Cross-domain single sign-on using SAML 2.0 with WebSphere Liberty, Part 2: Set up a secure hybrid cloud environment with IBM Cloud

Cristiano Mauri
Guido Gilardi

March 14, 2017

This 3-part series, "Cross-domain single sign-on using SAML 2.0 with WebSphere Liberty," introduces an end-to-end single sign-on (SSO) solution that uses IBM Cloud in a hybrid cloud environment. Part 2 shows how to connect your applications on the cloud to your local machine, without needing a public IP address, by using the Secure Gateway. It details how to configure and use the IBM Cloud Tools within your Eclipse workspace to manage servers and applications that run on IBM Cloud.

Introduction

The advantages of developing on the cloud are appealing for many organizations. However, applications that run in a public cloud with access from the internet are exposed to more security risks than applications that run in a private local network.

Part 1 of this series explained how to configure a service provider (SP)-initiated SSO with identity propagation. It showed how to enable Java™ EE security and how to propagate the authenticated subject over different domains. This part, Part 2, explains how to set up a hybrid cloud environment by using IBM® Cloud. You see how to deploy your applications on the IBM Cloud infrastructure by using the IBM Cloud Tools for Eclipse. You configure the IBM Secure Gateway service to allow the cloud applications to invoke the services that run on your on-premises (local computer) network. You implement SSO between all your applications that run in the hybrid cloud by using the SP-initiated SSO model that was described in Part 1.
Before you complete this part, you must read and complete the steps in Part 1. You also need an IBM Cloud account to access your Cloud space.

Try IBM Cloud
Get the Secure Gateway service

Overview of the solution

This part shows how to deploy two of the applications from Part 1—Frontend and CloudServices—on the IBM Cloud space. The following figure illustrates the solution for this part.

To set up the solution, you deploy the Frontend web application and the CloudServices web services, together with their server configurations, to the IBM Cloud space. The LocalServices application and the identity provider (IdP) still run on your local machine because you want to emulate the private services that must not be exposed to the internet.

In this solution, your browser mediates the interaction between the SP and IdP. Therefore, you do not need to expose the IdP to the cloud. Also, because the CloudServices application invokes the LocalServices web services, you must set up communication between the IBM Cloud space and your local machine. That is, you configure the Secure Gateway service on IBM Cloud and the Secure Gateway client on your local machine.

This solution also shows how session affinity works in the cloud. Session affinity is an important aspect of a cloud solution because, in a real scenario, you might want to make your applications to scale, depending on the load.

1. Set up and configure the IBM Secure Gateway solution

Local applications and databases in a corporate intranet are usually unreachable because of security restrictions and firewalls. IBM Secure Gateway solution offers an easy, scalable, and
secure way to access local resources across cloud and on-premises environments in both directions: from cloud to on-premises and from on-premises to cloud.

In this part, you see how to set up and configure the Secure Gateway service and client to allow the CloudServices application to call the on-premises web service that is exposed by the LocalServices application. The following figure illustrates the connection architecture.

The Secure Gateway client establishes a tunnel between the local network and IBM Cloud. Configuring the Secure Gateway solution entails the following steps, which are explained in the remainder of this part:

1. Add a Secure Gateway service instance to your Cloud space.
2. Install the Secure Gateway client in a server that run on the local intranet (local PC).
3. Configure the access control list (ACL) on the client.
4. Add a destination to the Secure Gateway service. A destination is a combination of intranet host names and ports. You must specify a destination, which can be one of the following types, for each target service or database:
   - Non-secure, non-encrypted destination
   - Secure destination with a server-side TLS
   - Destination with mutual authentication TLS options

   The supported protocols are HTTP, HTTPS, TCP, and UDP.

Because the Secure Gateway client opens the tunnel, you do not need a public IP on the machine where you install it. In this tutorial, you install the Secure Gateway client on your local machine.

