Implementing OAuth on IBM WebSphere DataPower Appliances, Part 8: Grant type scenario for implicit grants

John Rasmussen (rasmussj@us.ibm.com)  
Senior Software Developer  
IBM  
05 November 2014

Part 8 of this multi-part series describes WebSphere® DataPower Appliance support for the OAuth 2.0 implicit grant type. It covers configuration for both the authorization server and the enforcement point.

View more content in this series

Introduction

The OAuth authorization protocol focuses on sharing user resources without sharing user credentials. The implicit grant type is typically used within "public clients", such as browser-based web applications that have the following characteristics:

- They cannot guarantee the confidentiality of their client secret.  
- Their access token is immediately granted to the client application by the authorization server after the authenticated resource owner grants permission.

Public clients must be able to follow redirection URIs. The implicit grant does not provide for refresh tokens and, therefore, is typically used when temporary access is granted or when the user is regularly logged into the client API provider. Since the access token may be placed within the URI, the user-agent (browser) must be strongly trusted where the concern for access token accessibility is limited.

For more information about two-legged and three-legged processes, see Part 1 of the series, Introducing OAuth 2.0 support in DataPower firmware revision 6.0 or the IETF OAuth 2.0 Authorization Framework.

Figure 1 shows an example of an implicit grant flow. The labeling of the steps was chosen to match with Figure 4 in Section 4.2 of the OAuth 2.0 Specification.
The client API redirects the resource owner's user agent to the authorization endpoint. Among the elements of the request are:

- The client ID and, optionally, the requested scope.
- The redirection URI for the client to ultimately receive the access token. The redirection URI may optionally have been defined during client registration.

The resource owner's user agent follows the redirect to the authorization server.

The authorization server authenticates the resource owner and validates the resource owner's permission to grant the client access to their resources.

The authorization server redirects the user agent back to the client application's web resource with the access token as a #hash URI fragment.

The user agent retrieves an HTML/JavaScript resource, which is capable of parsing the #hash URI.

The HTML/JavaScript is returned and parses the access token.

The access token is returned to the original redirection URI.

Figure 1. Implicit grant type flow

This tutorial will step through the configuration of the OAuth implicit grant type on DataPower and the generation of an access token and its use for accessing the resource owner's assets.

The basic building blocks consist of the following:

1. Register the OAuth client by creating an OAuth client profile and associating it with an OAuth client group.
2. Create AAA policies for resource owner authentication and a resource enforcement point.
3. Create an authorization server using the web token service.
4. Create a resource server using the Multi-protocol Gateway (MPGW).

An OAuth client receives its AAA characteristics through membership in an OAuth client group. AAA policies reference OAuth client groups to specify which clients are eligible to participate in the OAuth flow. An OAuth client profile contains detailed information specific to a client, such as which grant types and OAuth roles are permitted. An OAuth client group serves to organize clients for management purposes. An AAA policy references a client group. Once the OAuth client and group are created, you can use AAA policies for authentication and authorization. This relationship is shown in Figure 2.

**Figure 2. AAA policies referencing OAuth clients**

![Diagram showing WTS, AAA, OAuth client group, Client profile 1, Client profile 2, Client profile 3 connections]

The web token service (WTS) generates tokens based on results from the AAA policy. You can configure a MPGW as an enforcement point. Its AAA policy will verify access tokens and allow or deny access to resources.

**Preparing for DataPower configuration**

In order to focus on OAuth configuration, some pre-configured DataPower objects are provided in the Downloads section of this tutorial.

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. This tutorial uses the same set of shared secret and SSL keys used by the other articles in this series. Download **OAuthArticleKeys.zip** and unzip it if you haven't already. Upload **sharedSecretKey.txt**, **sslserver-privkey.pem**, and **sslserver-sscert.pem** into the **cert:** directory of your new application domain.
4. This tutorial uses a simple XML Firewall for the backend. This is packaged in an archive named **AccountLoopback.zip**. Download and import this file if you haven't already. If this XML Firewall is already running in another application domain from a previous exercise, then skip this step to avoid a conflict on TCP port 5001.
5. Download and import **Part8BeginState.zip**. It contains the following:
   - An SSL Proxy Profile, **sslserver**, that references key files uploaded in Step 3.
   - A shared secret key object, **crypto-key**, that references the shared secret key file uploaded in Step 3.
   - An HTTPS front side handler, **HTTPS_EnforecmentPoint_5081**, for the Multi-Protocol Gateway to be created later.
   - A pair of matching rules, **MatchFavicon** and **MatchAll**, to streamline the creation of policy rules.
6. Verify the import:
   - a. The SSL Proxy Profile **sslserver** state is **up**.
   - b. The Crypto Shared Secret Key **crypto-key** state is **up**.
   - c. The XML Firewall **loopback** state is **up**.
d. The HTTPS front side handler `HTTPS_E_EnforcementPoint_5081` should be present. It will not be up until you attach it to a service proxy.
e. Both matching rules, `MatchFavicon` and `MatchAll`, are up.

