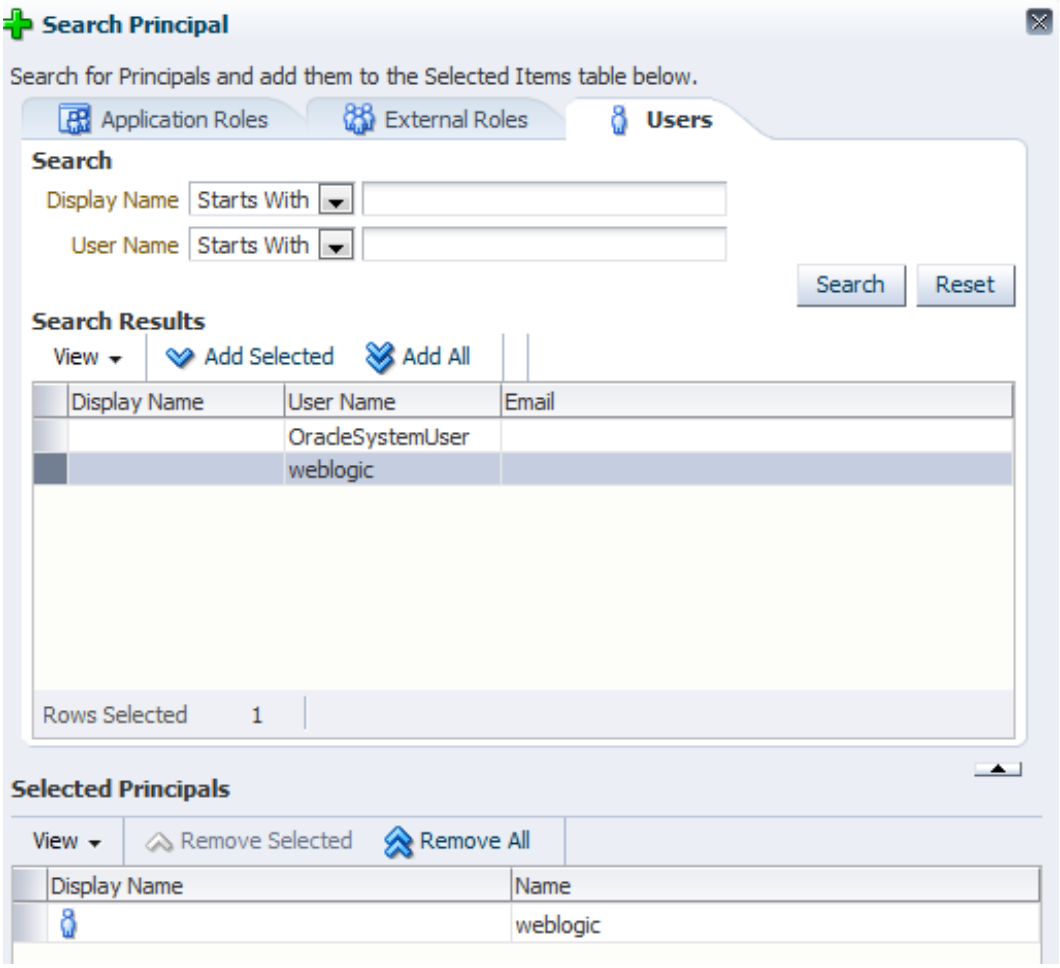
screenshot:



Step 5 - Configure the authorization policy

Now you must create the authorization policy that will determine whether to permit or deny access to the resource:

1. Double-click the **Authorization Policies** node beneath the **Default Policy Domain** node in the tree.
2. Click **New** above the table showing the existing **Authorization Policies**.
3. Enter `MyPolicy` as the name of the new policy.
4. Chose to "Permit" access to the resource target using the corresponding **Effect** check box.
5. Configure **Principals** (users, roles, or both) that can access the resource by clicking the "plus" button to the right of the **Principals** table.
6. Select the **Users** tab in the **Search Principal** window:



7. Select the default `weblogic` user from the table and click **Add Selected** to add the `weblogic` user to the **Selected Principals** table, as shown in the previous screenshot.

   Note The *Configure Oracle Entitlements Server* on page 59 required a `weblogic` user to be available in OES.

