Connectivity Proxy Component

Scenarios

The Connectivity Proxy component can be used for following scenarios:

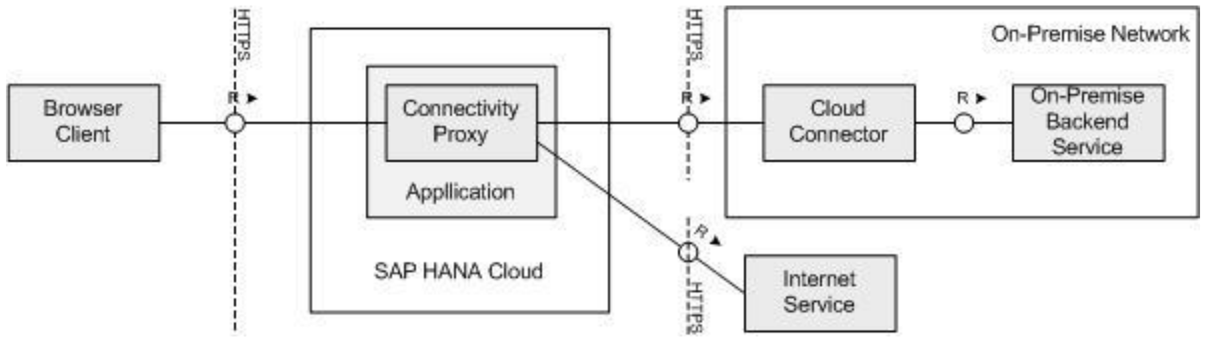
1. A JavaScript application running in a Web browser and deployed on SAP HANA Cloud wants to read/write data to/from a backend service. Due to the Same-Origin-Policy of JavaScript the application is not allowed to access a backend service that is accessible under a different domain as the originating domain of the application. The Connectivity Proxy can be used by the Web application as proxy between the JavaScript client and the backend service to make sure the same-origin-policy is kept.

2. An on-demand application not running on SAP HANA Cloud wants to integrate with on-premise backend services using the SAP HANA Cloud Connector as on-premise agent to establish the secure technical connectivity. For this, the Connectivity Proxy component can as well be used as proxy application on SAP HANA Cloud.

Connectivity Proxy

The Connectivity Proxy is developed under the Apache License v2.0 and is provided in SAP HANA Cloud's github branch [1]. Interested users are welcome to use the Connectivity Proxy and are invited to contribute extensions it in case useful functionality is missing.

Let's start with a description of the proxy. Shortly summarized, the proxy is a standard Java servlet which accepts GET/POST/PUT/DELETE requests and delegates them to a backend service, as illustrated by the following figure.



The backend service has to be defined by a *destination* on SAP HANA Cloud (see [2] for more details on destinations), while the destination name is expected in the servlet URL adhering to the following pattern:

```
/<context-path>/<servlet-path>/<destination>/<relative-path-to-backend-service>
```

The different parts of the URL are following:

- The `<context-path>` and `<servlet-path>` are defined statically by the application and are the standard configurations of Java EE Web applications. The <servlet-path> is defined by the servlet mapping within the `web.xml` of the Web application

- The `<destination>` path specifies the backend service. Destinations are the recommended approach on SAP HANA Cloud to configure remote calls to external services, like public internet or on-premise backend services. Destinations used by an application can be configured after deployment of the application, and the connectivity service takes care of tenant isolation, i.e. makes sure that only the application of the originating SAP HANA Cloud account is able to access the related destination configuration.

The Connectivity Proxy passes the request headers of the origin client request to the backend service, as well as copies the response headers received from the backend service when sending back the response to the originating client.

The response body received from the remote service might contain URLs which point again to the remote service. The Connectivity Proxy rewrites those URLs in the response sent to the originating client so that they point again to the Connectivity Proxy and specified destination.

How to use the Connectivity Proxy

Option A: Deploy Connectivity Proxy as WAR file

1. Download the Connectivity Proxy from github and build it in your local environment. The github project is prepared for Maven builds, i.e. a simple “mvn install” on the project will generate the WAR file in the “/target” folder. The servlet-path for the proxy servlet specified in the project is “/proxy”. If you like to change this path, go into the “/src/main/webapp/WEB-INF” folder of the project and change the url-pattern for the “ConnectivityProxy” servlet.

2. Deploy the WAR file containing the Connectivity Proxy in the SAP HANA Cloud application where you want to use it. See [3] for more details on how to deploy multiple WAR files for a single application. After deployment, you are able to call the proxy servlet via “/connectivity.proxy/proxy”.

Option B: Copy the ProxyServlet into your Web application project

1. Copy the `com.sap.cloudlabs.connnectivity.proxy.ProxyServlet.java` file into the source folder of your Web application project.

2. Modify the `web.xml` file of your Web application to specify the proxy servlet and the destination factory as a used resource as following:

```
<!-- ============================================================= -->
<!-- Connectivity Proxy servlet  -->
<!-- ============================================================= -->

<servlet>
       <display-name>ConnectivityProxy</display-name>
       <servlet-name>ConnectivityProxy</servlet-name>
       <servlet-class>com.sap.cloudlabs.connectivity.proxy.ProxyServlet</servlet-class>
</servlet>
<servlet-mapping>
       <servlet-name>ConnectivityProxy</servlet-name>
       <url-pattern>/proxy/*</url-pattern>
</servlet-mapping>

<!-- ============================================================= -->
<!-- JNDI resource definition of DestinationFactory -->
<!-- ============================================================= -->

<resource-ref>
       <res-ref-name>connectivity/DestinationFactory</res-ref-name>
       <res-type>com.sap.core.connectivity.api.DestinationFactory</res-type>
</resource-ref>
```

Limitations

The Connectivity Proxy has been tested against backend systems which provide REST APIs and produce OData documents as responses. For other types of backends (e.g. called via SOAP Web Services) extensions might be needed, e.g. in the area of URL rewriting.

References

[1]
Connectivity Proxy in github:
https://sap.github.io/cloud-connectivity.proxy

[2]
Documentation about Destinations:
https://help.hana.ondemand.com/help/frameset.htm?e4f1d97cbb571014a247d10f9f9a685d.html

[3]
Documentation about deploying multiple WAR files using SAP HANA Cloud Console Client:
https://help.hana.ondemand.com/help/frameset.htm?030863cd5d0d4dd3b742957970f8eec9.html