Implementing OAuth on IBM WebSphere DataPower Appliances, Part 3: Form login policies and the role of AAA

Paul Glezen (pglezen@us.ibm.com)
Consulting IT Specialist
IBM

Shiu Fun Poon (shiufun@us.ibm.com)
Security Architect
IBM

May 2015
(First published 01 August 2012)

Part 3 of this multi-part article series discusses DataPower AAA policies, form login policies, and their role in supporting the OAuth 2.0 protocol. Since the form login policy provides value beyond its OAuth role, a form-based authentication service proxy is implemented with a multi-protocol gateway to demonstrate how the AAA policies and form login polices work together. As mentioned in Part 2, the WTS wizard is employed in a form-based OAuth scenario to demonstrate the automation of many of the manual MPGW steps.

View more content in this series

Introduction

AAA is a framework within the WebSphere DataPower firmware. It stands for authentication, authorization, and auditing. DataPower takes advantage of AAA extensively to support the OAuth 2.0 protocol. AAA is used to authenticate both the resource owner’s and OAuth client's identities. It is also used for authorizing a request. In release 3.8.1, DataPower introduced form-based authentication, which is tied closely with the web application firewall service. As of the 5.0 firmware release, its support was expanded to other service objects. In Part 3 of the series, we will cover how AAA is used for OAuth support and explain how form-based authentication works with DataPower. You cannot use form-based authentication in an XML Firewall service.

Types of OAuth support in DataPower

There are two types of support for OAuth within DataPower; both rely heavily on AAA.

1. Native support:
   - DataPower acts as an authorization server. An extension point is provided for customization.
• DataPower acts as an enforcement point for the resource server. An extension point is provided for customization.

2. Out-of-the-box (OOTB) support with Tivoli® Federated Identity Manager V6.2.2:
• Federated Identity Manager acts as an authorization server.
• DataPower acts as enforcement point for the resource server.

For an overview of the OAuth protocol, refer to Part 1 of this series, Introducing OAuth 2.0 support in DataPower.

An OAuth client is identified by the client id and optionally verified through a client secret. The resource owner grants permission to an OAuth client to access the owner’s resource within a given resource scope, without sharing the resource owner’s credential with the OAuth client.

Overview of AAA
Figure 1 shows an overview of AAA. AAA is made up of seven phases. Some phases consume the results from a previous phase.

Figure 1. Overview of AAA

![AAA (Authenticate, Authorize, Audit) Policies](image)

- **Extract identity (EI):** This defines how a requestor’s identity is extracted from the incoming request.
- **Authenticate (AU):** This defines how to authenticate the identity extracted from EI.
- **Map identity/Map credential (MC):** This defines how to map the credential from AU to another credential.
- **Extract resource (ER):** This defines how to determine the requested resource from the incoming request.
- **Map resource (MR):** This defines how to convert the resource name from ER to another form for authorization.
- **Authorize (AZ):** This defines how to verify whether the identity from EI/AU/MC can access the resource from ER or MR.
- **Post process (PP):** This provides auditing and defines how to convert the identity to another security token.
**Note:** From firmware 5 to 6, the names of the AAA phases changed from verbs to nouns. For example, "Extract Identity" became "Identity extraction."

The following sections describe the role of each AAA phase in terms of its relevance to OAuth scenarios. The action taken in a phase depends on the OAuth role addressed. The three roles are:

- OAuth authorization server
- Standalone enforcement point (EP) for the resource server
- EP in conjunction with IBM Federated Identity Manager (FIM)

Table 1 provides a column for each of these roles. Each row corresponds to a box in Figure 1. It lists the configuration for that AAA phase pertinent to the role.

### Table 1. Summary of AAA phases for each OAuth role

<table>
<thead>
<tr>
<th>AAA phase</th>
<th>Authorization server</th>
<th>PEP standalone</th>
<th>PEP with FIM</th>
</tr>
</thead>
</table>
| EI        | • Extract and verify OAuth client identity using the client ID and client secret.  
           • Extract the resource owner’s identity (except in the case of the client credential grant type where the client is the resource owner). | • Extract the OAuth client credential (client ID and potentially client secret). Only done for confidential clients.  
           • Extract and verify access token. If the client credential is provided, it will compare this to the client credential that originally requested the access token as an additional check. | Extract OAuth client credential using any method. |
| AU        | Define how to authenticate the resource owner from EI. | Select **Pass Identity Token to Authorize Step** since the access token was verified in the EI phase. | Client authentication may be performed using any method. |
| MC        | Define how to map the resource owner’s credential from EI or AU. | Select **None**. There is no resource owner in this step. The access token was verified in the EI step. | Select **None**. |
| ER        | • Select **Processing Metadata**.  
           • Choose **oauth-scope-metadata** for “Processing Metadata Items.” | • Select **Processing Metadata**.  
           • Choose **oauth-scope-metadata** for “Processing Metadata Items.”  
           • (Optional) Identify other resources that need to be verified against the scope. This can be done in the MR phase with a custom stylesheet. | Use any method to extract the resource. |
| MR        | Select any addition verification that is needed for the scope. Usually this is **None**. | (Optional) Verify scope from the access token against output from the ER phase. The method is "custom," requiring a stylesheet. | Use any method to map the resource. |
| AZ        | Select **Allow Any Authenticated Client**. You may select a different option if you wish to restrict an authenticated resource owner’s access to a scope. | Select **Allow Any Authenticated Client**. | Select **Contact OAuth STS.**  
           • Provide the FIM authorization endpoint information.  
           • Indicate whether DataPower should enforce the scope check or defer to backend resource server. |
Post-processing is optional in the native OAuth protocol support.

