Implementing Salesforce federated single sign-on with WebSphere DataPower, Part 3: Implementing a service provider initiated SSO using a signed SAML assertion

Dipak K. Pal (dipakpal@in.ibm.com)
Software Developer
IBM

13 May 2015

This four-part tutorial series describes a Salesforce® federated single sign-on solution using WebSphere® DataPower® as an identity provider. Part 3 discusses how to implement the service provider initiated single sign-on to Salesforce using a signed SAML assertion.

Introduction

Part 3 of this tutorial series focuses implementing a service provider initiated single sign-on (SSO) to Salesforce using IBM® WebSphere DataPower (hereafter called DataPower) as an identity provider, powered by a signed SAML assertion. Figure 1 and Figure 2 represent a high level overview of this solution. This tutorial does not recap the fundamentals of Salesforce federated SSO already covered in Part 1.

Figure 1. Overview of service provider initiated SSO login to Salesforce
Figure 2. Service provider initiated SSO login process flows

Single sign-on setup on the Force.com platform

This section provides the steps to implement an SSO setup on the Force.com platform. If you have already implemented an example from any of the tutorials in this series, you may not need to repeat the same steps again. However, make sure you have configured the additional setup required for this example.

Step 1: Create Salesforce user accounts

First, create a user account (Salesforce Administrator) from the Salesforce developer site if you do not have one already. Figure 3 shows the Salesforce user (Developer Edition) sign-up process.

Figure 3. Salesforce user account (Developer Edition) creation at Salesforce developer site

Examples illustrated by this tutorial series use the following sample user account for an SSO setup. We strongly recommend that you do not enable the SSO feature for your organization's Salesforce administrator because any outage of the identity provider (DataPower) application will prevent all administrative related activities.

User ID: sdfc.user@sdfc.com
Federation ID: sdfc.user@sso.sdfc.com
Implementing Salesforce federated single sign-on with WebSphere DataPower, Part 3: Implementing a service provider initiated SSO using a signed SAML assertion

Figure 4. Enter a federation ID to enable SSO

![Figure 4](image)

Figure 5. Sample user account details at Salesforce

![Figure 5](image)

Step 2: Create a self-signed or CA-signed certificate

Certificates are used for authenticated SSO with an external identity provider. You need a Salesforce certificate for the service provider initiated SSO to sign the SAML authentication request on the Force.com side and to validate the same signature on the identity provider side. Once you create the certificate, download and save it into your local system. Examples illustrated by this tutorial series use the following self-signed certificate (Label: Salesforce SSO) are shown in Figure 6 and Figure 7.
Step 3: Manage the Salesforce domain

Using My Domain (Figure 8), you can define a custom Salesforce domain name. This name helps you better manage the login and authentication for your organization in several ways. Refer to the Salesforce documentation for domain management details. Examples illustrated by this tutorial series use following domain name. We suggest that you use a suitable domain name for your organization. An example of a sample domain name is https://dipak-sdfc-dev-ed.my.salesforce.com/.
Implementing Salesforce federated single sign-on with WebSphere DataPower, Part 3: Implementing a service provider initiated SSO using a signed SAML assertion

Figure 8. My Domain overview at Salesforce

Step 4: Set up SSO settings at Salesforce

1. Login as the Salesforce administrator and go to Setup > Security Controls > Single Sign-On Settings. Make sure the SAML Enabled checkbox is checked in the Federated Single Sign-On Using SAML tab, and click the New button in the SAML Single Sign-On Settings tab. Figure 9 and Figure 10 show the Salesforce SSO setup used by this example. Prior to this SSO setup in the Force.com platform, make sure you have already created the DataPower crypto key and the associated certificate described in Step 6.

Figure 9. SAML SSO setting at Salesforce
2. Upload the DataPower certificate as the **Identity Provider Certificate** such that the Force.com platform can validate the signature associated with the SAML assertion response.

