Implementing OAuth on IBM WebSphere DataPower Appliances, Part 2: Introducing the Web Token Service

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

Krithika Prakash (krithika@ca.ibm.com)
Advisory Software Engineer
IBM

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

May 2015
(First published 01 August 2012)

Part 2 of this multi-part article series provides an introduction to the WebSphere® DataPower Web Token Service (WTS). The WTS is an authentication and token mediation service used to implement the authorization server role within an OAuth scenario. We'll demonstrate how to configure and debug a WTS. We'll also provide an introduction to the forms-based login policy that can be used within Authentication, Authorization, and Auditing (AAA) policies associated with the WTS to produce login and associated forms.

View more content in this series

Introduction

WebSphere DataPower support for OAuth includes an OAuth authorization service to support the OAuth protocol on the device. The Web Token Service (WTS) provides a powerful mechanism for the rapid creation of this service type. Part 2 describes the fundamentals of the DataPower WTS and its associated objects and demonstrates how to create them. We will explain typical use cases for the WTS and its advantages compared with other service types. The Multi-Protocol Gateway (MPGW) can also be used as an authorization service. Finally, we'll demonstrate use of the WTS with a forms login policy and demonstrate debugging and analysis techniques. These skills are important corner stones for implementing OAuth on DataPower. Part 3 will continue this discussion by demonstrating the WTS creation wizard.
Overview of the Web Token Service

A token service is a service that authenticates an entity and issues a security token. It negotiates trust between client applications and services and removes the need for a direct relationship between them.

DataPower supports token services through objects such as the AAA Policy and the Web Service Proxy. The WTS introduced in DataPower provides a dedicated and simplified service to act as the token service using the existing functionality of the AAA object and other actions of a processing policy. It is a loopback service (a service object without a backend) that can be configured as both the authorization and token endpoint of an authorization server. For information about the role of the authorization server within an OAuth context, refer to Part 1 of this series, Introducing OAuth 2.0 Support in DataPower. Since the WTS is implemented with a multi-step policy within DataPower, customers can expand and modify its behavior using the full array of DataPower multi-step actions.

Creating a Web Token Service

You may manually create a Web Token Service or use the provided wizard.

Creating a Web Token Service with the wizard

The WTS creation wizard is oriented toward OAuth scenarios. Inputs to this wizard include the following:

- AAA policy defining the authorization policies
- SSL Proxy profile for encrypting the transport

Once these are defined for an OAuth scenario, the WTS wizard automates the creation of additional policy rules required for the OAuth authorization server. Since details of defining an OAuth AAA policy are explained in Part 3 of this series, Form Login Policies and the role of AAA, the demonstration of the WTS wizard, which depends on such an AAA policy, is deferred to Part 3.

Creating a Web Token Service manually

To illustrate the manual approach to constructing a WTS, this article steps you through creating a WTS that implements form-based authentication for a simple HTTP request. While this is not an OAuth scenario, it will demonstrate concepts required by OAuth scenarios in later parts of this article series. This WTS depends upon some supporting DataPower objects.

- SSL Proxy Profile for the SSL transport
- AAA Policies for authenticated and unauthenticated service access
- Processing Policy and rules for service access

It is assumed that you are familiar with basic DataPower configuration. So, rather than working through the creation of these objects individually, we've provided a DataPower configuration export.
that you can import into a DataPower application domain. This allows us to concentrate on WTS, forms based login policies, and OAuth configuration details.

1. Create a new DataPower application domain and switch to it.
2. Change the log level for the default log to information.
3. This article uses the same set of shared secret and SSL keys used by the other articles in this series. Download OAuthArticleKeys.zip from the Download section of this article and unzip it. Upload sslserver-privkey.pem and sslserver-sscert.pem into the cert: directory of your new application domain. The sharedSecretKey.txt file is not needed for this exercise, but it will be used in later parts of this article series.
4. Import the Part2BeginState.zip file from the Download section of this article and import all objects.
5. Verify the following after the import:
   - The SSL Proxy Profile object named sslserver exists under Objects > Crypto Configuration > SSL Proxy Profile and is in the up state.
   - Under Services > Multi-Protocol Gateway > Multi-Protocol Gateway Policy, there is an OAuth-WTS1 entry in the up state.

You will create a Web Token Service that does simple authentication of an HTTP request. Requests will be verified for previous authorization via an HTTP cookie. The cookie is sent by the DataPower Forms Login Policy after a successful authentication. Requests failing authentication will be redirected via an HTTP 302 response to an HTML login page defined within the Forms Login Policy.

Figure 1 shows an event diagram for this process. A request for /resource is processed by the authenticated rule. A request lacking credentials is redirected to LoginPage.htm. The login page posts the login form data to the j_security_check URI. This request is processed by the unauthenticated rule. A successful login is redirected to the original requested URI along with a fresh cookie. A login failure is redirected to the error page.

