WebSphere Liberty on z/OS

SAF Security

IBM Washington Systems Center

Lab Version Date: January 20, 2025

Table of Contents

Important: There is a folder on the Windows desktop named *CopyPaste Files.* This folder contains file with the commands and other text used in this workshop. Locate the file identified in the *General Exercise Information and Guidelines* section of this exercise and copy it to the desktop. Open the file and use the copy-and-paste function (**Ctrl-C** and **Ctrl-V**) to enter commands or text. It will save time and help avoid typo errors. As a reminder text that appears in this file will be highlighted in yellow.

Mitch Johnson (mitchj@us.ibm.com)

General Exercise Information and Guidelines

- ✓ This exercise requires the completion of the *WebSphere Liberty on z/OS – Basic Configuration* exercise before it can be performed.
- ✓ This exercise creates and performs the initial security configuration for a z/OS Connect Liberty server. Using z/OS Connect provides a good demonstrating of the various configuration options available.
- ✓ This exercise requires using z/OS user identies *FRED* and *USER1*. The RACF passwords for these users are *user1* and *fred* (lower case sensitive) respectively.
- ✓ There are examples of server.xml scattered through this exercise. Your server.xml may differ depending on which exercises have been previously performed. Be sure the red lines in these examples are either added or already present.
- ✓ The acronyms RACF (resource access control facility) and SAF (system authorization facility) are used in this exercise. RACF is the IBM security manager product whereas SAF is a generic term for any security manager product, e.g., ACF2 or Top Secret or RACF. An attempt has been to use SAF when referring to information appropriate for any SAF product and to use RACF when referring to specific RACF commands or examples.
- ✓ Any time you have any questions about the use of IBM z/OS Explorer, 3270 screens, features or tools, do not hesitate to ask the instructor for assistance.
- ✓ Text in **bold** and highlighted in **yellow** in this document should be available for copying and pasting in a file named *Basic Configuration CopyPaste* file on the desktop.
- ✓ Please note that there may be minor differences between the screen shots in this exercise versus what you see when performing this exercise. These differences should not impact the completion of this exercise.
- ✓ For information regarding the use of the Personal Communication 3270 emulator, see the *Personal Communications Tips* PDF in the exercise folder.

Mitch Johnson (mitchj@us.ibm.com)

Enabling SAF Security

Up to this point Liberty has been configured to use "basic" security – that is, defined in *server.xml* and managed by the Liberty server. In this section, security will be externalized to the local SAF product (RACF in the case of this workshop).

Using SAF for the security registry

___1. In a 3270-terminal session, use ISPF command ***=3.4*** to go to the data set list panel and display all the data sets whose names begin with *USER1.ZCEE30.**.

___2. Edit data set *USER1.ZCEE30.CNTL*. Next browse member **ZCEERACF**. The RACF commands are shown below.

```
ADDGROUP ZCEEUSRS OMVS(AUTOGID) OWNER(SYS1)

ADDGROUP WSGUESTG OMVS(AUTOGID) OWNER(SYS1)
ADDUSER WSGUEST RESTRICTED DFLTGRP(WSGUESTG) OMVS(AUTOUID -
  HOME(/u/wsguest) PROGRAM(/bin/sh)) NAME('UNAUTHENTICATED USER') -
  NOPASSWORD NOOIDCARD

CONNECT (FRED,USER1,JOHNSON) GROUP(ZCEEUSRS)

RDEFINE APPL BBGZDFLT UACC(NONE) OWNER(SYS1)
PERMIT BBGZDFLT CLASS(APPL) RESET
PERMIT BBGZDFLT CLASS(APPL) ACCESS(READ) ID(WSGUEST,USER2,ZCEEUSRS)
SETROPTS RACLIST(APPL) REFRESH

RDEFINE EJBROLE BBGZDFLT.zos.connect.access.roles.zosConnectAccess -
  OWNER(SYS1) UACC(NONE)
PE BBGZDFLT.zos.connect.access.roles.zosConnectAccess CLASS(EJBROLE) -
  RESET
PE BBGZDFLT.zos.connect.access.roles.zosConnectAccess -
  CLASS(EJBROLE) ID(ZCEEUSRS) ACCESS(READ)
SETR RACLIST(EJBROLE) REFRESH
```

What? In summary, these commands start by creating a z/OS Connect user group, ZCEEUSRS. Users connected to group *ZCEEUSRS* are authorized to use z/OS Connect.