3. Use the self-signed or CA-signed certificate generated in **Step 2** as **Signing Certificate**. Enter the appropriate values in the **Issuer** and **Entity Id** fields. Use the DataPower SSO login URL (such as `https://<dp-host:port>/sso/salesforce/saml/request`) in the **Identity Provider Login URL** field such that Force.com can redirect the request to the identity provider's SSO login URL. The SAML assertion used in this tutorial is not encrypted and contains the **Federation ID** of the login user in the NameIdentifier (NameID) field in the Subject statement.

4. Once you save the SAML SSO settings, note down the Salesforce login URL, such as `https://dipak-sdfc-dev-ed.my.salesforce.com?so=00DF0000000ghLC`, as shown in Figure 9 and Figure 10 above.

**Step 5: Update the Login Page Branding at My Domain**

Go to **Setup > Domain Management > My Domain**, edit **Login Page Branding** and check the appropriate **Authentication Service**. This **Login Page Branding** setup will ensure the display of the SSO login options at the domain specific login page as shown in Figure 11 and Figure 12.

**Figure 10. SAML SSO setting detail**

![SAML SSO setting detail](image)

2. Upload the DataPower certificate as the **Identity Provider Certificate** such that the Force.com platform can validate the signature associated with the SAML assertion response.

3. Use the self-signed or CA-signed certificate generated in **Step 2** as **Signing Certificate**. Enter the appropriate values in the **Issuer** and **Entity Id** fields. Use the DataPower SSO login URL (such as `https://<dp-host:port>/sso/salesforce/saml/request`) in the **Identity Provider Login URL** field such that Force.com can redirect the request to the identity provider's SSO login URL. The SAML assertion used in this tutorial is not encrypted and contains the **Federation ID** of the login user in the NameIdentifier (NameID) field in the Subject statement.

4. Once you save the SAML SSO settings, note down the Salesforce login URL, such as `https://dipak-sdfc-dev-ed.my.salesforce.com?so=00DF0000000ghLC`, as shown in Figure 9 and Figure 10 above.

**Step 5: Update the Login Page Branding at My Domain**

Go to **Setup > Domain Management > My Domain**, edit **Login Page Branding** and check the appropriate **Authentication Service**. This **Login Page Branding** setup will ensure the display of the SSO login options at the domain specific login page as shown in Figure 11 and Figure 12.

**Figure 11. Login Page Branding at My Domain**

![Login Page Branding](image)
Configure DataPower as a single sign-on identity provider

This section describes the steps to develop the necessary DataPower artifacts that primarily consist of an HTTPS Front Side Handler and a Multi-Protocol Gateway, including the required processing policy rules and AAA policies.

**Step 6: Manage the DataPower crypto key**

1. Generate a crypto key and a self-signed (or CA-signed) certificate using the DataPower crypto tools, which are required for a secured HTTP (SSL) connection and signature generation (see Figure 13).
2. Download the DataPower certificate from the temporary:// directory once you create it (Figure 14). If you have already implemented any of the examples in this tutorial series, you may reuse the same crypto configuration. Make sure you configure a crypto validation credentials, which includes the Salesforce certificate downloaded in Step 2.
Figure 14. DataPower crypto certificate used in this example

![Configure Crypto Certificate](image)

3. Create a crypto certificate and import the Salesforce certificate downloaded in Step 2, which is required for this example to validate the signature in the SAML authentication request (Figure 15).

Figure 15. Configure crypto certificate with importing the Salesforce certificate

![Configure Crypto Certificate](image)

4. Create a crypto identification credentials and add the DataPower crypto key (Figure 16), which is required for the SSL setup.
Figure 16. Configure crypto identification credentials adding the DataPower crypto key

5. Create a crypto validation credentials and add the Salesforce certificate as shown in Figure 17. You may need to add client certificates, such as a browser, if you enforce the SSL client certificate validation.

Figure 17. Configure crypto validation credentials

6. Create a crypto profile. Add the crypto identification credentials and crypto validation credentials (Figure 18).
Figure 18. Configure the crypto profile

7. Create an SSL Proxy Profile with the reverse SSL direction (Figure 19) and use the crypto profile created in the previous step.