For DataPower acting as an enforcement point, with Federated Identity Manager as an authorization server, you can use PP to generate the token produced by the AAA policy.

Figure 2 illustrates steps for the case where the service proxy is an enforcement point rather than an authorization server.

**Figure 2. Flow chart for defining AAA**

```
Define AAA to be used by DataPower
EI: OAuth
Registered Clients: OAuthClientGroup
ER: Processing Metadata
  oauth-scope-metadata

Is AAA for DP
as an Authorization
Server?

no

AJ: Pass Identity Token

yes

Verify scope from
access token or other
ER?

access token

AZ: Allow any authenticated client

1. Select other ER method
2. MR: Custom
   - return verified-scope from access token if scope verified
   - return an invalid scope from access token
     (or empty value) if resource is not allowed.

Done
```

Figure 3 describes AAA policy configuration in the case of an authorization server.
Form-based authentication

Form-based login authentication presents a user with an HTML login form. The user enters his or her credential (for example, name and password), and submits the form. These credentials are used for authentication.

For OAuth, the resource owner may be presented with a form for authentication. In this section, we will cover how DataPower supports form-based authentication and how it can be used as part of the OAuth flow by using the web token service (WTS) or multi-protocol gateway (MPGW) as the service gateway. Note that the XML Firewall is not supported for form-based authentication.

Then, we will demonstrate how AAA policies can be created to implement a simple form-based authentication service proxy using a MPGW. This scenario is independent of OAuth. Next, we’ll import an OAuth-specific AAA policy and use it to create a WTS using the wizard. An OAuth-specific AAA policy references client registration objects and uses special EI settings for OAuth. These details will be covered in each of the scenario-oriented articles in Parts 4, 5, and 6.

In this article, we undertake the following tasks:

1. Create an HTML form login policy.
2. Create an authenticated AAA policy to support form-based login.
3. Create an unauthenticated AAA policy to support form-based login.
4. Create a MPGW service proxy implementing form-based login using the two AAA policies.
5. Created an OAuth-based WTS using an imported AAA policy.

In Part 2, you imported an HTML form-based login policy and AAA action and then used them to create a WTS manually (without the wizard). In this part, we'll be creating them explicitly and incorporating them into a MPGW.
The Downloads section of this article provides the following files:

- **OAuthArticleKeys.zip**: This contains the shared secret key and SSL public and private keys needed for the configuration. Unzipping it creates a keys directory with the key files. This archive is the same for all articles in the series. You only need to download it once.
- **Part3BeginState.zip**: This is a DataPower import holding the OAuth AAA policy (that references the shared secret key) and an SSL Proxy Profile (that references the SSL public/private key files).
- **AccountLoopback.zip**: This is a DataPower import holding a simple backend service. This backend listens on Port 5001. It is used as the backend for several parts of this article series. It only needs to be imported once.

### Domain preparation

1. Create a new DataPower application domain for this article and switch to it.
2. Set the default log level of the domain to **information**.
3. Upload the SSL keys and shared secret key from OAuthArticleKeys.zip into the cert: folder.
4. Import the **Part3BeginState.zip** archive by selected all objects. Check that the following objects are in the up state.
   a. An SSL Proxy Profile named **oauth-server-sslpp**
   b. An AAA policy named **OAuth-FormLogin-AAA**
5. Import **AccountLoopback.zip** for the backend if you haven't already done it for another exercise. It only needs to be imported once.

### Create an HTML form login policy

1. Type **html** in the search box under the "Control Panel" and select **Add**, as shown in Figure 4.

**Figure 4. Create a new HTML forms login policy**

2. Specify the necessary information for the HTML form as shown in Figure 5. SSL should be considered for the form submission, since sensitive information (such as username and password) is transmitted using the form.
   - **Name**: **ACME-LoginForm**
   - **Redirect URL Type**: Host – redirects are based on an inbound host header.
   - **SSL Port**: **5030**. This is the listing port for the submission. This matches the listing port on the MPGW or WTS that will use the AAA referencing this form login policy.
• **Forms Storage Location:** Appliance. This indicates all the necessary HTML pages for supporting form-based login is stored on the appliance.