8. Click **Add Principals** at the bottom of the window.
9. Next, you must specify a resource target that this policy will act on. Click the "plus" button to the right of the **Targets** table.
10. Click the **Resources** tab on the **Search Targets** window.
11. Select `MyResourceType` in the **Resource Type** drop-down and click **Search**.
12. Select `MyResource` from the table and click **Add Selected** to add the resource to the **Selected Targets** table.
13. Click **Add Targets** at the bottom of the page.

Set up the OES client

The OES client distributes policies to individual security modules that protect applications and services. Policy data is distributed in a controlled manner or in a non-controlled manner. The distribution mode is defined in the `jps-config.xml` configuration file for each security module. The specified distribution mode applies to all application policy objects bound to that security module. Consult with the OES administrator to find out the distribution mode. For the purposes of this section, the *controlled* distribution mode is used.

Controlled mode

Complete the following steps to configure the OES client in controlled mode:

1. Open a command prompt and change directory to your OES client installation directory (this is referred to as `OES_CLIENT_HOME` throughout the remainder of this section).
2. Set the `JAVA_HOME` environment variable. For example:

   UNIX/Linux

   ```
   > export JAVA_HOME=/home/oesuser/Oracle/Middleware/jdk160_29
   ```

3. Edit the following file:

   ```
   OES_CLIENT_HOME/oessm/SMConfigTool/smconfig.java.controlled.prp
   ```

4. Ensure that the following values are set:

Parameter	Description
`oracle.security.jps.runtime.pd.client.policyDistributionMode`	Accept the default value `controlled-push` as the distribution mode.
`oracle.security.jps.runtime.pd.client.RegistrationServerHost`	Enter the address of the Oracle Entitlements Server Administration Server.

Parameter	Description
`oracle.security.jps.runtime.pd.client.RegistrationServerPort`	Enter the SSL port number of the Oracle Entitlements Server Administration Server. You can find the SSL port number from the WebLogic Administration console. By default, `7002` is used.

5. On UNIX-based systems, run the `config.sh` script located in the `OES_CLIENT_HOME/oessm/bin` directory.

   UNIX/Linux

   ```
   > ./config.sh –smConfigId MySM -prpFileName OES_CLIENT_
   HOME/oessm/SMConfigTool/smconfig.java.controlled.prp
   ```

6. When prompted, specify the following:
   - OES user name (Administration Server's user name)
   - OES password (Administration Server's password)
   - New key store password for enrollment

   The following shows some sample output:

   ```
   > ./config.sh -smConfigId MySM -prpFileName
   /home/oesuser/Oracle/Middleware/oes_
   client/oessm/SMConfigTool/smconfig.java.controlled.prp
   Configuring for Controlled Policy Distribution Mode
   Security Module configuration is created at:
   /home/oesuser/Oracle/Middleware/oes_client/oes_sm_instances/MySM
   Enter password for key stores: ******
   Enter password for key stores again: ******
   Passwords are saved in credential store.
   Keystores are initialized successfully.
   Please enter a value for OES Admin Server User name:weblogic
   Please enter a value for OES Admin Server Password:
   Enrollment is proceeded successfully.
   ```

7. Ensure that the security module has been configured correctly by checking that the `OES_CLIENT_HOME/oes_sm_instances/MySM` directory has been created.
8. Depending on your OES client setup, the registration process might not have generated a `cwallet.sso` file in your `OES_CLIENT_HOME/oes_sm_instances/MySM/config`

directory. If there is no `cwallet.sso` file present in the `config` directory, you can copy the one generated in the `config/enroll` directory to the `config` directory using the following commands:

```
# cd oes_sm_instances/MySM/config
[oesuser@oeseval config]$ ls
enroll  jps-config.xml  system-jazn-data.xml
# cp enroll/cwallet.sso ./
# ls
cwallet.sso  enroll  jps-config.xml  system-jazn-data.xml
```

Non-controlled mode

Alternatively, you can use the non-controlled or controlled pull distribution mode. Consult the Oracle documentation for configuring these modes.