Figure 19. Configure the SSL proxy profile

Step 7: Create an HTTPS Front Side Handler

Create an HTTPS Front Side Handler to accept the SSO requests from the browser (Figure 20).
Step 8: Create a Multi-Protocol Gateway

Create a Multi-Protocol Gateway with dynamic back-end routing (Figure 21). Also make sure you select **Non-XML** for both the request and response data type to deal with the HTTP form data as well as the HTML data. The next section, Step 9, describes the Multi-Protocol Gateway policy details.

Step 9: Create a Multi-Protocol Gateway Policy

Create a Multi-Protocol Gateway Policy with following processing rules.

- **First rule**: This rule deals with **favicon.ico** request from the browser (Figure 22).
  - **Direction**: Client to server.
1. **Action**: Match  
**Matching Type**: URL  
**URL Match**: /favicon.ico

2. **Action**: Advanced [Set Variable]  
**Variable Name**: var://service/mpgw/skip-backside  
**Variable Assignment**: 1

3. **Action**: Result

**Figure 22. Multi-Protocol Gateway Policy rule to process favicon.ico request from the browser**

• **Second rule**: This rule processes the SSO login requests from the service provider, which are redirected through the browser using an HTTP POST binding. This rule performs the following sequence of activities:
  1. Receives the HTTP POST request from the service provider through the browser, which contains the HTTP form data with the following parameters:
     • **RelayState**: This is the original resource **URI** requested by the user for access from the service provider.
     • **SAMLRequest**: This is the base-64 encoded SAML authentication request.
  2. Parses the HTTP form data and extracts the above mentioned parameters.
  3. Decodes the (base-64) SAML authentication request and extracts the Salesforce login URL from the `/AuthnRequest/@AssertionConsumerServiceURL` attribute.
  4. Validates the signature extracted from the SAML authentication request XML.
  5. Authenticates the SAML request **Issuer** ([https://dipak-sdfc-dev-ed.my.salesforce.com](https://dipak-sdfc-dev-ed.my.salesforce.com) in this example) by using an AAA policy.
  6. Upon successful authentication of the SAML request **Issuer**, it authenticates the user by using another AAA policy and generates a SAML assertion XML.
  7. Removes the **@NotBefore** attribute from the `<SubjectConfirmationData>` element as the Salesforce SAML Assertion Validator Framework does not allow this attribute.
  8. Signs the SAML assertion.
9. Generates an HTML page. Refer to the specific *TransformAction* for details.
10. Responds back to the browser.

- **Rule configuration details:**
  
  **Direction:** Client to server (Figure 23).

**Figure 23. Multi-Protocol Gateway Policy rule to process SSO login request from the service provider**

<table>
<thead>
<tr>
<th>Configure Multi-Protocol Gateway Style Policy</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Policy:</strong></td>
</tr>
<tr>
<td>Admin Policy</td>
</tr>
<tr>
<td><strong>Rule:</strong></td>
</tr>
<tr>
<td>Create rule: Click New, drag action icon onto line. Edit rule: Click on rule, double-click an action</td>
</tr>
<tr>
<td><strong>Create New Rule:</strong></td>
</tr>
<tr>
<td><strong>Order</strong></td>
</tr>
<tr>
<td>1.</td>
</tr>
<tr>
<td>2.</td>
</tr>
</tbody>
</table>

