Part 9 of this multi-part series describes how you can customize WebSphere DataPower's native OAuth scope and identity support.

View more content in this series

Introduction

Figure 1 shows native WebSphere® DataPower® support for OAuth with DataPower playing both the authorization and resource enforcement endpoint roles. You may recall from previous articles in this series that DataPower may act in either role or in conjunction with other services such as IBM's Tivoli® Federated Identity Management.

Figure 1. An OAuth exchange using an authorization code grant type

For DataPower to act as an authorization server, it has to provide at a minimum the following functions:
• Issue and verify an authorization code.
• Issue and verify an access token.
• Issue and verify a refresh token (new in 6.0.0 release).

Within DataPower processing, both the authorization code and access token are self-contained and cryptographically protected. These tokens are transmitted as a part of transactional data such as HTTP headers, query parameters, or within post-data.

DataPower allows you to customize the processing of these tokens through extension points. These extension points allow customizations such as:

• Check and validate the scope.
• Set the default scope if none is provided (6.0 feature).
• Determine validate the resource owner.
• Customize the authorization/consent form
• Additional OAuth processing (covered in Part 10)

During the customization steps for the following, DataPower will provide information to the custom stylesheet. It will also indicate which steps of the OAuth protocol DataPower is invoking for the stylesheet. The following list will also be expanded as new features are added:

• authorization_request: This custom process is triggered during an authorization code grant type when the authorization code is requested by the client.
• access_request: This custom process is triggered when the OAuth client/application is requesting for an access_token, which can be during the supported grant type.
• resource_request: This custom process is triggered when the OAuth client/application is accessing a resource with its validated access_token.
• refresh_request: This custom process is triggered when DataPower received a refresh grant_type from the OAuth client/application.
• validate_request: This custom process is triggered when DataPower received a validation grant_type for an access_token.
• client_revoke_request: This custom process is triggered when DataPower received a revoke request by the OAuth client/application to revoke the permission it received.
• owner_revoke_request: This custom process is triggered when DataPower received a request by the resource owner to revoke an OAuth client/application to the owner’s resource.

Checking the scope

The scope is checked in the following scenarios:

• When requesting an authorization code (for the authorization code grant type)
• When requesting an access token
• When requesting a resource using a previously provided access token

In these cases, the OAuth client indicates the requested scope. By default, DataPower uses the regular expression specified in the Scope field of the OAuth Client Profile (as seen in Figure
2) to verify what is requested. You can enable customization of this behavior by checking the **Customized Scope Check** checkbox.

**Figure 2. Enable the Custom Scope Check**

Selecting the custom scope option enables a stylesheet selection field named **Scope Customization Process**. DataPower delegates scope-authorization decisions to this stylesheet for the OAuth client. For example, in Figure 3, `dp-custom-scope.xsl` is used to determine what scope is supported by this OAuth client.

**Figure 3. Custom scope stylesheet field**

The input to the scope is an XML node set as shown in Listing 1. Note that this XML node set may be extended in the future to include additional information.

**Listing 1. Sample of custom scope processing input node**

```xml
<input>
  <scope>requestedscope another-requested-scope</scope>
  <oauth-id type="authorization_request | access_request | resource_request | …"> 
    … from AAA Extract Identity (EI) ….
  </oauth-id>
  <OAuthSupportedClient> 
    … the registered OAuth client profile ….
  </OAuthSupportedClient>
</input>
```

The "type" attribute of the `oauth-id` element describes the state or "type" of processing that DataPower is currently performing (see above for the list).
The output of the stylesheet must conform to either a success or failure format. The success format is shown in Listing 2.

Listing 2. Sample custom scope processing input node

```xml
<result>
  <scope>allowedscope1 allowedscope2</scope>
</result>
```

The failure format is shown in Listing 3.

Listing 3. Sample of failure scope processing

```xml
<result>
  <error>Scope is not allowed</error>
</result>
```

Listing 4 shows an example of a custom scope processing stylesheet. In this simple example, the `<result>` element is created and the `/input/scope` content is checked for `supportedScope`. If the test succeeds, a `<scope>` element is added. Otherwise, an `<error>` element is added. In this demonstration, the input scope of `supportedScope` is replaced with `realScope`. This is similar to the mapping capabilities of the AAA process, where credentials or resources may be "mapped" to other values.