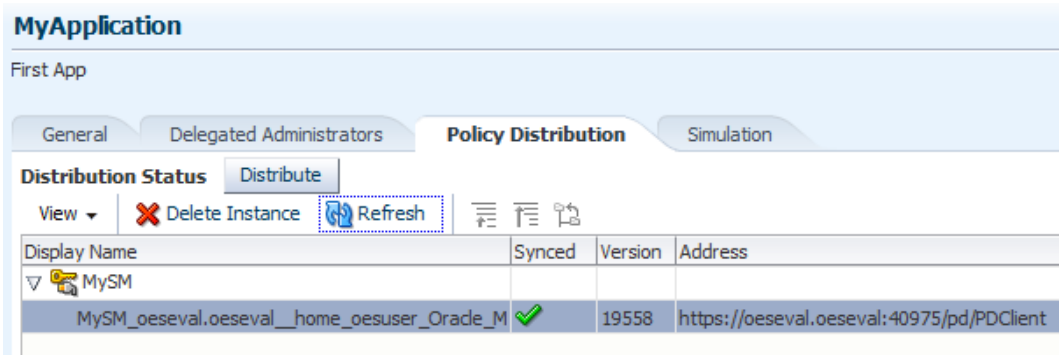
Distribute the OES policy

When the OES client has registered with OES, you can distribute the policy for that application so that clients making authorization requests for this resource will be subject to the policy enforcement rules.

Follow these steps in the OES Authorization Policy Manager web-based administration interface:

1. Double-click the **MyApplication** node in the tree on the **Authorization Management** tab.
2. Open the **Policy Distribution** tab for the application configuration.
3. Expand the `MySM` entry in the table.

   You should see an entry representing the recently registered OES client instance, as shown in the following screenshot:



4. Select the **MySM** application and click **Distribute** to push the authorization policy configured for this application.
5. You might need to click **Refresh** to update the **Synced** status. You should see a green tick to indicate a successful distribution.

Note You might have to restart WebLogic for your newly registered security module to be displayed in the list on the **Policy Distribution** tab.

Configure API Gateway

This section describes how to configure API Gateway to work with Oracle Entitlements Server.

Complete the following tasks:

- *Modify the API Gateway classpath* on page 69
- *Start API Gateway* on page 70
- *Configure API Gateway to delegate authorization to OES* on page 70

Modify the API Gateway classpath

API Gateway's classpath must be extended to include the OES client JAR. To achieve this, create a `jvm.xml` file at the following location:

```
INSTALL_DIR/apigateway/conf/jvm.xml
```

Edit this `jvm.xml` so that its contents are as follows, providing values for `OES_CLIENT_HOME` and `SM_NAME` that are based on the location where the OES client was installed and the SM name used when enrolling the OES client (`MySM`):

```
<ConfigurationFragment>
<!-- Change these ENV VARS to match the location where the OEM Client has
been installed and configured -->
<Environment name="OES_CLIENT_HOME" value="/home/oes/Oracle/Middleware/oes_client"
/>
<Environment name="SM_NAME" value="MySM" />
<Environment name="INSTANCE_HOME"
value="$OES_CLIENT_HOME/oes_sm_instances/$SM_NAME" />

<!-- Add OES Client JAR to the classpath -->
<ClassPath name="$OES_CLIENT_HOME/modules/oracle.oes.sm_11.1.1/oes-client.jar" />

<!-- Add OES JARs to the classpath →
<ClassPath name=”[PATH_TO_OES_JARS]/api.jar”/>
<ClassPath name=”[PATH_TO_OES_JARS]/asi_classes.jar”/>

<VMArg name="-Doracle.security.jps.config=$INSTANCE_HOME/config/jps-config.
xml"/>
<!-- Optional argument to add enhanced logging (via log4j) for the OES Client -->
<VMArg name=”-Djava.util.logging.config.file=$INSTANCE_HOME/logging.properties”/>
</ConfigurationFragment>
```

The following is an example `jvm.xml` file for Windows:

```
<ConfigurationFragment>
  <!-- Environment variables -->
  <!-- change these to match the location where the OEM Client has been installed
and configured -->
<Environment name="OES_CLIENT_HOME" value="C:\Oracle\product\11.1.1\as_1" />
<Environment name="SM_NAME" value="MySSM" />
<Environment name="INSTANCE_HOME" value="$OES_CLIENT_HOME/oes_sm_instances/$SM_NAME"
/>
<!-- Add OES Client to classpath -->
<ClassPath name="$OES_CLIENT_HOME/modules/oracle.oes.sm_11.1.1/oes-client.jar" />
<VMArg name="-Doracle.security.jps.config=$INSTANCE_HOME/config/jps-config.xml"/>
</ConfigurationFragment>
```

Start API Gateway

Start API Gateway so that it runs with the OES client classpath and the associated environment settings. Refer to the *API Gateway Installation Guide* for instructions on how to start API Gateway.