**Note:** Use an appropriate URL as per your organization’s standard.

1. **Action:** Match
   
   **Matching Type:** URL
   
   **URL Match:** /sso/salesforce/saml/request
   
   **Note:** Use an appropriate URL as per your organization’s standard.

2. **Action:** Convert Query Params to XML (Advanced Action) – see Figure 24

**Figure 24. Action: Convert Query Params to XML**

3. **Action:** Transform
   
   This action decodes (base-64) the SAML authentication request and extracts the Salesforce login URL from the /AuthnRequest/@AssertionConsumerServiceURL attribute. Listing 1 shows the XSLT used for this transformation.
Listing 1. XSLT [process-saml-auth-request.xsl]

```xml
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" extension-element-prefixes="dp"
    exclude-result-prefixes="dp" xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    xmlns:dp="http://www.datapower.com/extensions">
    <xsl:output method="xml" version="1.0" encoding="UTF-8" indent="yes"/>
    <xsl:template match="/">
        <xsl:variable name="vRelayState"
            select="/*[local-name()='request']/*[local-name()='args']/*[local-name()='arg' and @*['local-name()='name' and normalize-space(.) = 'RelayState']]/text()"/>
        <dp:set-variable name="var://context/saml/auth-request/RelayState"
            value="$vRelayState"/>
        <xsl:variable name="vSAMLRequest"
            select="/*[local-name()='request']/*[local-name()='args']/*[local-name()='arg' and @*['local-name()='name' and normalize-space(.) = 'SAMLRequest']]/text()"/>
        <xsl:variable name="vSAMLRequestXML" select="dp:parse($vSAMLRequest,'base-64')"/>
        <dp:set-variable name="var://context/saml/auth-request/sdfcLoginURL"
            value="$vSAMLRequestXML/*[namespace-uri()='urn:oasis:names:tc:SAML:2.0:protocol' and local-name()='AuthnRequest']@AssertionConsumerServiceURL"/>
        <xsl:copy-of select="$vSAMLRequestXML"/>
    </xsl:template>
</xsl:stylesheet>
```

Figure 25. Action: Transform

4. **Action**: Verify signature.
   This action verifies the signature from the SAML authentication request (Figure 26).
5. **Action:** AAA. Create an AAA policy to authenticate the SAML authentication request **Issuer** (Figure 27). Refer to **Step 10** for the AAA configuration details.

**Figure 27. Action: AAA (for SAML authentication request Issuer authentication)**

6. **Action:** AAA.
   Create another AAA policy to authenticate the user and to generate the SAML assertion XML (Figure 28). Refer to **Step 11** for the AAA configuration details.

**Figure 28. Action: AAA (for user authentication and SAML assertion generation)**
7. **Action:** Transform.
   Use the provided XSLT in the Download section of the tutorial ([remove-specific-attribute.xsl](#)) to remove the @NotBefore attribute from the `<SubjectConfirmationData>` element as the Salesforce SAML Assertion Validator framework does not allow this attribute (Figure 29).

**Figure 29. Use XSLT in Transform action to remove an attribute from the SAML assertion XML**

![Configure Transform Action](image)

**Listing 2. XSLT [remove-specific-attribute.xsl] to remove a specific attribute from the SAML assertion XML**

```xml
<?xml version="1.0" encoding="utf-8"?>
<xsl:stylesheet version="1.0" extension-element-prefixes="dp"
    exclude-result-prefixes="dp"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    xmlns:dp="http://www.datapower.com/extensions">
  <xsl:output method="xml" version="1.0" encoding="UTF-8" indent="no"/>
  <xsl:strip-space elements="*"/>
  <xsl:template match="/">
    <xsl:copy>
      <xsl:apply-templates select="/"/>
    </xsl:copy>
  </xsl:template>
  <xsl:template match="*[namespace-uri()='urn:oasis:names:tc:SAML:2.0:protocol' and
                   local-name()='Response']/*[namespace-uri()='urn:oasis:names:tc:SAML:2.0:assertion' and
                   local-name()='Assertion']/*[namespace-uri()='urn:oasis:names:tc:SAML:2.0:assertion'
                   and local-name()='Subject']/*[namespace-uri()='urn:oasis:names:tc:SAML:2.0:assertion'
                   and local-name()='SubjectConfirmation']/*[namespace-uri()='urn:oasis:names:tc:SAML:2.0:assertion'
                   and local-name()='SubjectConfirmationData']/@NotBefore"/>
</xsl:stylesheet>
```

8. **Action:** Sign.
   As shown in Figure 30, this action digitally signs the SAML Assertion XML (XPATH: `/*[namespace-uri()='urn:oasis:names:tc:SAML:2.0:protocol' and local-name()='Response']`) by using the DataPower crypto key and certificate generated in **Step 6.**
Figure 30. Sign SAML Assertion XML

9. **Action:** Transform.  
This **Transform Action** generates an HTML page, which includes an HTML FORM with an HTTP POST Action, and contains a base-64 encoded SAML Assertion XML and RelayState as hidden parameters. Figure 31 shows a sample HTML page responded by DataPower after successful authentication. Use the XSLT (**SSO-Generate-HTML-SP.xsl**) shown in Listing 3 and also provided in the **Download** section of this tutorial.