Also, by using the Secure Gateway, you can control access from the cloud. That is, on the gateway client, you can set up the ACL to select which of your local intranet services you want to expose to IBM Cloud. On the cloud side, you can enforce the security of your local services by configuring which cloud services can access your gateway client. This configuration avoids exposing all your local services to all the cloud applications.

The Secure Gateway service supports high availability configurations. To explore the details, see Managing the Secure Gateway.
1a. Add a Secure Gateway service instance to your IBM Cloud space

1. Open the IBM Cloud console, and open the Secure Gateway service by using one of the following options:
   - Search for Secure Gateway on the main catalog page, and click Secure Gateway.
   - Click Catalog on the right side of the menu at the top of the page:
     a. In the left menu, under Services, click Integrate.
     b. In the body of the page, click Secure Gateway.
2. Click Create to start the wizard that guides you through the creation process.
3. In the Secure Gateway page that opens, scroll down and click Add Gateway.
4. In the Add Gateway wizard, complete the following information:
   a. Enter the name that you want displayed in the Dashboard as the service name, for example: Sample Gateway.
   b. If you want to allow the connection to this gateway only from clients that use a specific token, select Require security token to connect clients.
   c. Select Token Expiration, and select a time for this token.
   d. Click Add Gateway.

You now see the gateway that you created in your Secure Gateway Dashboard.

In this dashboard, you see basic information about your newly created gateway (gray box on the right side):

- **Status.** The two red circles indicate that no client is connected.
- **Number of destinations.** You can have just one gateway that connects to all the on-premises (local) services
- **Gear button.** You can display and export all information that a client needs to connect to the gateway (see the following figure) when you click the gear button:
  - **Security Token.** This token is generated when you select the Require security token to connect clients check box in the Add Gateway wizard to avoid unwanted connections.
  - **Gateway ID.** This string indicates to IBM Cloud which gateway you want to connect to.
• **Regenerate Gateway Cert and Key.** You click this link to refresh the token after its expiration. The default is 90 days.

![Sample Gateway](image)

To configure the client without copying both strings, you can export them into a file.

**1b. Install the Secure Gateway client**

Next, you download, install, and configure the Secure Gateway client on your local machine. Three kinds of clients are available to connect your local network to the IBM Cloud gateways:

- **IBM Installer:** A software client that can be installed on a server to create the tunnel toward IBM Cloud. It's available on the following operating systems:
  - Windows
  - Mac OSX
  - RHEL 6+
  - Ubuntu for Linux on z Systems
  - Ubuntu 14+ PPC
  - Ubuntu 14+
- **Docker:** A Docker image that is built with the same capabilities as the IBM Installer option that can be run anywhere.
- **IBM DataPower:** An appliance that is optimized solution with the same base features as the Docker client, but with security enforced capabilities.

In this tutorial, we use the IBM Installer. To install the Secure Gateway Client software, start from the Secure Gateway Dashboard page:

1. Click your gateway service (large gray button with the name of your gateway).
2. Click **Connect Client.**

   A window opens that shows the three client options. It includes the gateway ID, the security token of your gateway service, and list of executable installers for the supported operating systems in the **Software Installers** list.
3. Select **IBM Installer**.
4. Click the **Download** icon that corresponds to the client for your operating system to download the executable installer. We used a Microsoft Windows 7 Enterprise operating system for this tutorial, but also successfully tested the Mac OS X client.
5. Run the installer to install the client on your target machine. For more information about installation on the supported operating systems, see **Setting up a client**.

You can install the Secure Gateway client on every machine that can connect to IBM Cloud. As mentioned previously, no public IP address is required because the Secure Gateway client will establish a connection to the Cloud space.

If you install the Secure Gateway client on a Microsoft® Windows® machine, a wizard guides you step-by-step, prompting you for basic information about the installation, such as the path and language. Because the installation is simple, we provide the following tips about installing the Secure Gateway client on Windows:

- You can install the client as a Windows Service, avoiding the need to manually start and manage the client application.
- When you are prompted, insert the information about the **Gateway IDs** and **Security tokens**. You can find this information on the **Secure Gateway Dashboard** by clicking the **gear** button. If you need to install the Secure Gateway client on a different operating system, see the **online help**.