Command example:

```
> startinstance -n "APIGateway1" -g "Group1"
```

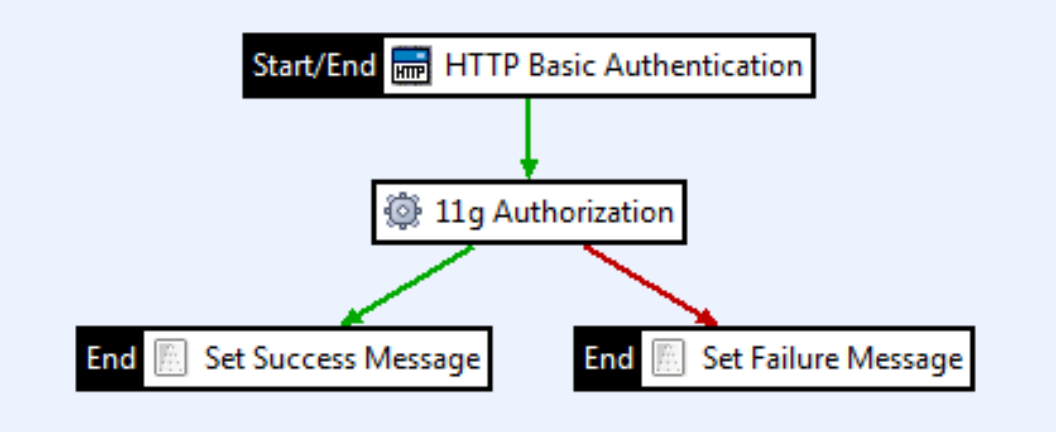
Configure API Gateway to delegate authorization to OES

This section explains how to configure API Gateway to delegate authorization decisions to Oracle Entitlements Server.

The following steps are required:

- *Step 1 - Configure the authentication filter* on page 71
- *Step 2 - Configure the OES 11g authorization filter* on page 72
- *Step 3 - Add the success message filter* on page 73
- *Step 4 - Add the failure message filter* on page 74
- *Step 5 - Add a relative path for the OES authorization policy* on page 74
- *Step 6 - Deploy the policy* on page 75

The resulting policy created in API Gateway will appear as follows:



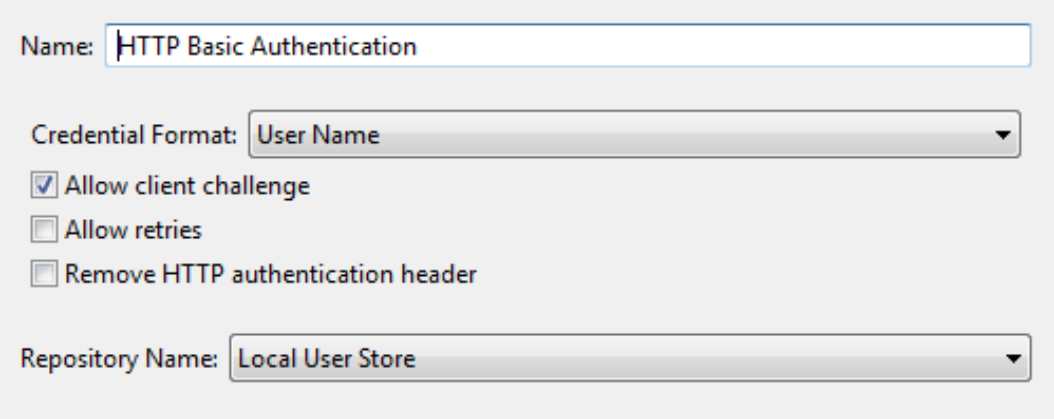
Step 1 - Configure the authentication filter

In this example, it is assumed that the user profile to be authorized through OES is also stored in the local user store of API Gateway. API Gateway can also delegate authentication decisions to other systems (for example, LDAP directories, databases, and other third-party identity management systems, including CA SiteMinder, Oracle Access Manager, RSA Access Manager, and so on). For simplicity, API Gateway's local user store is used in this example.