Figure 31. XSLT transformation to generate HTML response

Listing 3. XSLT [SSO-Generate-HTML-SP.xsl] to generate HTML page

```xml
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" extension-element-prefixes="dp"
    exclude-result-prefixes="dp" xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
>`
Implementing Salesforce federated single sign-on with WebSphere DataPower, Part 3: Implementing a service provider initiated SSO using a signed SAML assertion

```xml
<xs:variable name="relayState" select="dp:variable('var://context/saml/auth-request/RelayState')"/>
<xs:variable name="sdfcLoginURL" select="dp:variable('var://context/saml/auth-request/sdfcLoginURL')"/>
<xs:variable name="signedSamlSerialized">
  <dp:serialize select="/" omit-xml-decl="yes" />
</xs:variable>
<xs:variable name="base64Samldata" select="dp:encode($signedSamlSerialized, 'base-64')" />
<html>
<head>
<title>Login to Salesforce</title>
<meta http-equiv="Cache-Control" content="no-cache" />
<meta http-equiv="Pragma" content="no-cache" />
<meta http-equiv="Expires" content="-1" />
<meta http-equiv="X-UA-Compatible" content="IE=8" />
<style type="text/css">
  .body {background-color: #3EB1D0; }
  .data { background-color: #C6EDEC; margin-top: 10%; margin-left: 20%; margin-right: 20%; font-family: verdana; color: #0A6DA8; }
  .headline { font-size: 50px; }
  .button { background-color: #F5DAA3; color: #0D75AA; font-size: x-large; }
</style>
</head>
<body class="body">
<div align="center" class="data">
  <div>
    <form method="POST" enctype="application/x-www-form-urlencoded">
      <xsl:attribute name="action">
        <xsl:value-of select="$sdfcLoginURL" />
      </xsl:attribute>
      <div class="headline">
        <i><b>Salesforce Login</b></i>
      </div>
      <div><h2>Welcome <span><i><b>
        <xsl:value-of select="dp:variable('var://context/WSM/identity/username')" /></b></i>
        </span></h2></div>
      <div>
        <h4>
          <p>Clicking Submit button will redirect you to Salesforce home page</p>
          <p>Please contact <a href="mailto:support@example.com">support@example.com</a> if you experience any issues during Salesforce Login</p>
        </h4>
      </div>
      <div>
        <input type="hidden" name="idpConfig.recipient">
        <xsl:attribute name="value">
          <xsl:value-of select="$sdfcLoginURL" />
        </xsl:attribute>
        <input type="hidden" name="SAMLResponse">
        <xsl:attribute name="value">
          <xsl:value-of select="$base64Samldata" />
        </xsl:attribute>
      </div>
    </form>
  </div>
</div>
</body>
</html>
```
10. **Action:** Advanced [Set Variable] to skip a real back-end routing (Figure 32).

   **Variable Name:** var://service/mpgw/skip-backside

   **Variable Assignment:** 1

   **Figure 32. Advanced Action to skip back-end routing**

   ![Configure Set Variable Action](image)

11. **Action:** Result (Figure 33)

    **Figure 33. Result action**

    ![Configure Results Action](image)
Step 10: Create an AAA policy to authenticate the SAML request issuer

This AAA policy extracts the Issuer from the SAML authentication request as a custom token:

```xml
```

It authenticates against an AAA Policy Info file shown in Listing 4. Figures 34 to 36 show the AAA Policy configuration details.

Listing 4. Sample AAA Policy Info file