**Figure 5. Specify an HTML forms login policy**

3. Click **Apply** (at the top).

After the form-login policy has been created, there should now be two: the one you just created and the one that was imported (to be used later).

**Create an AAA policy for form-based authentication support**

1. Create a new AAA policy by entering **AAA** in the search box under the Control Panel and click **Add** as shown in Figure 6.
2. Create a new AAA policy for authenticated access with the following settings (see Figure 7):
   - **Name**: PlainFormLoginAAA
   - **Identity Extraction**: Select **HTML forms-based authentication** as the method. Choose the HTML form policy from Step 2 for the **HTML forms-based login policy**.
   - **Authentication**: Use an AAA information file; choose **store:///AAAInfo.xml**.
   - **Resource extraction**: URL sent by client

3. Click **Apply**.

4. Create another AAA policy for the unauthenticated access to the Login/Error/Logout form. Repeat the above steps for a new AAA policy and use the following setting for the new unauthenticated AAA.
   - **Name**: UnAuthPlainFormLoginAAA.
5. Click **Apply**.

You should now have three AAA policies: the one you imported and two you just created. The one you imported will be used later for the WTS creation wizard. The two you just created will be used in policy rules of the MPGW created in the next step.

### Create the Multi-Protocol Gateway using AAA policies

In this section, you use the two form-based AAA policies just created to configure a Multi-Protocol Gateway that implements form-based authentication. This demonstrates the form-based authentication capability beyond its application to OAuth.

1. In the **Control Panel**, click the **Multi-Protocol Gateway** icon and click the **Add** button.
2. For name, enter **MP-Form-Based**.
3. Set the **Request Type** and **Response Type** to **Non-XML**.
4. Set the **Default Backend URL** to `http://127.0.0.1:5001`.
5. Create a **Front Side Protocol** by clicking on "+" and choose **HTTPS (SSL) Front Side Handler**.
6. Name the HTTPS Front Side Handler as **HTTPS_5030**.
7. For **Port Number** enter **5030**. This must match the port defined in the HTML Forms Login Policy that the AAA polices are referencing.
8. Check the **GET** method.
9. For **SSL Proxy** select **oauth-server-sslpp** that was imported.
10. Click **Apply** for the front side handler panel. This should bring you back to the MPGW panel.
11. Click "+" for the **Multi-Protocol Gateway Policy** to create the policy.
12. For **Policy Name** enter **MP-Form-Based-Policy**.
13. Click **New Rule**. For **Rule Name**, enter **MP-Form-Based-Favicon-Rule** and set **Rule Direction** to **Client to Server** (see Figure 8). This rule will handle favicon.ico to avoid processing the icon requests from the browser. It uses the following three actions:
   a. **Match rule**: Create a new match rule called **MatchFavicon**. The URL is `/favicon.ico`.
   b. **Advanced action > Set Variable**: `var://service/mpgw/skip-backside = 1`.
   c. **Results** action.
14. For a second rule, click the **New Rule** button again and set the direction to **Client to Server**. This rule will handle the form-based authenticated access and resource backend access.
   a. **Rule Name**: **MP-Form-Based-Authenticate-Rule**
   b. **Match rule**: Create a new match rule called **MatchAuthenticated**. In the **Main** tab set the **Match with PCRE** to on. In the **Matching Rule** tab, add a URL match with the following regular expression (no spaces), `/getAccountBalance|j_security_check`, as the match rule. Click **Apply** and **Done** to save the match action.
   c. **Advanced action > Convert Query Params to XML**: Click **Done**.
d. **AAA action**: Choose **PlainFormLoginAAA** from the list of AAA policies for the authenticated access.

e. **Action**: Result action.

15. For the third rule, click the **New Rule** button again and set the direction to **Client to Server**. This rule will handle the form-based unauthenticated access to the Login/Logout/Error form.

a. **Rule Name**: MP-Form-Based-LoginPage-Rule

b. **Match rule**: Create a new match rule called **MatchLogin**. In the **Main** tab, set the **Match with PCRE** to **on**. In the **Matching Rule** tab, add a URL match with the following regular expression (no spaces), */(Login|Error)Page\.(htm(l)?(\?originalUrl=.*)?*, as the match rule. Click **Apply** and **Done** to save the match action.

c. **AAA action**: Choose **UnAuthPlainFormLoginAAA** from the list of AAA policies for the unauthenticated access to the forms.

d. **Action**: Result action.

**Figure 8. Create rules for form-based authentication**

16. After the rules have been entered, click **Apply Policy** and then **Close Window**.

17. Click **Apply** for the Multi-Protocol Gateway.

This completes the configuration of a form-based authentication service proxy. Here are some things to keep in mind regarding this simple example.