Configure the authentication filter as follows:

1. In Policy Studio, create a new policy called `OES Authorization`.
2. Drag a **HTTP Basic** filter from the **Authentication** category in the palette onto the canvas and configure it as follows:
   - **Name**: Enter `HTTP Basic Authentication` in the field provided.
   - **Credential Format**: Select `User Name` from the drop-down list.
   - **Allow Client Challenge**: Select the **Allow client challenge** check box.
   - **Repository Name**: Select `Local User Store` from the drop-down list.

   The completed configuration for the filter appears as follows:



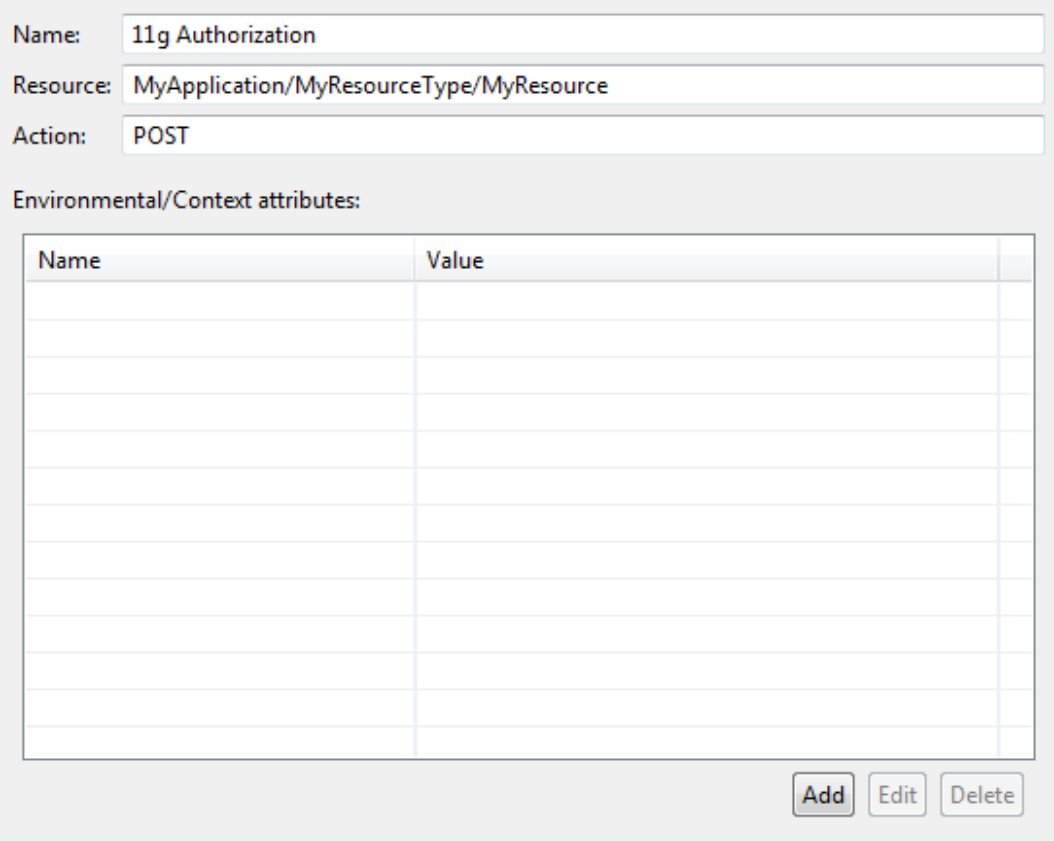
3. Click **OK**.
4. To set this authentication filter to be the starting filter of the policy, right-click the filter in the canvas and select **Set as Start**.

Step 2 - Configure the OES 11g authorization filter

Configure the OES 11g authorization filter as follows:

1. From the **Oracle Entitlements Server** category in the palette on the right of Policy Studio, drag the **11g Authorization** filter onto the canvas, and configure it as follows:
   - **Resource**: Enter a formatted string representing the resource that you created in OES and for which API Gateway will ask OES for authorization decisions. The resource you created earlier in OES can be represented with the string `MyApplication/MyResourceType/MyResource`.
   - **Action**: The rules created in the OES policy can permit/deny access to a resource based on the requested action, for example, POST, GET, DELETE, and so on. In this example, you will be POST-ing the request to the resource, so you must enter `POST` in the **Action** field. Remember, you configured POST access to the this resource earlier when configuring the OES policy.
   - You can optionally configure environment context attributes. However, for the purposes of this integration example it is not necessary to configure this section.