```xml
<?xml version="1.0" encoding="utf-8"?>
<AAAInfo xmlns="http://www.datapower.com/AAAInfo">
  <FormatVersion>1</FormatVersion>
  <Filename>local:///AAAInfoPolicy.xml</Filename>
  <Summary>AAA Policy Info XML file</Summary>
  <Authenticate>
    <Username>dipakpal</Username>
    <Password>dipakpal-password</Password>
    <OutputCredential>dipakpal@sso.sdfc.com</OutputCredential>
  </Authenticate>
  <Authenticate>
    <Username>sdfc.user</Username>
    <Password>sdfc-password</Password>
    <OutputCredential>sdfc.user@sso.sdfc.com</OutputCredential>
  </Authenticate>
  <Authenticate>
    <CustomToken>https://dipak-sdfc-dev-ed.my.salesforce.com</CustomToken>
    <OutputCredential>saml.auth.request.issuer</OutputCredential>
  </Authenticate>
  <Authenticate>
    <DN>CN=Alice, O=DataPower, L=Cambridge, ST=MA, C=US</DN>
    <OutputCredential>admin</OutputCredential>
  </Authenticate>
  <Authenticate>
    <IPHost>127.0.0.1</IPHost>
    <OutputCredential>admin</OutputCredential>
  </Authenticate>
  <Authenticate>
    <IPHost>::1</IPHost>
    <OutputCredential>admin</OutputCredential>
  </Authenticate>
  <Authenticate>
    <Username>guest</Username>
    <Password>guest</Password>
    <OutputCredential>user</OutputCredential>
  </Authenticate>
  <Authorize>
    <InputCredential>dipakpal@saml.sdfc.com</InputCredential>
    <InputResource>/sso/*</InputResource>
    <Access>allow</Access>
  </Authorize>
  <Authorize>
    <InputCredential>sdfc.user@sso.sdfc.com</InputCredential>
    <InputResource>/sso/*</InputResource>
    <Access>allow</Access>
  </Authorize>
  <Authorize>
    <InputCredential>saml.auth.request.issuer</InputCredential>
    <InputResource>/sso/*</InputResource>
    <Access>allow</Access>
  </Authorize>
</AAAInfo>
```
<Authorize>
<InputCredential>guest</InputCredential>
<InputResource>/sso/salesforce</InputResource>
<Access>deny</Access>
</Authorize>

<Authorize>
<InputCredential>admin</InputCredential>
<InputResource>/sso/salesforce</InputResource>
<Access>allow</Access>
</Authorize>

<Authorize>
<InputCredential>
  <Version>3</Version>
  <SerialNumber>0</SerialNumber>
  <SignatureAlgorithm>sha1WithRSAEncryption</SignatureAlgorithm>
  <Issuer>C=US, ST=MA, L=Cambridge, O=DataPower, CN=Alice</Issuer>
  <NotBefore>2002-11-23T01:15:33Z</NotBefore>
  <NotAfter>2012-11-23T01:15:33Z</NotAfter>
  <Subject>C=US, ST=MA, L=Cambridge, O=DataPower, CN=Alice</Subject>
  <SubjectPublicKeyAlgorithm>rsaEncryption</SubjectPublicKeyAlgorithm>
  <Extensions/>
</InputCredential>
<InputResource>/sso/*</InputResource>
<Access>allow</Access>
</Authorize>

</AAAInfo>

Figure 34. AAA: Extract a custom token from SAML authentication request XML
Figure 35. AAA: Authenticate the SAML authentication request Issuer against AAA Policy Info XML

![Diagram](image1)

Figure 36. AAA: Authorize any authenticated client

![Diagram](image2)

Step 11: Create an AAA policy to authenticate the user

1. Create an AAA policy to authenticate the user and to generate an SAML assertion XML. Examples illustrated by this tutorial use a HTTP basic authentication method and validate user credentials against a DataPower AAA policy info XML shown in Listing 4. However, you can implement an appropriate authentication method as per your organization’s standard. Figures 37, 38, and 39 show the AAA policy configuration details. If you have already implemented any of the examples illustrated by the previous tutorials in this series, you can reuse the same AAA policy.
Figure 37. AAA policy – Extract identity from request