Identity *WSGUEST* is created as the unauthenticated user, see https://www.ibm.com/support/knowledgecenter/SSAW57_liberty/com.ibm.websphere.wlp.nd.multiplatform.doc/ae/twlp_config_security_saf.html. Note that it is very important that SAF identities and groups used with z/OS Connect have OMVS segments.

The commands then creates an RACF application(APPL) and an EJROLE resources which will be used for controlling access to this z/OS Connect server. These will take the place of the equivalent security definitions in the basic security server.xml file.

Mitch Johnson (mitchj@us.ibm.com)

___3. Submit the job for executions by entering ISPF command ***SUBMIT*** on the command line. The job should complete with a return code of zero.

___4. Go to the ISPF Edit Entry Panel (option *2)* by entering ISPF command ***=2*** on the command line and pressing **Enter**.

___5. Enter ***/var/liberty/includes*** into the area beside *Name* under *Other Partitioned, Sequential or VSAM Data Set, or z/OS UNIX file:* and press **Enter**.

___6. Use the ***VA*** (View ASCII) line command to open the *safSecurity.xml* file in browse mode. You should see:

```
<?xml version="1.0" encoding="UTF-8"?>
<server description="saf security">

    <!-- Enable features -->
    <featureManager>
        <feature>appSecurity-2.0</feature>
        <feature>zosSecurity-1.0</feature>
    </featureManager>

    <webAppSecurity allowFailOverToBasicAuth="true" />

    <safRegistry id="saf" />
    <safAuthorization racRouteLog="ASIS" />
    <safCredentials unauthenticatedUser="WSGUEST"
       profilePrefix="BBGZDFLT" />

</server>
```

1

2

Tech-Tip: The value for the profilePrefix resource (*BBGZDFLT)* in the above commands must match the value of RACF application and value of the EJBRole prefix defined earlier.

Notes

1. The *zosSecurity-1.0* feature adds the z/OS security feature
2. The *safRegistry, safAuthorization* and *safCredentials* elements enable authentication and authorization using a SAF product.

___7. Repeat these steps to access ***/var/zosconnect/servers/myServer*** and edit *server.xml*.

___8. Change the *include* statement for *basicSecurity.xml* to an *include* for *safSecurity.xml*.

<include location="${shared.config.dir}/safSecurity.xml" />

___9. Add a new include statement for *keystore.xml*, see below:

<include location="${shared.config.dir}/keystore.xml" />

```
<?xml version="1.0" encoding="UTF-8"?>
<server description="new server">
<include location="${shared.config.dir}/safSecurity.xml"/>
<include location="${shared.config.dir}/ipic.xml"/>
<include location="${shared.config.dir}/keystore.xml"/>
```


Mitch Johnson (mitchj@us.ibm.com)

___10. Enter the following MVS command to refresh the IBM z/OS Connect server configuration:

F BAQSTRT,ZCON,REFRESH

> **Tech-Tip:** MVS and JES2 commands can be entered from SDSF by enter a / (slash) on the command line followed by the command itself (e.g. /D T). The command results can be found in the system log. If a command is especially long then simply entering a / (slash) to display *a SDSF – System Command Extension* panel where a command can span multiple lines. When a MVS command must be entered, the instructions in these exercises will indicate that the command is a MVS command. MVS commands can be entered at the prompt by using the / (slash) prefix or using the *SDSF – System Command Extension* panel.

___11. Close all instances of the Firefox browser (we want to force another prompt for ID, and closing the browser clears the cookies from before).

___12. Start Firefox and enter the following URL:

https://wg31.washington.ibm.com:9443/zosConnect/apis

___13. Next you will be prompted for a userid and password. Enter Enter **Fred** and **fredpwd** as you have before.

It should fail !