   The completed configuration appears as follows:



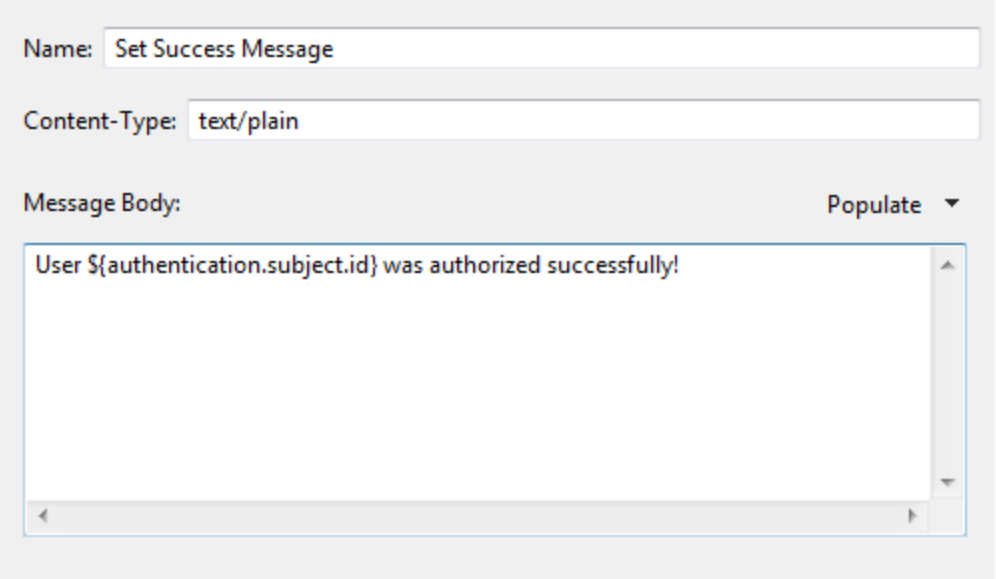
2. Click **OK**.
3. Set the success path from the **HTTP Basic Authentication** filter to point at the newly created OES 11g authorization filter.

Step 3 - Add the success message filter

Display a success message after successfully authorizing the user by adding a **Set Message** filter.

1. Drag a **Set Message** filter from the **Conversion** category onto the canvas and configure it as follows:
   - **Name**: Enter `Set Success Message` in the text field.
   - **Content-type**: Enter `text/plain` as the content-type of the message to return to the client.
   - **Message Body**: Enter the following message to return to the client: `User '${authentication.subject.id}' was authorized successfully!`

   The configuration for the **Set Success Message** filter should now look like this:



2. Click **OK**.
3. Set the success path of the 11g authorization filter to the **Set Success Message** filter.

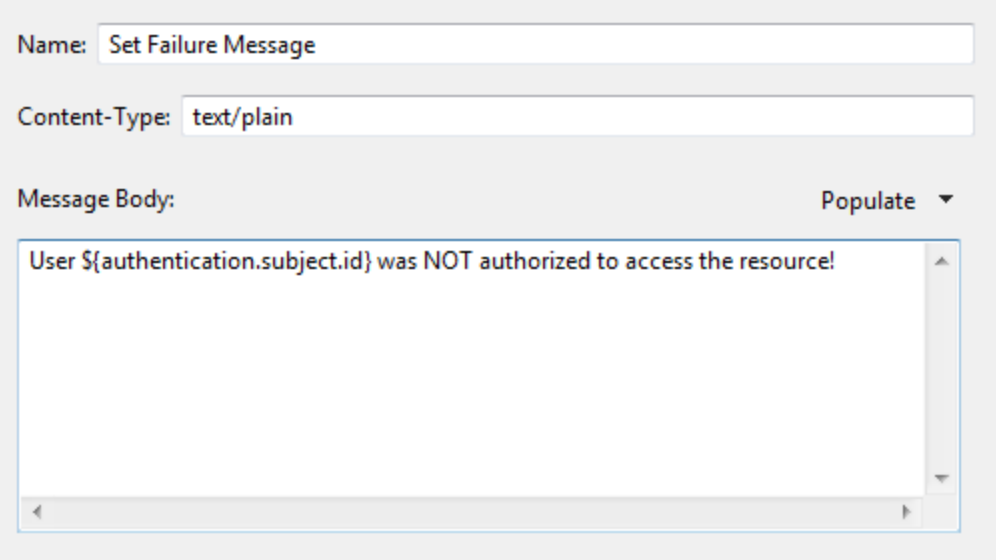
Step 4 - Add the failure message filter

If OES returns false for the authorization request you should return an appropriate error message to the client.