Figure 38. AAA policy – Authenticate the user
2. Configure this AAA access control policy to generate an SAML assertion response upon successful authentication. Figure 40 shows an AAA Access control policy page enabling an SAML assertion. Make sure you use the same SAML Issuer (such as https://datapowersso.ibm.com/) configured at the Salesforce SSO settings (Step 4).

Figure 40. AAA Policy – Generate SAML assertion response

3. Set the Entity ID value from the Salesforce SSO settings (Step 4) in the SAML Audience field in the AAA Access Control Policy page. Use the Salesforce login URL for both the SAML Recipient and Response Destination fields. Create the SAML attribute definitions as shown in Figure 41.
Step 12: Test the single sign-on

1. Enter a domain specific service provider, such as Salesforce, resource URL, or login URL in the web browser as shown in Figure 42. This example uses a book-marked URL that displays list of users at Force.com. For example:

   Figure 42. Salesforce login page (domain specific)

2. Make sure the Salesforce resource or login URL is domain specific. Clicking the appropriate SSO identity provider option (DataPower SSO for this example) on the login page redirects the request (HTTP POST) to DataPower. DataPower validates the SAML authentication request and then challenges the user for the login credentials if the user is not already logged in (Figure 43).
3. Upon successful authentication, DataPower responds back with a SSO re-direct (HTTP POST binding) page as shown in Figure 44, which includes an HTML FORM with HTTP POST Action and contains a base-64 encoded SAML assertion XML and RelayState as hidden parameters. It requires the user to submit the HTML page by clicking the Login to Salesforce button, which results into submitting an HTTP POST request to the Salesforce login URL (for example, https://dipak-sdfc-dev-ed.my.salesforce.com?so=00DF0000000ghLC).

4. Salesforce validates message integrity using the embedded signature in the SAML assertion XML against the DataPower certificate, which is already uploaded during the SSO setup process. Upon successful signature validation, it processes the SAML assertion statement, extracts and validates the Federation ID, and finally redirects to the page originally requested. Figure 45 shows the Salesforce landing page after a successful SSO login attempt by this example.
Step 13: Debug and troubleshoot guide

Assuming you are familiar with DataPower common debugging and troubleshooting methods, this tutorial will primarily focus on Salesforce.

You can capture the base-64 encoded SAML authentication request using Developer Tools in Google Chrome as shown in Figure 46 and Figure 47. Though you can get this data from the HTML page source, you can use Developer Tools at Google Chrome or Internet Explorer to analyze HTTP events, including their request/response header and data structure. You can decode (base-64) this data by using any online tool, such as http://www.string-functions.com/base64decode.aspx, to get the SAML authentication request XML. Listing 5 shows a sample SAML authentication request XML. Listing 6 shows a sample SAML authentication response.
Implementing Salesforce federated single sign-on with WebSphere DataPower, Part 3: Implementing a service provider initiated SSO using a signed SAML assertion

Figure 46. Start Developer Tools at Google Chrome

Figure 47. HTTP form data of DataPower SSO login request shown by Developer Tools

Listing 5. Sample of signed SAML authentication request