You can test the loopback process by performing the following cURL GET requests. It returns a sample REST document if it is properly configured. You may wish to review the XML Firewall and the policy rules it contains.

```bash
curl http://<appliance-host>:5001/getCustomerInfo
returns { "id":0, "name":"", "access_token":"Unknown" }
curl http://appliance-host:5001/getAccountInfo
returns { "name": "myAccount", "balance":1.00 }
```

### Registering the client

Registration of an OAuth client in DataPower amounts to creating an OAuth client profile object and assigning it to an OAuth client group. Figure 3 shows the OAuth client profile creation window. Follow the steps below to start the exercise:

1. Search for "OAuth Client Profile" in the WebGUI search box.
2. Click **OAuth Client Profile**.
3. Click **Add** and complete the following fields:
   - **Name**: oauth-implicit
   - Leave **OAuth Role** as Authorization and Enforcement Points
   - **Grant Type**: Check Implicit Grant, uncheck Authorization Code
   - **Client-Type**: Public
   - **Scope**: `/getAccountInfo|/getCustomerInfo`
   - **Shared Secret**: crypto-key
   - **Redirect URI**: `https://{1}$+`
     - Be sure to click the **Add** button for the redirect URI entry.
The **Shared Secret** object is required to support the encryption of the access token. It was imported at the beginning. For information on generating shared secret keys, see the **The shared secret key** section in Part 4 of the series.

Add a new client group and add the new client profile to it. The OAuth client group window is shown in Figure 4.

1. Search for "OAuth Client Group" in the WebGUI search box.
2. Click **OAuth Client Group**.
3. Click **Add**.
4. For **Name**, enter `oauth-implicit-group`.
5. In the **Client** dropdown list, select the `oauth-implicit` client profile and click **add**.
6. Click **Apply**.

![Figure 3. OAuth Client Profile configuration](image-url)
Creating the AAA policies

This section creates the AAA polices used in the service proxies. The first subsection details the AAA policy creation for the web token service. The second subsection provides the required settings of the AAA policy for the enforcement point without the detailed screenshots.

Creating the Web Token Server AAA policy

In this section, you create an AAA policy used by the web token service to implement the authorization service. It will respond to authenticated requests by providing an access token valid for a specific client and scope. Before creating the AAA policy, let’s first discuss how we’re going to authenticate and authorize client requests.

Most configurations will use an LDAP server such as Tivoli® Access Manager for authentication and authorization. To focus on OAuth configuration in this tutorial, we’re going to use the convenience and simplicity of the DataPower AAA info file. You can use entries within the AAA info file to authenticate your OAuth client credentials and authorize the clients’ requests for specific resources.

You will be using a simple AAA info file in the store: directory as shown in Figure 5.
Figure 5. AAAInfo.xml in the store: directory

Within the file are authentication elements, `<Authenticate>`, and authorization elements, `<Authorize>`. In this example, you are authenticating using basic authentication headers with the username "fred" and password "smith". You will not reference the AAA info file for authorizing the resource. Rather, you will be using the "Allow any authenticated client" option within the AAA policy so that a specific authorization entry is not required.

Listing 1 shows the entry in `store:///AAAInfo.xml` pertinent to this example.

Listing 1. Sample access token request and response

```xml
<Authenticate>
  <Username>fred</Username>
  <Password>smith</Password>
  <OutputCredential>admin</OutputCredential>
</Authenticate>
```

Create an AAA Policy for authentication as follows:

1. Search for "AAA" in the Web UI search box.
2. Click **AAA Policy**.
3. Click **Add**.
4. For **Name**, enter **WTS-AZ**.
5. Select the **Identity extraction** tab as shown in Figure 6.
6. Select the following two methods: **HTTPAuthentication Header** and **OAuth**.
7. For the **Registered OAuth clients**, select the **OAuth client group** created in the previous step (**oauth-implicit-group**).
8. Select the **Authentication** tab as shown in Figure 7.
9. Select **store:///AAAInfo.xml**.
10. Select the **Resource extraction** tab as shown in Figure 8.
11. Select **Processing Metadata**.
12. Choose **oauth-scope-metadata**. This identifies the scope for the client request.
Figure 8. Define the Extract Resource step

13. Select the **Authorization** tab as shown in Figure 9.
14. Select **Allow any authenticated client**.
15. Click **Apply**.