Display a failure message after an unsuccessful authorization event by adding another **Set Message** filter:

1. Drag a **Set Message** filter from the **Conversion** category onto the canvas, and configure it as follows:
   - **Name**: Enter `Set Failure Message` in the text field.
   - **Content-type**: Enter `text/plain` as the content-type of the message to return to the client.
   - **Message Body**: Enter the following message to return the client: `The user '${authentication.subject.id}' was NOT authorized to access the resource!`

   The configuration for the **Set Failure Message** filter should now look like this:



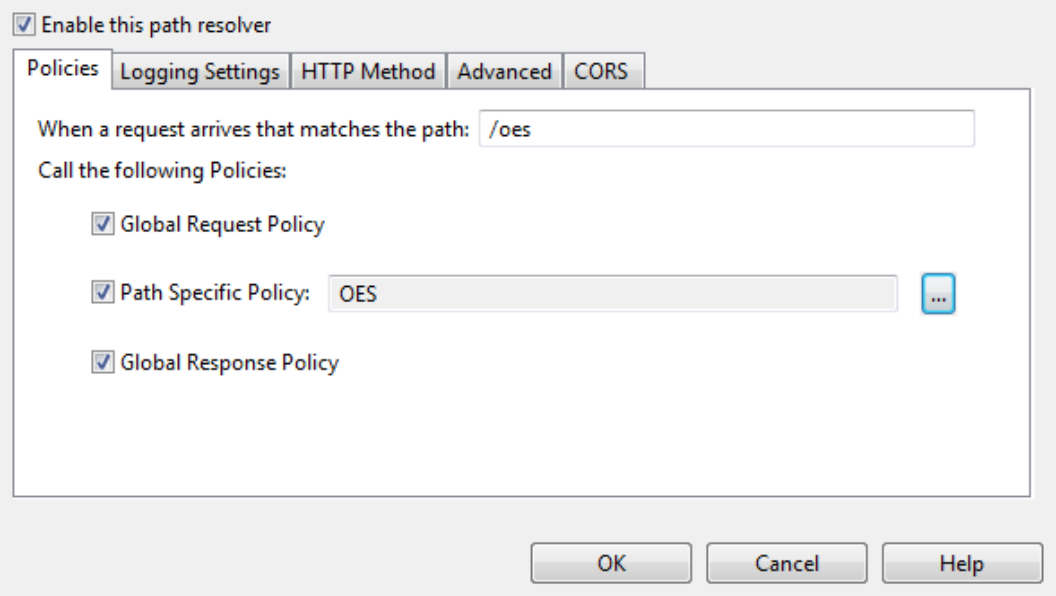
2. Click **OK**.
3. Set the failure path of the 11g authorization filter to the **Set Failure Message** filter.

Step 5 - Add a relative path for the OES authorization policy

In order for API Gateway to invoke this policy for certain requests you must create a relative path and map it to the policy. All requests received on this path are automatically forwarded to the policy for processing.

To add a relative path for this policy click **Add Relative Path** in the toolbar beneath the policy canvas.

Enter the path on which to receive requests for this policy in the field provided in the Resolve path to Policies dialog:



For example, if you enter a relative path of `/oes`, you can see that this path is automatically mapped to the `OES Authorization` policy created earlier in this section.

Step 6 - Deploy the policy

To push the configuration changes to the live API Gateway instance you must deploy the new policy. You can do this by pressing the **F6** button.

Test the integration

Having completed the integration steps, you can now test the setup using the cURL testing utility.

Note You must have added a `weblogic` user to the API Gateway local user store as outlined in *Prerequisites* on page 58.