- The Secure Gateway client offers a client user interface that is accessible by using a web browser. During the installation of the client for Windows, the wizard prompts you to enter the port to use for the client user interface.
After the installation completes successfully, start the Secure Gateway client. In the Microsoft Windows client, you click **Start -> All Programs -> IBM folder**. If you installed the Secure Gateway client as a Windows Service, you can start it from the **Services** panel.

### 1c. Configure the access control list

To configure the ACL on your Secure Gateway client:

1. Start the Secure Gateway client as explained in the previous section.
2. Open a web browser, and go to `http://localhost:<client-ui-port>/dashboard`, where `<client-ui-port>` is the one that you configured previously. The default is `9003`. The Secure Gateway client console is now displayed.
3. Click **Access Control List**. On the next page, you see two lists: Allow access and Deny access.
4. Add the **host name** and the **port** of the server that run your LocalServices application to the **Allow access list**. If you did not change the configuration from **Part 1**, the host name is `localhost`, and the port is `9445`.
5. Click the plus (`+`) button.

If you installed the LocalServices application on a different machine than the one where you installed the Secure Gateway client, the Secure Gateway client must be able to connect to the LocalServices machine. Otherwise, the web service that is exposed by the LocalServices application will not be visible to the cloud.

After you start and connect the gateway client, go to the **Secure Gateway Dashboard** in the IBM Cloud console where you can see that the status and number of connections are changed.

### 1d. Add a destination to the Secure Gateway client

After you connect the LocalServices application to IBM Cloud, you define a destination.

1. Go to the **Sample Gateway Dashboard** in the IBM Cloud console.
2. Click your gateway (the gray button).
3. In the gateway detail screen, click **Add Destination**.
4. In the window that opens, select **On-Premises**, and click **Next**.
5. For the **LocalServices** that we want to consume from IBM Cloud, enter the host name (`localhost`) and the port (9445). In our example, we use `localhost` for the host name because the Secure Gateway client and the LocalServices application are installed on the same
machine. The port 9445 is the HTTPS port of the LocalServer Liberty profile that runs the LocalServices application. Click **Next**.

6. Specify the **protocol** that you want to use. Select **HTTPS protocol** for both communications between the **CloudServices** application and the Secure Gateway client, and between the Secure Gateway client and the **LocalServices** application. You do not upload any certificate because you use the default certificate. Click **Next**.

7. Select **Destination-Side** for the authentication. Click **Next**.

8. Set the destination that you are configuring to **private**. A **private destination** means that access is allowed only from specific IPs and ports. You can set up the IP ranges statically (usually for resources that have static IPs) or dynamically by using the Secure Gateway API (that is, IBM Cloud applications where the IP can change dynamically). In this tutorial, for simplicity, no restriction is set up. Click **Next**.

9. For Destination, enter **LocalServices**, and click **Finish**.

Your destination is now created. The host name that you specified in step 5 is resolved from the client, and not from the IBM Cloud server, which is the reason why we can set `localhost` in the IBM Cloud destination configuration wizard. Note also the following points:

- You can expose a service that is not visible directly from the internet.
- Your client machine does not need a public or static IP address.

On the Sample Gateway Dashboard, when you click the gray button with your gateway name, you see the newly created destination. If you click the **gear** button, you see the details of the destination in a pop-up window as shown in the following figure. Here, you see the Cloud Host : Port entry, which is the URL that the IBM Cloud service that is generated for the **LocalServices** destination in the cloud. Remember the Cloud Host : Port of your destination because you will need it later in this tutorial.

Test your destination:

1. Start the **LocalServer**.
2. Copy the **Cloud Host : Port URL** to the clipboard.
3. Append `/LocalServicesEJB/LocalMessageServiceService?wsdl`:
   https://cap-sg-prd-4.integration.ibmcloud.com:16977/LocalServicesEJB/LocalMessageServiceService?wsdl

4. Paste the URL into a web browser.

If you configured everything properly, you see the WSDL of the web service that is exposed by the LocalServices application in the browser. Look at the gateway client command line that you started, and you see a message like the following example:

```
Connection #number is being established to localhost:9445
```

This step completes the configuration and makes the LocalServices application available to the CloudServices application that we deploy on IBM Cloud.