Figure 9. Authorize request
Creating the enforcement point AAA policy

Creating the enforcement point AAA policy is similar to the one created in the previous section. From the list of AAA policies, click Add as you did in Step 3 above. The settings are:

- **Name:** MPGW-EP
- **Identity extraction:** Check the HTTP Authentication header and OAuth. For the Registered OAuth clients field, select oauth-implicit-group that was created earlier.
- **Authentication:** Pass the identity token to authorization phase.
- **Resource extraction:** Check Processing metadata and select oauth-scope-metadata.
- **Authorization:** Allow any authenticated client.

Configuring the authorization server

The authorization server will be implemented with a web token service. This service receives the implicit grant access request and responds with an access token for the specific scope.

1. Search for "New Web Token Service" in the Web UI search box.
2. Enter the name as WTS-AZ as shown in Figure 10.
3. Click Next.

   **Figure 10. Create the Web Token Service**

   ![Create a Web Token Service](image)

   Configure a source address to receive the authentication requests. There could be multiple source addresses. The authentication requests will contain credential information, so it is important to use SSL/TSL to encrypt the request.

4. Enter 5080 for the port number as shown in Figure 11.
5. For SSL proxy profile, select sslserver. This was created by the import.
6. Click Add to add the source address.
7. Click **Next**.
8. Select the AAA policy **WTS-AZ** created in the last section, and click **Next** as shown in Figure 12.

**Figure 12. Select an AAA policy**

9. Click **Commit** to create the new web token service as shown in Figure 13.

**Figure 13. Confirm changes**

After the web token service is created, you can view the object's processing policy (see Figure 14). It has created two rules, both in the client-to-server direction:

- Match `/favicon.ico` to ignore the icon requests sent by browsers.
- Match `*` to execute the "Convert Query Params to XML" action for the necessary HTTP conversion and the AAA policy action.
Having created the authentication WTS, you now need to create an enforcement point service. This service authorizes the requests by verifying the access token, including its scope and lifetime. You will use a MPGW for this purpose.

**Configuring the enforcement point**

The WTS wizard automatically created the policy rules to process the icon (favicon) requests and authentication requests. The MPGW does not provide this automatic policy creation. We'll assume a working knowledge of MPGW configuration by stepping through its configuration, but we have omitted the screenshots.

1. Navigate to the list of multi-protocol gateways and click **Add**. Use Figure 15 as a guide for the steps that follow.
2. For **Multi-Protocol Gateway Name**, enter `implicitEP`.
3. For **Summary**, enter `Part 8 enforcement point`.
4. For **Default Backend URL**, enter `http://127.0.0.1:5001`. This is the AccountLoopback XML firewall that was imported earlier.
5. For the Front Side Protocol, select `HTTPS_FrontSideHandler_ENFORCEMENTPoint_5081` from the list. Click **Add** to add it to the list.
6. Set the Response Type to **Pass through**.
7. Set the Request Type to **Non-XML**.
8. In the **Front Side Protocol** list, select the **HTTPS_EnforcementPoint_5081** front side handler that was imported earlier. Click **Add** to add it to the handler list.
9. Click the "+" next to **Multi-Protocol Gateway Policy**. This starts the policy editor. The policy will have two rules: one for favicon and one for token enforcement. The completion of the steps in the next section is summarized in Figure 16.

**Figure 16. Summary of implicitEP_policy**

10. In the **Policy Name** field, enter **implicitEP_policy**.
11. Click the **New Rule** button and name the rule as **implicitEP-favicon_rule**.
12. Change Rule Direction to **Client to Server**.
13. Double-click the match action and select the **MatchFavicon** rule imported earlier.
14. Add a results action after the match action.
15. Click the **New Rule** button once again. Name the rule **implicitEP-enforcement_rule**.
16. Ensure the Rule Direction is **Client to Server**.
17. Double-click the match action and select the **MatchAll** rule imported earlier.
18. Drag an **Advanced** action after the match action. Double-click it and select **Convert Query Params to XML** from the list. Click **Next** and then click **Done**.
19. Drag an **AAA** action to the end of the rule. Double-click it and select the **MPGW-EP AAA** policy that you created earlier. Click **Done**.
20. Click **Apply Policy** and close the policy editor window.
21. Click **Apply** for the MPGW panel.