**Figure 1. Forms based login processing events**

GET https://<host>:port/resource

302 Location: https://<host>:port/LoginPage.htm&originalUrl=https://<host>:port/resource

GET https://<host>:port/LoginPage.htm&...

200: LoginPage.htm (j_security_check, _username, _password)

POST https://<host>:port/j_security_check&amp;username=

302 Location: https://<host>:port/ErrorPage.htm

GET https://<host>:port/ErrorPage.htm

---

Implementing OAuth on IBM WebSphere DataPower Appliances,
Part 2: Introducing the Web Token Service
You will need one Processing Policy Rule for the login page and another for access to resources. The unauthenticated login page request will use the Forms Login Policy to produce the HTML login page. The authenticated resource access will require previous authentication.

You will see in the following sections that the WTS wizard can be used to automate the creation of these rules. But, in this first example, we will use the rules supplied in the `Part2BeginState.zip` import and the AAA policies it contains.

Figure 2 shows the unauthenticated rule that was imported. It has a Match action using a regular expression to match `LoginPage.html` or `ErrorPage.html`.

**Figure 2. Unauthenticated policy rule**

The AAA policy shown in Figure 2 uses the **HTML forms-based authentication** in the Identity extraction phase and the `formLogin` HTML Forms-based login policy. Figure 3 shows this in the **Identity extraction** panel of the unAuthenticatedAccess AAA policy.

**Figure 3. Unauthenticated AAA policy**

The HTML Forms login policy (Figure 4) shows the port (5020) used for reaching the login page, as well as the location of various login HTML forms. SSL has been enabled for login processing,
which should always be the case when passing credential information. The **Redirect URL Type** field has been updated for DataPower version 6.0 to provide a Host option. This enhances the previous use of IP address for redirection and provides resiliency to front side load balancing.

**Figure 4. HTML Forms Login Policy**

![HTML Forms Login Policy](image)

The policy will respond to a successful authentication by sending a Set-cookie HTTP response header to the browser. In this way, authentication requests are cached for a lifetime established on the Forms Login Policy. Listing 1 shows an example of the Set-Cookie response header from a successful authorization. Notice the "Max-Age" attribute is equal to the "Inactivity Timeout" of the Forms policy.

**Listing 1. Set-Cookie from authentication**

```
Set-Cookie forms.formLogin.session=5154FB…1jb29raWU+; Max-Age=50; Path=/; Version=1
```

You can explore the other steps of the OAuth-WTS1_unAuthenticatedAccess AAA policy. Since the purpose of this policy is to supply login and error pages, you'll find:

- **Authentication phase** uses "Pass identity token to authorization phase".
- **Resource extraction phase** uses "URL sent to back end".
• **Authorization phase** uses "Always allow".

The OAuth-WTS1_authenticateAccess rule matches requests for /resource/* and j_security_check URIs. These are the standard resource requests and the security check sent by the login page. The policy shown in Figure 5 contains a Convert Query Params to XML action prior to the AAA action. This converts the HTML query strings sent by the login page to an XML document for processing by the AAA action.

**Figure 5. Authenticated access rule**

The "authenticateAccess" AAA policy also uses the Forms Based Login extract identity option, but differs from the "unauthenticatedAccess" AAA by using "Use AAA information file" as the Authentication option. If you are unfamiliar with the use of the AAA Info file, you can view the details in store:///AAAInfo.xml. You'll see the list of pre-authenticated credentials within. Listing 2 shows an example of an authentication entry from the AAAInfo.xml document.

**Listing 2. Sample authenticate entry from AAAInfo.xml**

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

The final action within the authenticate access rule is a simple XSLT transformation. In a traditional service proxy, you would use the token received from the WTS to access the real service, perhaps a resource server with your newly minted OAuth access token. But, here we add a transform action to produce a simple HTML page returned in the event of successful authentication.

**Note:** Do not use this technique to protect a final transform action from executing. Because it's part of the rule, it always executes! You won't always see its output in your browser; but it always executes. This technique was used here simply to demonstrate the path of the redirects and avoid setting up another backend service. The protected resource should always be a separate service.

Listing 3 shows the XSLT.
Listing 3. youAreAuthenticated.xsl

```xml
<xsl:stylesheet version="1.0"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    xmlns:dp="http://www.datapower.com/extensions"
    extension-element-prefixes="dp">
  <xsl:output method="xml"/>
  <xsl:template match="/">
    <dp:set-response-header name='Content-Type' value='text/html'/>
    <html>
      <head><title>Web Token Service</title></head>
      <body>You have been Authenticated</body>
    </html>
  </xsl:template>
</xsl:stylesheet>
```

Now that you have the supporting objects in place, let's configure the Web Token service and demonstrate its use.