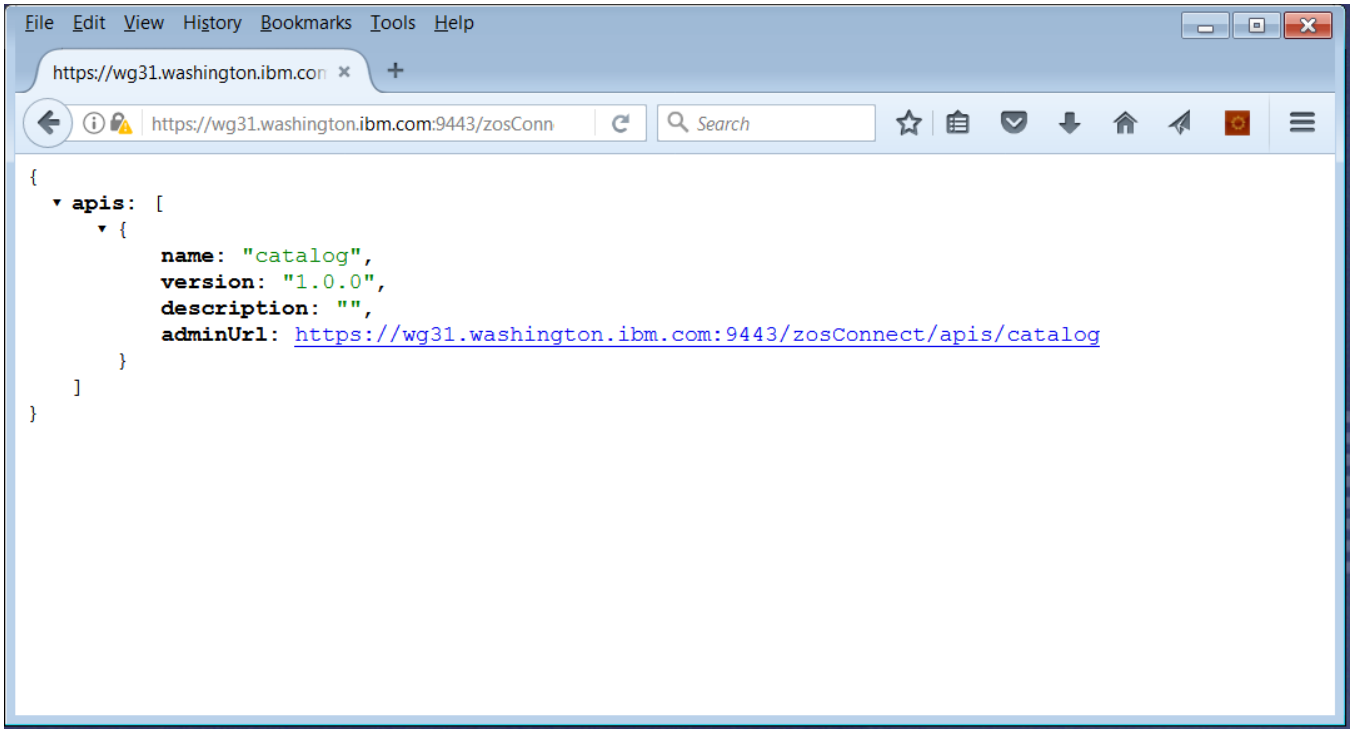
> **Why?** Because the password fredpwd is what we had in the basic registry in server.xml, but that is now gone. Now we are using SAF where Fred's password is FRED.

___14. In the userid/password prompt, enter ***Fred*** and ***FRED***.

___15. With SAF, case does not matter. All userid and password values are stored in upper-case. Anything entered in lowercase or mixed is folded to uppercase and compared against the SAF registry.

Mitch Johnson (mitchj@us.ibm.com)

___16. You should see a list of the APIs:



___17. Close the browser again and restart it and access the same URL. This time enter user USER2 and USER2's password of USER2 on the login prompt.

___18. The request should fail with message *Error 403: Authorization Failed*. Check the system log using SDSF and you should an ICH408I message (see below). USER2 does not have access to the EJBROLE resource protecting the z/OS Connect server.

```
ICH408I USER(USER2   ) GROUP(SYS1    ) NAME(WORKSHOP USER2
   BBGZDFLT.zos.connect.access.roles.zosConnectAccess
   CL(EJBROLE )
   INSUFFICIENT ACCESS AUTHORITY
   ACCESS INTENT(READ   )  ACCESS ALLOWED(NONE   )
```

Summary

The registry and authorization information was removed from the *server.xml*, and other XML was added to tell the Liberty z/OS server to use SAF as its registry (for userid and password) and role checking (EJBROLE).