### 2. Configure the Eclipse workspace

To deploy the existing applications to IBM Cloud, you can use the IBM Cloud Tools that are available in your Eclipse development environment that you installed in Part 1. In this step, you create a new IBM Cloud server and configure it to run in your cloud space. After you complete this step, you can deploy the Liberty servers and applications from your Eclipse workspace.

1. Open the Eclipse workspace that you configured in Part 1.
2. Select **File -> New -> Other**.
3. In the wizard, search for the **Server** folder, expand it, and select **Server**. Then, click **Next**.
4. In the Define a New Server page, expand the **IBM** folder, and select **IBM Cloud**. Then, click **Next**.
5. In the IBM Cloud Account window, enter your IBM Cloud account credentials and the same geographic region that you selected when you created your account. Click **Validate Account**.

If the validation is successful, click **Next**.
6. In the next screen, expand your account name. You now see the IBM Cloud spaces that are available in your account. In our example, as shown in the following figure,
our Cloud space is space000. Select one space from the list, and click **Finish**.

You now see a new entry of IBM Cloud in the list of configured servers.

### 3. Deploy the CloudServer Liberty profile to IBM Cloud

You are now ready to deploy the FrontendServer and CloudServer Liberty profiles, together with the Java EE applications that you deployed on the server in **Part 1**, to IBM Cloud. By using the IBM Cloud Tools, you can deploy an application or a whole Liberty server to IBM Cloud. The configuration that you performed in Part 1 to enable the SP-initiated SSO will be installed on IBM Cloud, enabling SSO across the cloud. First, you deploy the CloudServer. In step 4, you deploy the FrontendServer.

#### 3a. Change the endpoint URL

Before you deploy the CloudServer profile, you must modify the Java class that performs the web service call to the LocalServices application. You do this step because you configured the Secure Gateway client to expose the LocalServices web service to IBM Cloud. Also, the endpoint URL changed from:

https://localhost:9445/LocalServicesEJB/LocalMessageServiceService

The endpoint URL is now:

https://Cloud Host : Port/LocalServicesEJB/LocalMessageServiceService

The endpoint URL is the destination that you configured in **1d. Add a destination to the Secure Gateway client**, which is:

https://cap-sg-prd-4.integration.ibmcloud.com:16977/LocalServicesEJB/LocalMessageServiceService?wsdl

To modify the Java class, double-click the **CloudMessageService.java** class, in the workspace under `cloudServicesEJB/earModule/it.ibm.cs.samples.ejb` to open it. Change `String endpointURL:`
public String callLocalService()
{
    LocalMessageServiceService service = new LocalMessageServiceService();
    LocalMessageServiceSEI port = service.getLocalMessageServicePort();
    BindingProvider bp = (BindingProvider) port;

    String endpointURL = "https://cap-sg-prd-4.integration.ibmcloud.com:16977/LocalServicesEJB/
LocalMessageServiceService";
    bp.getRequestContext().put(BindingProvider.ENDPOINT_ADDRESS_PROPERTY, endpointURL);
    return port.getLocalMessage();
}

3b. Export the new LocalServices certificate

Export the new LocalServices certificate by exporting the certificate from the site that you used
before to test the Secure Gateway client setup:

https://Cloud Host : Port/LocalServicesEJB/LocalMessageServiceService?wsdl

In this example, we use the Firefox web browser to download the certificate. Other browsers have
similar steps to download the certificate.

1. Open the URL in a browser, replacing Cloud Host : Port with your destination.
2. Click the small lock icon near the address bar.
3. Click the arrow near the certificate entry.
4. Click More Information.
5. In the dialog box that opens, click View Certificate.
6. On the Details tab, click Export.
7. Export the certificate as a PEM file in a location on your system.