2. Click Add.
3. For Name enter OAuth-WTS1.
   Next, we'll add the interfaces (source addresses) that access the Token Service.
4. Click the Add button in the Endpoints box shown in Figure 6. This displays the dialog shown in Figure 7.
   
   **Figure 6. Adding source endpoints to the Web Token Service**

   ![Figure 6. Adding source endpoints to the Web Token Service](image)

5. Set Port to 5020, same as the port used for the form login policy.
6. Set SSL to on.
7. Set the SSL proxy profile to sslserver.
8. Click Apply.
9. Select the **OAuth-WTS1** from the **Processing Policy** drop down list. This policy was included in the **Part2BeginState.zip** import file. The source addresses and processing policy appear as shown in Figure 8.

10. Click **Apply** to complete configuration of the WTS.

**Figure 8. Selecting the processing policy**

You're done with configuration of the WTS. The processing rules are in place. The AAA policies for both authenticate and unauthenticated access are complete. Let's run through a demonstration to see how it works.

1. Open a web browser.
2. Enter the URL of the device and service you've created. In our example, this is **https://<dp-host>:5020/resource/fee**.
3. If you used the self-signed certificate from the **OAuthArticleKeys.zip** file, then your browser may require you to accept an untrusted certificate. If challenged, accept the certificate. You'll execute the unauthenticated rule and receive a login screen.
4. Enter the credentials of "fred" and "smith" as shown in Figure 9.

**Figure 9. Forms based login policy login form**
Once you enter the credentials of "fred" and "smith", you are authenticated and the XSLT produces the simple page shown in Figure 10.

**Figure 10. Authenticated access**

![Authenticated access](https://192.168.1.105:8888/resource/fee)

You have been Authenticated

This completes the demonstration of a Web Token Service and the use of form-based login.

**Troubleshooting**

Let's look at some of the debugging information to understand what has transpired. In Figure 11, you can see entries from the DataPower log set to information level. Reading from the bottom up, you see the following items:

- Request "Get for /resource/fee"
- Selection of the "resource-check" match
- Conversion of the query strings
- Checking of previous cookies (if any)
- Authentication
- Authentication failure
- Redirection to /LoginPage.htm

**Figure 11. DataPower Log showing authentication failure and redirection**

You can also use the DataPower Probe to view content, service, and system variables and the flow within the individual rules and actions. Figure 12 shows the two requests generated by the GET and subsequent redirection.

**Figure 12. DataPower Probe showing resource request and Login page access**
You can also find valuable information from the browser's perspective. There are many tools available for this. Figure 13 shows Firebug, a Firefox plug-in, which shows the GET and subsequent "302 redirect" and the GET for LoginPage.htm.

Figure 13. Firefox Firebug capture showing Get and 302 redirection

Applying the Web Token Service

The Web Token Service is used as the authorization server token endpoint. Note that WTS does not have a backend and, hence, it cannot be used as the enforcement point for a resource server.

The advantage of using a WTS over a MPGW for an OAuth authorization server in DataPower is that the WTS comes with a wizard to help you configure an OAuth authorization server. It is a simplified service and there are no complicated setups that are irrelevant to a token service. The wizard eases the configuration of form-based login for resource owner authentication in an OAuth protocol exchange.

New in firmware 6.0 release

In the 6.0 firmware release, you can generate the HTML login form using an XSLT file (see Figure 14). An example of such an XSLT is found in store:///Form-Generate-HTML.xsl. This support provides the flexibility to customize the request parameters and it eliminates the last HTTP 302 redirect of Figure 1. It makes the following assumption to simplify the configuration:

- Redirect handling is front side load-balancing friendly.
- No SSL configuration (this support relies on the SSL profile defined for the service object).
- Only one AAA is needed to support HTML form based authentication.

Figure 14. Customized form generation and response
Conclusion

This article explained the DataPower Web Token Service feature. It provided instructions to create and configure a WTS as a simple authorization service. Note that although the functionality provided by a WTS is available in a MPGW or XML Firewall service, the advantages of using WTS are its simplicity and the ease of configuration by using its wizard. As this article demonstrated, you can also use WTS for other types of token exchange; it is not limited to OAuth.

Acknowledgements

The authors wish to thank Ken Hygh, Martin Lansche, and Shiu-Fun Poon for reviewing this article and providing helpful comments.
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>Part2BeginState.zip</td>
<td>523KB</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.

Krithika Prakash

Krithika Prakash is a Software Engineer with ten years of industry experience, specializing in the areas of networking, security, Web services security, and XML/XSLT technologies. She has filed patents and written several articles for developerWorks in the areas of security.

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