## Validating the access token

You have completed the configuration of the authorization server and the resource server on the DataPower appliance. Now you can use cURL, a command line tool, to send requests to the DataPower appliance to request access tokens. You can then use the access token for the resource access.

Listing 2 shows a sample request using cURL to send to the authorization server with the required parameters. It also shows a response containing the access token.

### Listing 2. Sample access token request and response

```bash
curl -k "https://{DP_IP}:5080/authen?
response_type=token&client_id=
oauth-implicit&redirect_uri=
https://192.168.1.105&scope=/getAccountInfo%20/getCustomerInfo&state=
Implicit" --user fred:smith -o implicit.html
```

Note that the output was sent to a file called "implicit.html". Opening this file presents the resource owner challenge to allow access to the client for a specific scope (getAccountInfo and getCustomerInfo). Figure 17 shows the file opened with the browser.

### Figure 17. Request for Resource Owner Permission

**Request for Permission**

Welcome fred

Do you allow Example Inc. access to:

Choosing “Allow access” with no scope selected, DataPower will interpret it as the resource owner grants permission to all the requested scope(s).

- [x] /getAccountInfo
- [x] /getCustomerInfo

[Allow access] [No thanks]

[Submit] Clicking Submit will redirect you to https://192.168.1.105.

When the resource owner acknowledges the request and provides permission, the user agent is redirected to the web resource endpoint with the access token appended to the URI with the #hash fragment. In our simple cURL examples, the browser will not be able to follow the redirect. Figure 18 shows the initial section of this string with termination due to browser window truncation.
Figure 18. Redirection with "#" hash fragment containing access-token and redirection uri

You can now use the access token to authenticate to the resource server. You can simply copy the access token from the browser. In Figure 18, you only see the first section of the token. Make sure you copy the entire string up until any parameters that may follow. Parameters are separated by "&" in the URL encoding.

Our sample resource service is a simple loopback firewall. You can now use the MPGW to validate the access token. As the access token is already URL encoded, you can cut and paste it, as is, into the next cURL example.

Listing 3 shows the access token being sent to the resource service, which is the MPGW in our configuration. The AAA validated the access token and the requested resource (JSON document) is returned by the "loopback" XML Firewall.

Listing 3. Access token verified by the MPGW Resource service

```bash
curl -k https://{DP_IP}:5081/getAccountInfo -H "Authorization:Bearer AAEObd2F1dGgtaW1wbGljaXTYDzcC5a7%2FB5y8hkp0tzyhjSgcXbYGsDC29nxAAwCFcFtnZX65bTh76wy4D%2BkU1Afk1n%2FXFjhyClkw3hQ99NTdMneK0lZ1ak4YJ1Ek1XQmvWxA62spAxuR89%2B4f1M"
{
  "name": "myAccount",
  "balance":1.00
}
```

Conclusion

Part 8 of this series showed how to configure DataPower for the OAuth 2.0 implicit grant type. It demonstrated how to create an OAuth client with the implicit grant type enabled. A Web Token Service is used as an authorization server and a Multi-Protocol Gateway as a resource server. Lastly, the sample request and response of the implicit grant type helped to provide guidance when generating and sending requests to the DataPower appliance.

Acknowledgement

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>Part8BeginState.zip</td>
<td>516KB</td>
</tr>
<tr>
<td>Code sample</td>
<td>OAuthArticleKeys.zip</td>
<td>3KB</td>
</tr>
<tr>
<td>Code sample</td>
<td>AccountLoopback.zip</td>
<td>499KB</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 DataPower zone
- WebSphere DataPower discussion forum
About the author

John Rasmussen

John Rasmussen is a Senior Software Engineer with IBM’s Software Group. He has worked with DataPower Corporation and IBM since 2002 as a product development engineer and services specialist, assisting many clients in the implementation of DataPower appliances. John has experience in software development and security, including work with McCormack & Dodge and Fidelity Investments, and as an independent developer of application software and security systems.

© Copyright IBM Corporation 2014
Trademarks
(www.ibm.com/developerworks/ibm/trademarks/)