3c. Import the LocalServices certificate in the CloudServer keystore

Import the downloaded LocalServices certificate to the CloudServer keystore. To begin, open
a command line, and go to <WLP_SERVERS>\CloudServer\resources\security\. Then, run the
following command, where <downloaded_certificate_absolute_path> is where you saved the
PEM file in the step 3b:

keytool -importcert -file <downloaded_certificate_absolute_path> -keystore cloudkey.jks -storepass passw0rd

3d. Deploy the application to IBM Cloud

To deploy the CloudServer application to IBM Cloud:

1. Republish the CloudServices application on a local CloudServer. In the Servers view, right-
   click the CloudServer application and select Publish.
2. Stop the CloudServer.
3. Deploy the whole CloudServer application to IBM Cloud. Right-click IBM Cloud, and select
   Add and Remove.
4. In the Add and Remove window, select **CloudServer** from the Available pane on the left side, and click **Add** to move it to the Configured pane on the right side. Then, click **Finish**.

![Image of Add and Remove window](image)

5. In the Application details window, enter the name of the application as you want it to display in IBM Cloud. We used **cloudServices**. Because this name will be used as part of the public domain name, CloudService.mybluemix.net, it might already be in use. In this case, try to use a different name. For Buildpack URL, leave the field blank because we don't use it in this tutorial. Click **Next**. In the Launch Deployment window, you see the host name for the server that you created in IBM Cloud. In this scenario, the host name is **cloudService.mybluemix.net**. You might have to choose a different host name because it is a public domain and the name cannot be duplicated. You can also see and change the memory limit. In this scenario, 512 MB is enough memory.

6. Click **Validate** to see whether the host name is available. If it is not available, change it. Click **Next**.

7. In the next window, leave the Secure Gateway unbound so that it can be called by any application that knows its IP. Click **Finish**.

In the Eclipse console view, notice all the logs about your configuration activity. The deployment can take several minutes. When the deployment completes successfully, you see in the log a message, stating that the CloudServices application has been started.

Go to your dashboard in the **IBM Cloud console**, you see that your new IBM Cloud server is up and running. From the dashboard, open the details that are associated with the **CloudServices** application. You see the number of instances of the server, the memory that is assigned, and the log files. From the detailed view, you can also do basic operations, such as to stop and start the server.
4. Deploy the FrontendServer Liberty profile to IBM Cloud
The procedure to deploy the FrontendServer profile is similar to the procedure for deploying the CloudServer profile.

4a. Change the endpoint URL
Change String endpointURL in the AbstractMessageServlet.java file that is in the FrontendWeb/src/it.ibm.fe.samples.servlets path:

```java
protected void doService(HttpServletRequest request, HttpServletResponse response, ServiceResults serviceResults) throws ServletException, IOException {
    ...
    String serverName = request.getServerName();
    String endpointURL = "https://CloudServices.mybluemix.net/CloudServicesEJB/CloudMessageServiceService";
    bp.getRequestContext().put(BindingProvider.ENDPOINT_ADDRESS_PROPERTY, endpointURL);
    ...
}
```

Change CloudServices.mybluemix.net with your actual domain.

4b. Export the new CloudServices certificate
Export the certificate from the following URL:

```
http://CloudServices.mybluemix.net/CloudServicesEJB/CloudMessageServiceService?WSDL
```

Replace CloudServices.mybluemix.net with your domain, as you did in the previous step.

Download the CloudServices certificate. Again, we use Firefox, but you can use other browsers that have similar steps to download the certificate.

1. Open the URL in a browser, replacing CloudServices.mybluemix.net with your domain.
2. Click the small lock icon near the address bar.
3. Click the arrow near the certificate entry.
4. Click More Information.
5. In dialog box that opens, click View Certificate.
6. On the Details tab, click Export.
7. Export the certificate as a PEM file in a location on your system.

