Implementing Salesforce federated single sign-on with WebSphere DataPower, Part 4: Service provider initiated single sign-on using an encrypted and 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. This last part of the tutorial series, Part 4, discusses how to implement the service provider initiated single sign-on to Salesforce using an encrypted and signed SAML assertion.

Introduction

Part 4 of this tutorial series does not recap the fundamentals of the Salesforce federated single sign-on (SSO). This tutorial primarily discusses how to implement the service provider initiated SSO to Salesforce using IBM® WebSphere DataPower (hereafter called DataPower) as an identity provider, powered by an encrypted and signed SAML assertion. Figure 1 and Figure 2 show a high level overview of this solution.

Figure 1. Overview of service provider initiated SSO login to Salesforce
Single sign-on setup at Force.com platform

This section provides the steps to implement a single sign-on setup for the Force.com platform. If you have already implemented an example illustrated by 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) at 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 the Salesforce developer site

Login to Salesforce by entering the credentials obtained during the sign-up process and create another user account. Make sure you enter a federation ID as shown in Figure 4 to enable the SSO login for this user account.

User ID: sdfc.user@sdfc.com
Federation ID: sdfc.user@sso.sdfc.com
Figure 4. Enter a federation ID to enable SSO

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

Certificates are used for authenticated single sign-on 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. In this example, the identity provider additionally encrypts the SAML assertion response by using the Salesforce certificate. 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) 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 the 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/.
Step 4: Set up SSO settings at Salesforce

1. Login as 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 a DataPower crypto key and the associated certificate described in Step 6.
2. Upload the DataPower certificate as an **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** for both the **Signing Certificate** and **Assertion Decryption Certificate** fields. Enter the appropriate values in the **Issuer** and **Entity Id** fields. Use the DataPower SSO login URL (for example, `https://<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 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 (for example, `https://dipak-sdfc-dev-ed.my.salesforce.com?so=00DF0000000ghLC&sc=0LEF0000000Kytb`).

**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 displays the SSO login options on the domain specific login page, as shown in Figure 11 and Figure 12.

**Figure 11. Login Page Branding at My Domain**
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 policies 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 (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 13. DataPower crypto key used in this example
3. Create a crypto certificate and import the Salesforce certificate downloaded in Step 2, which is required to validate the signature in the SAML authentication request and to encrypt the SAML assertion response (Figure 15).

**Figure 15. Configure crypto certificate with importing the Salesforce certificate**

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

**Figure 16. Configure the crypto identification credentials**
5. Create a crypto validation credentials and add the Salesforce certificate as shown in Figure 17. You may need to add a client (such as a browser) certificates if you enforce the SSL client certificate validation.

**Figure 17. Configure the crypto validation credentials**

![Configure Crypto Validation Credentials](image)

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

**Figure 18. Configure a crypto profile**

![Configure Crypto Profile](image)

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 (Figure 20) to accept the SSO requests from the browser.

Figure 20. Configure a HTTPS Front Side Handler

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 an HTTP form data as well as an HTML data. The next section, Step 9, describes the Multi-Protocol Gateway policy details.
Figure 21. Configure a Multi-Protocol Gateway

![Multi-Protocol Gateway Configuration](image)

**Step 9: Create a Multi-Protocol Gateway Policy**

Create a Multi-Protocol Gateway policy with the following processing rules.

- **First rule**: This rule deals with the *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
Second Rule: This rule processes SSO login requests from the service provider, which are redirected through the browser using the HTTP POST binding. This rule performs the following sequence of activities:

1. Receives HTTP POST request from the service provider through the browser, which contains 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 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 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 an 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. Encrypts the SAML assertion by using the Salesforce certificate.
9. Signs the SAML assertion.
10. Generates an HTML page. Refer to the specific Transform action for details.
11. Responds back to the browser.