By attempting to use Fred's password as coded in the server.xml (and the failure to authenticate), you verified that the server.xml copy of the registry was no longer in effect.

Mitch Johnson (mitchj@us.ibm.com)

Controlling z/OS Connect access using SAF groups

___1. Stop the server by entering the MVS command ***P BAQSTRT*** at the SDSF (ISPF command ***=sdsf.da***) command prompt.

___2. Next use ISPF command ***=6*** to go to the TSO command entry panel and add 2 new groups using the ***ADDGROUP*** command, e.g.

- ***ADDGROUP GMADMIN OMVS(AUTOGID)***
- ***ADDGROUP GMINVOKE OMVS(AUTOGID)***

> **Tech-Tip:** Enabling SAF security requires that user identities and groups must have OMVS segments.

___3. Connect user FRED to group *GMADMIN* using the ***CONNECT*** command, e.g.

- ***CONNECT FRED GROUP(GMADMIN)***

___4. Connect user USER1 to group *GMINVOKE* using the ***CONNECT*** command, e.g.

- ***CONNECT USER1 GROUP(GMINVOKE)***

___5. Access ***/wasetc/zc3lab/groupAccess.xml*** and you should see the XML elements as shown below.

```
<?xml version="1.0" encoding="UTF-8"?>
<server description="new server">

  <zosconnect_zosConnectManager
            globalInterceptorsRef="interceptorList_g"
            globalAdminGroup="GMADMIN"
            globalInvokeGroup="GMINVOKE"/>

  <zosconnect_authorizationInterceptor id="auth"/>

  <zosConnectInterceptors id="interceptorList_g"
         interceptorRef="auth" />

</server>
```

The lines in bold enable authorization checking at the global level. Two groups are designated for access. Group G*MADMIN* for administrators and *GMINVOKE* for users of the APIs.

___6. Repeat the steps required to access ***/var/zosconnect/servers/myServer*** and edit *server.xml*.

Mitch Johnson (mitchj@us.ibm.com)

___7. Add the *include* statement for *groupAccess.xml* (see below) to the list of include files in the *server.xml*.

<include location="${shared.config.dir}/groupAccess.xml"/>

```
<?xml version="1.0" encoding="UTF-8"?>
<server description="new server">
<include location="${shared.config.dir}/safSecurity.xml"/>
<include location="${shared.config.dir}/ipic.xml"/>
<include location="${shared.config.dir}/keystore.xml"/>
<include location="${shared.config.dir}/groupAccess.xml"/>
```

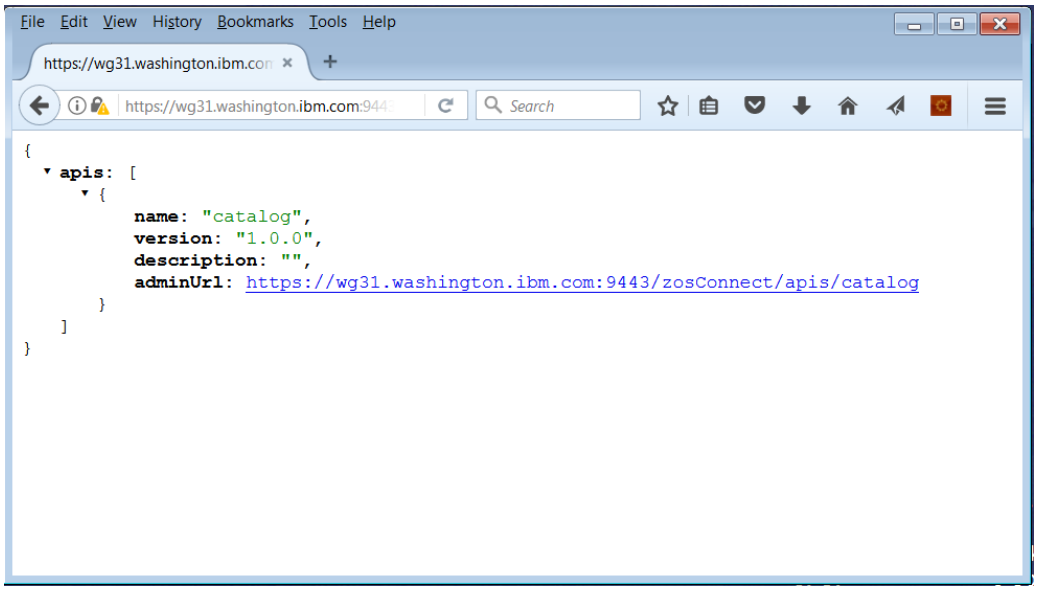
___7. Use SDSF to start your server with MVS command ***S BAQSTRT***. You should see your server start:

___8. Close all instances of the Firefox browser (we want to force another prompt for ID, and closing the browser clears the security token).

___9. Start Firefox and enter the following URL:

https://wg31.washington.ibm.com:9443/zosConnect/apis

___10. On the *Authentication Required* popup window, enter ***Fred*** and ***FRED***. You should see:

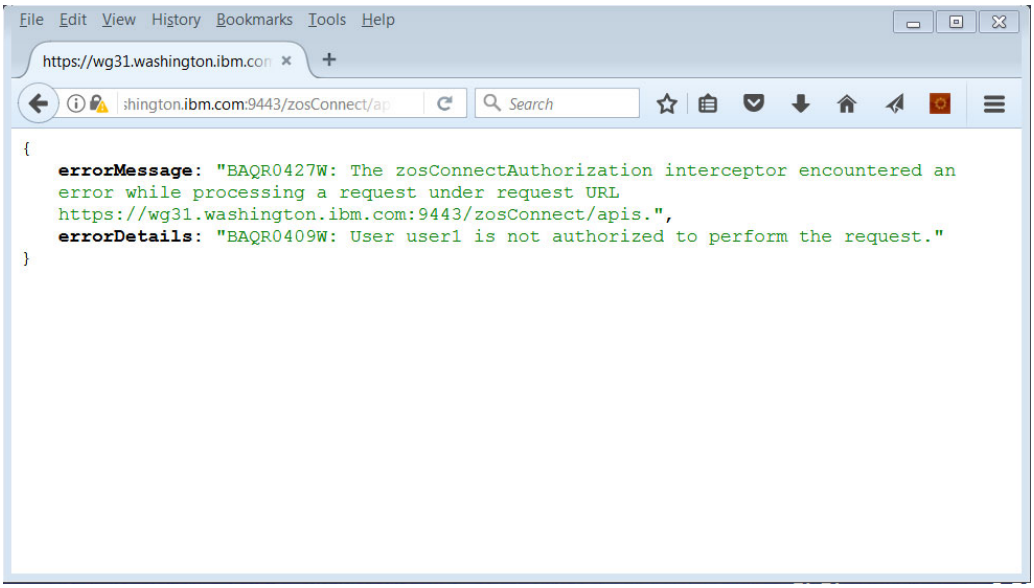


SAF identity FRED in in the global admin group and has the authority perform this function.

___11. Close Firefox session to clear the security token and restart and access the same URL.

Mitch Johnson (mitchj@us.ibm.com)

___12. On the *Authentication Required* popup enter ***USER1*** and USER1's password of USER1. You should see:



Next try to invoke an API.

___13. Use the *Command Prompt* icon on the desktop to open a DOS command prompt session.

___14. In the session, use the change directory (cd) command to go to directory c:\z\admin, e.g.

cd c:\z\admin

___15. Paste the command below at the command prompt and press **Enter**.

curl -X POST --user USER1:USER1 --header "Content-Type: application/json" -d @inquireSingle.json --insecure https://wg31.washington.ibm.com:9443/zosConnect/services/inquireSingle?action=invoke