- Its valid users are defined in the **store:///AAAInfo.xml** file in this sample. But, you can copy this file to **local:///AAAInfo.xml** and add your own users. You can reference LDAP or use any of the other authentication methods.
• In Step 14b, the match rule specified /getAccountBalance or /j_security_check. The getAccountBalance URL is the only one authenticated. No other URL will be authenticated. In practice, there should be a default rule that catches "other URLs" and handles them appropriately.
• You can test the login using https://host:5030/getAccountBalance, where <host> is your DataPower application IP. Use "fred" and "smith" for the user name and password, respectively since these are defined in the default AAAInfo.xml file. The authentication should succeed and allow the AccountLoopback to reply with the account balance.

Create the Web Token Service using the AAA policy

In the previous exercise, we demonstrated how form-based login policies and AAA policies are used to implement a form-based login authentication service proxy. It was not an OAuth scenario; but, it employed tools that are heavily used in OAuth scenarios. It required creating all the multi-step policy rules from scratch, which served to give us a deeper understand of just how these elements work together.

In this section, we create a WTS using an imported AAA policy named OAuth-FormLogin-AAA. This AAA policy is similar to PlainFormLoginAAA that we configured earlier. It differs by specifying OAuth in some of the AAA stages and referencing client registration objects that will be covered in the scenario-driven articles later in this series (Parts 4, 5, 6, and 8). This sample will show how the WTS wizard generates much of what we created manually in the previous section for an OAuth-based form login.

1. Create a new Web Token Service by entering web token in the search box under the Control Panel. Select New Web Token Service. For Web Token Service Name, enter ACME-OAuth-AZSvr and click Next, as shown in Figure 9.

   Figure 9. Using the wizard to create a WTS for OAuth

2. Enter 5031 for the port as shown in Figure 10. This is the port specified on the imported form-based login policy named OAuthFormLoginPolicy. These must match. Reuse the imported SSL Proxy Profile named oauth-server-sslpp, click "+" (Add), and then click Next.
3. Choose the imported AAA policy named **OAuth-FormLogin-AAA**, which supports both OAuth and form-based authentication for the resource owner. Then click **Next**, as shown in Figure 11.

**Figure 11. Choose the AAA**

4. The WTS wizard automatically creates the necessary rule and AAA action for the unauthenticated access required by the form. Click **Commit** to save the WTS, as shown in Figure 12.

**Figure 12. Commit to create the new WTS**

5. Click the **View Web Token Service** button.
6. At the bottom, click the "..." button next to **Processing Policy**.
Figure 13. Web Token Service Processing Policy

Note the following about the panel in Figure 13:

- The three rules created manually with the MPGW were created automatically for us using the WTS wizard: one for favicon, one for authenticated, and one for unauthenticated.
- Hover your cursor over the AAA action of the bottom rule. You'll see it references the AAA policy that we imported, **OAuth-FormLogin-AAA**. Hover your cursor over the AAA action of the middle rule. You'll see that it references an AAA policy that didn't exist before: **OAuth-FormLogin-AAA_unauthenticated**.
- Navigate to **Objects > XML Processing > AAA Policy**. For a preview of things to come, explore the imported and new AAA policy created by the WTS wizard. The AAA policy list is shown in Figure 14.

Figure 14. AAA policies after WTS Wizard
Conclusion

This article explained how the AAA policy is used to support OAuth within DataPower. Additionally, it covered how to configure form-based authentication in AAA for user identity extraction. The article also showed how the wizard for the Web Token Service simplifies the complexity of form-based resource owner authentication when used by the OAuth authorization server.

Acknowledgements

The authors would like to thank Martin Lansche and John Rasmussen 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>OAuthArticleKeys.zip</td>
<td>3KB</td>
</tr>
<tr>
<td>Code sample</td>
<td>Part3BeginState.zip</td>
<td>519KB</td>
</tr>
<tr>
<td>Code sample</td>
<td>AccountLoopback.zip</td>
<td>524KB</td>
</tr>
</tbody>
</table>
Resources

- IBM WebSphere DataPower SOA Appliance Handbook
- IETF Web Authorization Protocol document (oauth)
- WebSphere DataPower Knowledge Center
- WebSphere DataPower library
- developerWorks WebSphere zone
- WebSphere DataPower discussion forum
About the authors

Paul Glezen

Paul Glezen is an IT Specialist with IBM Software Services for WebSphere (ISSW). He started application server consulting in 1998 with the IBM CORBA-based C++/Java product known as Component Broker. His current product focuses are DataPower and WebSphere Application Server development and security configurations.

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 2012, 2015
Trademarks
(www.ibm.com/developerworks/ibm/trademarks/)