4c. Import the CloudServices certificate to the FrontendServer keystore
The Frontend application calls the web service by using SOAP over HTTPS. Therefore, you must import the CloudServices IBM Cloud certificate into the Frontend keystore.

To begin, open a command line, and go to <WLP_SERVERS>\FrontendServer\resources\security\*. Then, enter the following command, where <downloaded_certificate_absolute_path> is where you saved the PEM file in step 4b:
keytool -importcert -file <certificate_file_with_absolute_path> -keystore frontendkey.jks -storepass passw0rd

4d. Perform the FrontendServer deployment to IBM Cloud

To deploy the FrontendServer to IBM Cloud, you repeat the steps from 3d that you completed for the CloudServer profile. The difference is in the domain. For this scenario, we use FrontendServices.mybluemix.net. You might need to choose a different domain because this domain name is a public domain name and cannot be duplicated worldwide.

The two servers—one for the CloudServer profile and one for the FrontendServer profile—are now installed on IBM Cloud.

5. Test the applications that are deployed on IBM Cloud

To test the solution, enter the following URL, where you replace FrontendServices.mybluemix.net with the domain name that you chose for the FrontendServer profile when you deployed it on IBM Cloud:

https://FrontendServices.mybluemix.net/FrontendWeb

Enter alice for the user name and passw0rd for password. You now see the home page.

In the left navigation pane, click the **Restricted Services** link. If you completed the setup correctly, you see the message that comes from your LocalServices application in the body of the page.
Session affinity

In a clustered solution, you must consider session affinity. Think about the SSO model that we adopted in this solution. You authenticate against the identity provider. The identity provider generates a SAML token and posts it back to the service provider. The service provider uses the information that is in the SAML token to create a security context. The service provider remembers the SAML token because it needs it to propagate the user identity to the web services.

In a traditional clustered environment (not running on the cloud), you avoid the risk of initiating a new SSO process multiple times by enabling the session affinity. If you have many server instances, and you enable session affinity, the HTTP server in front of your cluster generates an affinity token (a string identifier of a server instance in the cluster) and appends it to the session cookie. In this way, every request that your browser makes always lands on the same application server instance of the cluster. In IBM Cloud, we do not have a cluster, so how do we solve this problem?

The Cloud Foundry (CF) specification introduces the concept of HTTP Routing and session affinity in a multi-instance cloud environment. We implemented session affinity in the FrontendWeb application project. To see the code that enables session affinity, from your Eclipse workspace, open the /FrontendWeb/WebContent/affinitySetup.jsp file. This code, which is shown in the following snippet, has been adapted to run on IBM Cloud.

```java
boolean sessionfound = false;
Cookie[] cs = request.getCookies();
if (cs != null)
{
    for (int index = 0; index < cs.length; index++)
    {
        Cookie c = cs[index];
        if (c != null)
        {
            if (c.getName() == null)
                continue;
            else if (c.getName().equalsIgnoreCase("JSESSIONID"))
            {
                sessionfound=true;
                break;
            } else if (c.getName().equalsIgnoreCase("__VCAP_ID__"))
            {
                sessionfound=true;
                break;
            }
        }
    }
}
```
Conclusion

You have implemented a secure SSO architecture and deployed it to the IBM® IBM Cloud implementation. You also configured a secure tunnel to expose your private intranet services to your public services that are running in the cloud, by using the IBM Secure Gateway. You learned how to implement session affinity in a multi-instance application. In Part 3 of this series, you extend the SSO scenario on the cloud to integrate Microsoft Windows authentication by using the Simple and Protected GSS-API Negotiation Mechanism (SPNEGO) and the Active Directory Domain Services.

Acknowledgments

The authors thank Massimo Leoni, Carlo Randone, and Pavel Travnik for reviewing this article and providing their comments.
Related topics

- What is IBM Cloud
- IBM Cloud Documentation
- WebSphere Liberty (WASDev) Developer Center
- IBM Cloud Developer Center

© Copyright IBM Corporation 2017

Trademarks
(www.ibm.com/developerworks/ibm/trademarks/)