The rule configuration details are:

**Direction:** Client to server (Figure 23).
**Figure 23. Multi-Protocol Gateway Policy rule to process SSO login request from the service provider**

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

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

3. **Action:** Transform (Figure 25)  
   This action decodes the (base-64) 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"
xmns: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="$vRelayState"/>
<xsl:variable name="vSAMLRequest" select="/*[local-name()='request']/*[local-name()='args']/*[local-name()='arg'
and @*[local-name()='name' and normalize-space(.) = 'SAMLRequest']/text()"/>
<dp:set-variable name="vSAMLRequestXML" select="dp:parse($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 (Figure 26)**
This action verifies the signature from the SAML authentication request.

Figure 26. Action: Verify

5. **Action: AAA (Figure 27)**
Create an AAA policy to authenticate the SAML authentication request **issuer**. Refer to **Step 10** for the AAA configuration details.
6. **Action:** AAA (Figure 28)
   Create another AAA policy to authenticate the user and to generate the SAML assertion XML. Refer to Step 11 for the AAA configuration details.

**Figure 28. Action: AAA (user authentication and SAML assertion generation)**

7. **Action:** Transform (Figure 29)
   Use the provided XSLT (remove-specific-attribute.xsl) as shown in Listing 2, to remove the `@NotBefore` attribute from the `<SubjectConfirmationData>` element as the Salesforce SAML Assertion Validator framework does not allow this attribute.
Figure 29. Use XSLT in Transform action to remove an attribute from the SAML assertion XML

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="@* | node()">
  <xsl:copy>
    <xsl:apply-templates select="@* | node()"/>
  </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:** Encrypt (Figure 30)

This action encrypts the SAML assertion by using the Salesforce certificate. Create a Document Crypto Map specifying the following XPATH (Listing 3), as shown in Figure 31.

**Listing 3. XPATH used at Document Crypto Map**

```xml
/*[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"/>
```
Figure 30. Encrypt SAML assertion

9. **Action:** Sign (Figure 32)
   This action digitally signs the encrypted 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 the certificate generated in Step 6.

Figure 31. Create a Document Crypto Map for encryption
Implementing Salesforce federated single sign-on with WebSphere DataPower, Part 4: Service provider initiated single sign-on using an encrypted and signed SAML assertion

**Figure 32. Sign SAML assertion XML**

![Figure 32](image)

10. **Action**: Transform (Figure 33)

   This action generates an HTML page, which includes an HTML FORM with an HTTP POST Action and contains the base-64 encoded SAML assertion XML and RelayState as hidden parameters. Figure 33 shows a sample HTML page responded by DataPower after successful authentication. Use the XSLT ([SSO-Generate-HTML-SP.xsl](#)) shown in Listing 4 and also provided with this tutorial in the Download section.

**Figure 33. XSLT Transformation to generate HTML response**

![Figure 33](image)

**Listing 4. 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 4: Service provider initiated single sign-on using an encrypted and signed SAML assertion
11. **Action:** Advanced [Set Variable] to skip a real back-end routing (Figure 34).

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

**Variable Assignment:** 1

**Figure 34. Advanced action to skip back-end routing**

12. **Action:** Result (Figure 35)

**Figure 35. Result action**
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 and authenticates against an AAA Policy Info file shown in Listing 5:

```
XPATH: //*[namespace-uri()='urn:oasis:names:tc:SAML:2.0:protocol' and local-name()='AuthnRequest']
     /*[namespace-ri()='urn:oasis:names:tc:SAML:2.0:assertion' and local-name()='Issuer']/text()
```

Figures 36, 37, and 38 show the AAA policy configuration details.

Listing 5. 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 36. AAA: Extract a custom token from SAML authentication request XML
Step 11: Create an AAA policy to authenticate the user

1. Create another AAA policy to authenticate the user and to generate an SAML assertion XML. The example illustrated by this tutorial uses a basic HTTP authentication method and validates the user credentials against a DataPower AAA policy info XML shown in Listing 5. However, you can implement an appropriate authentication method per your organization’s standard. Figure 39 to 41 show the AAA policy configuration details. If you have already implemented any of the examples illustrated by the previous tutorials of this series, you can reuse the same AAA policy.
Implementing Salesforce federated single sign-on with WebSphere DataPower, Part 4: Service provider initiated single sign-on using an encrypted and signed SAML assertion

Figure 39. AAA Policy – Extract identity from request

Figure 40. AAA Policy – Authenticate the user
2. Configure this AAA Access Control Policy to generate an SAML assertion response upon successful authentication. Figure 42 shows an AAA Access Control Policy page enabling an SAML assertion. Make sure you use the same SAML issuer (for example, https://datapowersso.ibm.com/) configured at the Salesforce SSO settings (Step 4).