Listing 4. Sample of custom scope processing

```xml
<xsl:template match="/">
  <result>
    <xsl:choose>
      <xsl:when test="contains(/input/scope, 'supportedScope')">
        <scope><xsl:value-of select='realScope'/></scope>
      </xsl:when>
      <xsl:otherwise>
        <error>requested scope is not allowed</error>
      </xsl:otherwise>
    </xsl:choose>
  </result>
</xsl:template>
```

Instead of using custom scope processing, you can also use AAA > Mapped Resource to archive this. The following is an example of the stylesheet on how to match a wild card “approved scope” to a list of URI requested by the client:

```xml
approved scope contains '/photo/spoon/*
requested resource by the client/application is '/photo/spoon/mutt.jpg
```

In order to map `/photo/spoon/mutt.jpg` to `/photo/spoon/*`, you can use a similar script as shown in Listing 5.

Listing 5. Sample of mapped resources processing (include the right namespace)

```xml
<xsl:stylesheet xmlns:...
  xmlns:str="http://exslt.org/strings"
```
Specifying the default scope

In the 6.0.0 firmware release, DataPower allows for the specification of a default value of the scope when none is provided in the request. The format of the scope string is a space delimited string within the URL parameters. An example of specifying multiple scopes is using the "curl" command as follows:

```
```

Beginning with the 6.0.0 firmware release, the request can omit the scope and DataPower will use the field, as shown in Figure 4, to populate the scope attribute from the request. This field is found on the **Advanced** tab of the OAuth client profile object.

**Figure 4. Default scope field**

You can send the request by using the following format:

```
```

The above request with the default scope field shown in Figure 4 is equivalent to the following request with no default scope. Note that for the URL-encoding, `%20` is a space and `%2f` is a forward slash.
Determining the resource owner

By default, DataPower determines the user name from multiple steps during the OAuth process. First, by what already exists in the token, followed by the authenticated identity, and then the actual identity. This may not be what is required to track the resource owner's identity. You can customize this behavior through a stylesheet. Figure 5 shows a section of the Advanced tab of the OAuth client profile object. Check the box labeled Custom Resource Owner Handling and select the stylesheet as shown in Figure 5.

Figure 5. Customize the identity of the resource owner

The input for the resource owner process is an XML node set in the format shown in Listing 6. Note that this may be extended in the future to include additional information.

Listing 6. Input for the resource owner process

```xml
<input>
  <identity>....from AAA Extract Identity (EI) ....</identity>
  <credentials>.... from AAA Authenticate (AU) ....</credentials>
</input>
```

The output of the stylesheet must conform to the format shown in Listing 7. Since this retrieves the resource owner's identity, it should not return a failure.

Listing 7. Output for the resource owner process

```xml
<result>
  <resource-owner-name>internal resource owner name</resource-owner-name>
</result>
```

Listing 8 shows a snippet from the stylesheet. This snippet encrypts the resource owner information such that when the information is presented in an authorization/consent form, the user will not know his or her internal enterprise identity.

Listing 8. Sample of the custom resource owner processing

```xml
<xsl:template match="/">
  <result>
    <!-- encrypt the credential, so it will not be shown on the html page -->
    <xsl:variable name="oauth-identity" select="/input/identity/entry[@type='oauth']"/>
```
Customizing the authorization/consent form

DataPower generates the authorization/consent form using a script, which provides maximum flexibility for you. For example, you can add your company logo to the form, add additional information about the application, or add a link to a description about what the application will do. For more details, see the provided file, store:///OAuth-Generate-HTML.xsl, in the Download section to create your own local:///coolapp-az-form.xsl.
Conclusion

Part 9 of this multi-series described how you can customize WebSphere DataPower's native OAuth scope and identity support. By using the attached stylesheets in the Download section, you can use the extension points individually or combine them in different combinations to meet your business requirements.

Acknowledgments

The author would like to thank Paul Glezen for reviewing this article.
## Downloads

<table>
<thead>
<tr>
<th>Description</th>
<th>Name</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Code sample</td>
<td>code_sample.zip</td>
<td>3KB</td>
</tr>
</tbody>
</table>
Resources

- OAuth 2.0 IETF specification
- Regular expression
- WebSphere DataPower Knowledge Center
- WebSphere DataPower library
- developerWorks WebSphere zone
- WebSphere DataPower discussion forum
About the author

Shiu Fun Poon

Shiu Fun Poon is the Security Architect in the WebSphere DataPower Appliance development team. She has worked in many different areas and features of DataPower appliances. She currently leads DataPower’s support for OAuth 2.0 functionality. Her past DataPower projects included WS-Security Policy, various WS-Security features, appliance RAS features, 7198/7199 appliance release, and third party integration with the appliance.

© Copyright IBM Corporation 2015
Trademarks
(www.ibm.com/developerworks/ibm/trademarks/)