Assuming you are running API Gateway on a machine called `apigateway` on the default port of 8080, you can send a POST request to the authorization policy on API Gateway using HTTP basic authentication with the following command:

```
>curl  --user weblogic:weblogic --data "test=data" http://apigateway:8080/oes
User 'weblogic' was authorized successfully!
```

You can see that the success message has been returned by API Gateway meaning that the `weblogic` user has been successfully authorized by OES to access the resource. A quick look at the API Gateway's trace output shows the OES 11g Authorization filter making the successful authorization request for the `MyApplication/MyResourceType/MyResource` resource:

```
DEBUG   23/Oct/2012:15:28:45.183 [42687940] run circuit "OES Authorization"...
DEBUG   23/Oct/2012:15:28:45.183 [42687940] run filter [HTTP Basic Authentication] {
DEBUG   23/Oct/2012:15:28:45.183 [42687940]    VordelRepository.checkCredentials:
username=weblogic
DEBUG   23/Oct/2012:15:28:45.183 [42687940]    Attempt to map incoming format Username
to repository authZ format Username
DEBUG   23/Oct/2012:15:28:45.183 [42687940] } = 1, filter [HTTP Basic
Authentication]
DEBUG   23/Oct/2012:15:28:45.183 [42687940] Filter [HTTP Basic Authentication]
completes in 0 milliseconds.
DEBUG   23/Oct/2012:15:28:45.183 [42687940] run filter [11g Authorization] {
DEBUG   23/Oct/2012:15:28:45.183 [42687940]    creating subject from 'weblogic'
DEBUG   23/Oct/2012:15:28:45.183 [42687940]    checking 'POST' to resource:
MyApplication/MyResourceType/MyResource
DEBUG   23/Oct/2012:15:28:45.185 [42687940]    Request: {weblogic, POST,
MyApplication/MyResourceType/MyResource}
Result: true
DEBUG   23/Oct/2012:15:28:45.185 [42687940]    result from OES: true
DEBUG   23/Oct/2012:15:28:45.185 [42687940]    no obligations in response
DEBUG   23/Oct/2012:15:28:45.185 [42687940] } = 1, filter [11g Authorization]
DEBUG   23/Oct/2012:15:28:45.185 [42687940] Filter [11g Authorization] completes in
2 milliseconds.
```

Now let's see what happens when you authenticate with a user that is present in the API Gateway's local user store, but that has not been configured with authorization rights in OES. To demonstrate this, you can send up a request using the credentials of the default API Gateway `admin` user:

```
>curl  --user admin:changeme --data "test=data" http://apigateway:8080/oes
User 'admin' was NOT authorized to access the resource!
```

If you take a quick look at the API Gateway's trace output again, you can see that the 11g Authorization filter has blocked the authorization request:

```
DEBUG   23/Oct/2012:15:29:22.678 [41f67940] run circuit "OES Authorization"...
DEBUG   23/Oct/2012:15:29:22.678 [41f67940] run filter [HTTP Basic Authentication] {
DEBUG   23/Oct/2012:15:29:22.678 [41f67940]    VordelRepository.checkCredentials:
username=admin
DEBUG   23/Oct/2012:15:29:22.678 [41f67940]    Attempt to map incoming format Username
to repository authZ format Username
DEBUG   23/Oct/2012:15:29:22.678 [41f67940] } = 1, filter [HTTP Basic
Authentication]
DEBUG   23/Oct/2012:15:29:22.678 [41f67940] Filter [HTTP Basic Authentication]
completes in 0 milliseconds.
DEBUG   23/Oct/2012:15:29:22.678 [41f67940] run filter [11g Authorization] {
DEBUG   23/Oct/2012:15:29:22.678 [41f67940]    creating subject from 'admin'
```

```
DEBUG   23/Oct/2012:15:29:22.678 [41f67940]    checking 'POST' to resource:
MyApplication/MyResourceType/MyResource
DEBUG   23/Oct/2012:15:29:22.679 [41f67940]    Request: {admin, POST,
MyApplication/MyResourceType/MyResource}
Result: false
DEBUG   23/Oct/2012:15:29:22.679 [41f67940]    result from OES: false
ERROR   23/Oct/2012:15:29:22.679 [41f67940]    Failed to authZ to OES
DEBUG   23/Oct/2012:15:29:22.680 [41f67940] } = 0, filter [11g Authorization]
DEBUG   23/Oct/2012:15:29:22.680 [41f67940] Filter [11g Authorization] completes in
2 milliseconds.
```