Figure 42. 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.

4. Use the Salesforce login URL for both SAML Recipient and Response Destination fields.

5. Create the SAML attribute definitions as as shown in Figure 43.
Step 12: Test the single sign-on

1. Enter a domain specific service provider (for example, Salesforce) resource URL or a login URL in the web browser as shown in Figure 44. This example uses the following bookmarked URL that displays a list of users at Force.com:

   https://dipak-sdfc-dev-ed.my.salesforce.com/005?retURL=%2Fui%2Fsetup%2FSetup%3Fsetupid%3D0Users
   &setupid=ManageUsers

2. Make sure the Salesforce resource or login URL is domain specific. Clicking the appropriate SSO identity provider (DataPower SSO for this example) option 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 (Figure 45) if the user is not already logged in.
3. Upon successful authentication, DataPower responds back with a SSO re-direct (HTTP POST binding) page as shown in Figure 46, which includes an HTML FORM with an 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=00DF0000000ghL&sc=0LEF0000000Kytb).

4. Salesforce validates the 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 decrypts the encrypted assertion, processes the SAML assertion statement, extracts and validates the federation ID, and finally redirects to the page originally requested. Figure 47 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 48 and Figure 49. Although you can get this data from the HTML page source, using **Developer Tools** or **Internet Explorer** will help you analyze HTTP events, including their request/response header and data structure. You can decode (base-64) this data by using any online tool (for example, [http://www.string-functions.com/base64decode.aspx](http://www.string-functions.com/base64decode.aspx)) to get an SAML authentication request XML. Listing 6 shows a sample SAML authentication request XML.
Implementing Salesforce federated single sign-on with WebSphere DataPower, Part 4: Service provider initiated single sign-on using an encrypted and signed SAML assertion

Figure 48. Start Developer Tools in Google Chrome

Figure 49. HTTP Form data of DataPower SSO login request shown by Developer Tools

