KB – SAP ARIBA INTEGRATION HTTPS SETUP INSTRUCTIONS

PURPOSE

This KB outlines how to configure and setup HTTPS (Secure HTTP) for the ONESOURCE Global Tax for SAP Ariba Solutions when installed on-premise and secure HTTP is desired to be used.

PREREQUISITES

- Understanding of deployment of Java and Web Application Server based solutions
- Ability to update xml structure files
- Understanding of networking and site-specific setup
- Installed and configured ONESOURCE Determination and Ariba Integration 1.0.2.x or higher

SETUP STEPS

By default, the Ariba integration applications receive and send data using the HTTP protocol. The applications can be readily configured to instead use the HTTPS protocol, if desired, by using properties of the Spring Boot implementation framework.

There are two standard networking solutions for providing an HTTPS security layer on top of web services:

1. A **proxy server** stands in for the integration server, to verify authenticity for all inbound traffic.
2. Ariba ERP to Integration **direct communication** via HTTPS.

CONFIGURATION FOR PROXY SERVERS

This setup option allows a proxy server to check white lists for inbound traffic as well as qualifying domain names fully for HTTPS authentication. There will be no need to have any SSL certificates on the actual integration application, because that task has been handled by the proxy server.

1. Configure the integration application's server.port property using a standard non-reserved port number that is not already in use on that server.

2. When launching the application, add the following parameters to the command line:
   -Dhttps.proxyHost=<IP address, or DNS alias, for the proxy server for HTTPS traffic>
   -Dhttps.proxyPort=<The port number for HTTPS proxied traffic>
   -Dhttp.proxyHost=<IP address, or DNS alias, for the proxy server for HTTP traffic>
   -Dhttp.proxyPort=<The port number for HTTP proxied traffic>

If you find the command line to be too lengthy, you can place the parameters into a ".conf" file using the name of the integration; for example, "ariba-cloud-integration.conf" or "ariba-cloud-integration-v2.conf". Here is a sample file:

```
JAVA_HOME=/app/utils/java8
RUN_ARGS=--spring.config.location=/app/taxadmin/spring-boot/
JAVA_OPTS="-Xmx2g \
        -Dhttp.keepalive=true \
        -Dhttp.maxRedirects=20 \
        -Dhttps.proxyHost=webproxy.int.mydomain.com \
        -Dhttps.proxyPort=80 \
        -Dhttp.proxyHost=webproxy.int.mydomain.com \
        -Dhttp.proxyPort=80 \
        -Doracle.jdbc.timezoneAsRegion=false"
```

CONFIGURATION FOR DIRECT ERP TO INTEGRATION HTTPS TRAFFIC

NOTE: For this configuration approach, each application that has HTTPS communications must be individually configured: ariba-cloud-ui, ariba-cloud-integration, and ariba-cloud-integration-v2. You may be able to share common properties, see Install Guide for more information.

1. The integration application will need an SSL certificate that is stored inside a keystore. It can be a self-signed, generated certificate, or it can be an existing SLL Certificate that you import into a keystore for this purpose.

   NOTE: If you use a Self-Signed certificate, your browser may give you an error message like this:
   localhost:8070 uses an invalid security certificate.
   The certificate is not trusted because it is self-signed.
   Error code: MOZILLA_PKIX_ERROR_SELF_SIGNED_CERT

   To generate a self-signed certificate, follow these steps:

   a) Generate a new keystore with this command:
      keytool -genkeypair -alias YOURKEYSTORENAME -keyalg RSA -keysize 2048 -storetype PKCS12 -keystore keystore.p12 -validity 3650

      1. You will be asked to enter a password for the keystore. It must be at least 6 characters in length.
      2. When you are asked to input your first and last name, enter the name of your host.
      3. You may use default values or enter correct values for the next five questions.
      4. For the question "Is CN=<yourhost>, ...", enter "yes" if it echoes your data selection correctly.
      5. Use the same password as the keystore for the key password.
      6. Verify your keystore is correct with this command:
         keytool -list -v -storetype pkcs12 -keystore keystore.p12

   OR

   b. To import an existing SSL certificate into a new keystore, use this command:
      keytool -import -alias <YOURKEYSTORENAME> -file <YOUREXISTINGCERTIFICATENAME>.crt keystore keystore.p12 -storepass <YOURKEYSTOREPASSWORD>

2. Enable HTTPS traffic in the integration application using Spring Boot properties, by following these steps:

    a. Edit the application.yml file for the integration application ("ariba-cloud-integration/application.yml"), and add or edit the following properties:

    server.port: 8443
    # Require spring security to use ssl, disallow HTTP requests
    security.require-ssl: true

    #Set your format for keystore, either PKCS12 or JKS
    server.ssl.key-store-type: PKCS12

    #Where is the keystore master file on this system
    server.ssl.key-store: /<complete path designation>/keystore.p12

    #what's the password to use for certificate
    server.ssl.key-store-password: <password goes here>

    # what is the name or alias for the certificate?
    server.ssl.key-alias: YOURKEYSTORENAME

    b. Save this file and then launch the integration jar file:
    java -jar ./ariba-cloud-integration.jar

    c. Confirm that HTTPS works by accessing the application using HTTPS.

REFERENCE DOCUMENTATION

For more information on HTTPS configurations in a Spring Boot setup please consult this link:
https://docs.spring.io/spring-boot/docs/current/reference/htmlsingle/#howto-configure-ssl