```xml
<samlp:AuthnRequest
    AssertionConsumerServiceURL="https://dipak-sdfc-dev-ed.my.salesforce.com?
so=0D0F00000000ghLC"
    Destination="https://100.99.99.80/sso/salesforce/saml/request"
    ID="_2RWCvoeG0FNhRmMRwdlJnZzhVKmgME5u1MUEJgguj_5CpqN_8xvDmqUF fingerprints=685f85425f1c44df929f9e22f952f85e12503434b47f1a7f881fedba4b713e835"
    IssueInstant="2014-07-28T03:36:23.875Z"
    ProtocolBinding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST"
    Version="2.0"
    xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol">
    <saml:Issuer xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"/>
    <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
        <ds:SignedInfo>
            <ds:CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
            <ds:SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
            <ds:Reference URI="/request"/>
            <ds:Transforms/>
        </ds:SignedInfo>
    </ds:Signature>
</samlp:AuthnRequest>
```
Implementing Salesforce federated single sign-on with WebSphere DataPower, Part 3: Implementing a service provider initiated SSO using a signed SAML assertion

Figure 48. Capture base-64 encoded SAML assertion XML from HTML page source

Listing 6. Sample of signed SAML authentication response

<saml2:Response Version="2.0" ID="SAML-46528e40-71c1-4065-be36-3f3b041a9c93"
IssueInstant="2014-07-28T03:35:23Z"
InResponseTo="2_RwCvoie6F9NhMmRwdi1JnZzhVkmgMESu1MUEjJgqju_jCpqN_8xvDmQxUF_fdnuF_uw8BMVwvnhA/yHs6Sh25dCPro5G3hnTEMLCD6GPoPbaUus21uqOFCcOQwDSQrVI4A4Xh51ftntpta2AmJMX1687011b2ugUZ9F
KxRM1fYffvFZber6LQX4enbqitN20qngp.k1_jZFsQgq70o2pwxwC_btwIA"
Destination="https://dipak-sdfc-dev-ed.my.salesforce.com?so=00DF0000000ghLC"
xmlns:saml2="urn:oasis:names:tc:SAML:2.0:assertion"
xmlns:samlp2="urn:oasis:names:tc:SAML:2.0:protocol">
<samlp2:Issuer
xmlns:samlp2="urn:oasis:names:tc:SAML:2.0:protocol">
https://datapowersso.ibm.com/</samlp2:Issuer>

https://dipak-sdfc-dev-ed.my.salesforce.com
Implementing Salesforce federated single sign-on with WebSphere DataPower, Part 3: Implementing a service provider initiated SSO using a signed SAML assertion
Login to Salesforce as administrator and validate the base-64 encoded SAML assertion to check whether the SSO identity provider (DataPower) generates the correct SAML assertion as expected by Salesforce. Figures 49 to 52 show the SAML assertion validation using the Force.com SAML Assertion Validator Framework.

**Figure 49. Start the SAML Assertion Validator Framework at Salesforce**

![Start the SAML Assertion Validator Framework at Salesforce](image)

**Figure 50. Enter base-64 encoded SAML assertion into the Salesforce SAML Validator**

![Enter base-64 encoded SAML assertion into the Salesforce SAML Validator](image)
Implementing Salesforce federated single sign-on with WebSphere DataPower, Part 3: Implementing a service provider initiated SSO using a signed SAML assertion

While testing the SSO login, you can monitor the login history at Force.com from a separate login as the Salesforce administrator. Figure 53 shows a login history page at Force.com site, which can help you troubleshoot the SSO login failures.
Conclusion

Organizations that have fewer Salesforce users may not require a single sign-on solution as they can continue to use the Salesforce standard login process. However, organizations with a large number of Salesforce users can leverage this solution by implementing a DataPower based single sign-on infrastructure for all their internal applications as well as cloud-based and external applications (for example, Force.com), which support Federated Identity Management using an external SSO identity provider.

To continue with this series, see the following tutorials:

- Part 1: Identity provider initiated SSO using a signed SAML assertion
- Part 2: Identity provider initiated SSO using an encrypted and signed SAML assertion
- Part 4: Service provider initiated SSO using an encrypted and signed SAML assertion
## Downloads

<table>
<thead>
<tr>
<th>Description</th>
<th>Name</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Code sample</td>
<td>code_sample_part3.zip</td>
<td>8KB</td>
</tr>
</tbody>
</table>
Resources

- Single Sign On (SSO) Basics
- SAML 2.0 Basics
- Single Sign On with SAML on Force.com
- IBM WebSphere DataPower Knowledge Center
- Hands-on Training: Enable Single Sign-on with SAML and Salesforce Identity
- Single Sign-On Best Practices (Salesforce)
- Setting up Single Sign-On (Salesforce)
- developerWorks WebSphere zone
- developerWorks WebSphere DataPower discussion forum
About the author

Dipak K. Pal

Dipak K. Pal works as an Integration Designer in the IBM Global Business Services team. He has more than 10 years of IT experience in enterprise integration, cloud integration, service-oriented architecture, and J2EE-based application design and development. He is currently working on a cloud integration project, integrating several cloud applications with various on-premise applications using IBM WebSphere Cast Iron, IBM Integration Bus, and IBM WebSphere DataPower.

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