Listing 6. Sample signed SAML authentication request

```xml
<samlp:AuthnRequest
  AssertionConsumerServiceURL="https://dipak-sdfc-dev-ed.my.salesforce.com?so=0D0F0000000ghL&sc=0LEF0000000Kytb"
  Destination="https://100.99.99.80/sso/salesforce/saml/request"
  ID="_2_qtBx5SCYoFeRPM783dL0kCopCzg0G1JGFCNYWjziC_kcRnB_jkteElIwYxXMSmn8Phcmpv8e_xvZel9S
  Q_AoHf.ebf1pLaAmMX4wiij1MFvUtsMOHdby2kWjFfrFFKg87TGUQoXIw5ASSQ9Nxx0escBvg.Nzxp.sQ0a24rHa
  XK0l7K31Uamo.qzWP.2vYvGpP.kj_iZFFs9ig70q2pW6C_xwKpoA"
  IssueInstant="2014-07-28T04:32:35.342Z"
  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">
    https://dipak-sdfc-dev-ed.my.salesforce.com
  </saml:Issuer>
  <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="#_2_qtBx5SCYoFeRPM783dL0kCopCzg0G1JGFCNYWjziC_kcRnB_jkteElIwYxXMSmn8Phcmpv8e_xvZel9S
      Q_AoHf.ebf1pLaAmMX4wiij1MFvUtsMOHdby2kWjFfrFFKg87TGUQoXIw5ASSQ9Nxx0escBvg.Nzxp.sQ0a24rHa
      XK0l7K31Uamo.qzWP.2vYvGpP.kj_iZFFs9ig70q2pW6C_xwKpoA"
      IssueInstant="2014-07-28T04:32:35.342Z"
      ProtocolBinding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST"
      Version="2.0">
        <ds:Transforms>
          <ds:EnvelopedSignature Algorithm="http://www.w3.org/2001/04/xkms-transport-profile#rsa-sha1"/>
          <ds:EnvelopedSignature Algorithm="http://www.w3.org/2001/04/xkms-transport-profile#rsa-sha1"/>
        </ds:Transforms>
      </ds:Reference>
    </ds:SignedInfo>
    <ds:SignatureValue>
      <ds:KeyInfo>
        <ds:X509Data>
          <ds:X509Certificate>-----BEGIN CERTIFICATE-----
            MIIDzDCCAmIBADCCAzOwDAYDQECDQYDVQQIAwExZWRpbmNlcy1wcm9jZWxlc3 quasi...-----END CERTIFICATE-----
          </ds:X509Certificate>
        </ds:X509Data>
      </ds:KeyInfo>
      <ds:Object>
        <ds:KeyValue>
          <ds:Data>
            "-----BEGIN PUBLIC KEY-----
              MIIDzDCCAmIBADCCAzOwDAYDQECDQYDVQQIAwExZWRpbmNlcy1wcm9jZWxlc3 quasi...-----END PUBLIC KEY-----"
          </ds:Data>
        </ds:KeyValue>
      </ds:Object>
    </ds:SignatureValue>
  </ds:Signature>
</samlp:AuthnRequest>
```
You can capture base64 encoded SAML assertion response data using **Developer Tools** in Google Chrome as shown in Figure 50. Listing 7 shows a sample SAML assertion XML.

**Figure 50. Capture base-64 encoded SAML assertion XML from HTML page source**

You can capture base64 encoded SAML assertion response data using **Developer Tools** in Google Chrome as shown in Figure 50. Listing 7 shows a sample SAML assertion XML.

**Listing 7. Sample encrypted and signed SAML authentication response**

```
<saml2p:Response Version="2.0" ID="SAML-55173ac1-93ab-4a69-a7f9-cbf85b0de345"
IssueInstant="2014-07-28T04:31:35Z"
InResponseTo="_2_qtBx5CYOfRePpj783dLkCOpCzgQGl6GcfNyWjJ2c_cKeRnB\.jktE1lWiyXMSmmn8PHcmpv8e_xV2z1sSecQ_Aohf.ebf1pLAmMX4Wjj1MFrVUtSMoHdBY2KwJTufWKFG87GQTj0gXl5As5Q9Nx8esc8vg.Nzxp.sQ0a24rHakXb17CK3LUnary.zqwP.2VtYgPp.kI_j2FpsG1q70o2pwwC_xwKpoA"
>
```
Login to Salesforce as an administrator and validate the base-64 encoded SAML assertion to check whether the SSO identity provider (DataPower) generated the correct SAML assertion as
expected by Salesforce. Figures 51 to 54 show the SAML assertion validation using the Force.com SAML Assertion Validator framework.

**Figure 51. Start SAML Assertion Validator framework at Salesforce**

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

**Figure 52. 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 4: Service provider initiated single sign-on using an encrypted and signed SAML assertion

Figure 53. Successful SAML assertion validation at the Salesforce SAML Validator

Figure 54. Invalid SAML assertion with an expired timestamp
While testing the SSO login, you can monitor the login history on Force.com from a separate login as a Salesforce administrator. Figure 55 shows a login history page on the Force.com site, which can help you troubleshoot SSO login failures.

**Figure 55. Login history page on Force.com**

### Conclusion

Organizations that have fewer Salesforce users may not require a single sign-on solution as they can continue using 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 of 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.

This tutorial, Part 4, concludes the tutorial series on implementing Salesforce federated single sign-on with WebSphere DataPower.
## Downloads

<table>
<thead>
<tr>
<th>Description</th>
<th>Name</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Code sample</td>
<td>code_sample_part4.zip</td>
<td>10KB</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.