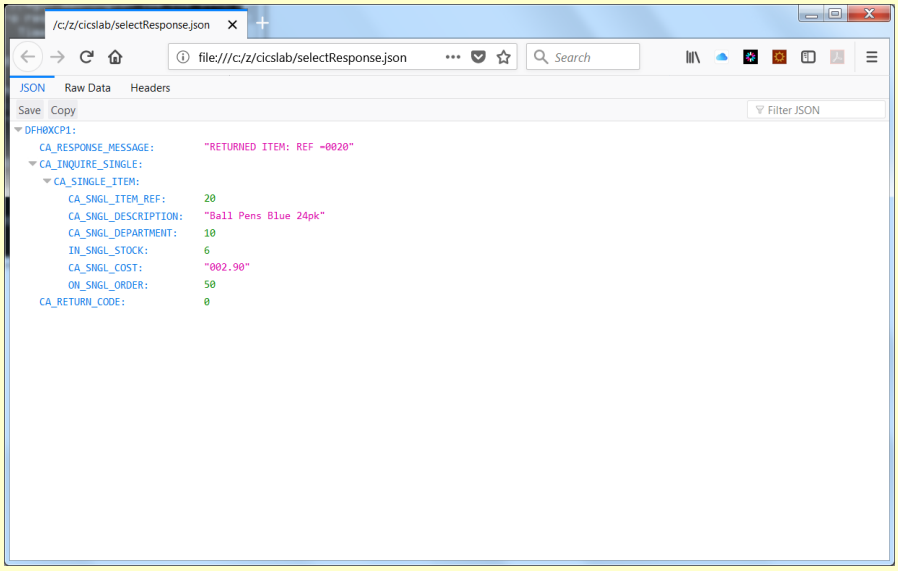
Mitch Johnson (mitchj@us.ibm.com)

___16. You should see the response below:


```
{"DFH0XCP1":{"CA_RESPONSE_MESSAGE":"RETURNED ITEM: REF =0020","CA_INQUIRE_SINGLE
":{"CA_SINGLE_ITEM":{"CA_SNGL_ITEM_REF":20,"CA_SNGL_DESCRIPTION":"Ball Pens Blue
 24pk","CA_SNGL_DEPARTMENT":10,"IN_SNGL_STOCK":6,"CA_SNGL_COST":"002.90","ON_SNG
L_ORDER":50}},"CA_RETURN_CODE":0}}
```


USER1 can invoke the service but has no administrative authority.

Tech-Tip: Adding the *-o* flag to the cURL command will write the JSON response message to a file rather than back to the terminal session. So if you add ***-o selectResponse.json*** to the cURL command and use the command ***firefox file:///c:\z\admin\selectResponse.json*** you will see a browser session open with the JSON response formatted as below:



___17. To demonstrate an operational function, paste the command below at the command prompt and press **Enter**.

curl -X PUT --user USER1:USER1 --insecure https://wg31.washington.ibm.com:9443/zosConnect/services/inquireSingle?status=stopped

___18. You should see the response below:


```
{"errorMessage":"BAQR0406W: The zosConnectAuthorization interceptor encountered
an error while processing a request for service  under request URL https://wg31.
washington.ibm.com:9443/zosConnect/services/inquireSingle.","errorDetails":"BAQR
0409W: User USER1 is not authorized to perform the request."}
```


Again, USER1 can invoke the service but has no administrative authority.

Mitch Johnson (mitchj@us.ibm.com)

Using SAF for TLS and key store management

Now let's go through the steps to use SAF for transport layer security (TLS) management.

Note: Transport Layer Security(TLS) is the successor to Secure Socket Layer(SSL). For our purposes the acronyms are interchangeable.

___1. Stop your server again with MVS command ***P BAQSTRT***.

___2. Go to data set *USER1.ZCEE30.CNTL* data set and browse the **ZCEETLSS** (TLS server role) member. You will see something like this:

```
RACDCERT CERTAUTH GENCERT SUBJECTSDN(CN('CA for Liberty') -
  OU('LIBERTY')) WITHLABEL('Liberty CA') TRUST -
  SIZE(2048) NOTAFTER(DATE(2020/12/31))

RACDCERT ID(LIBSERV) GENCERT SUBJECTSDN(CN('wg31.washington.ibm.com') -
  O('IBM') OU('LIBERTY')) WITHLABEL('Liberty Client Cert') -
  SIGNWITH(CERTAUTH LABEL('Liberty CA')) SIZE(2048) -
  NOTAFTER(DATE(2020/12/30))

RACDCERT ID(FRED) GENCERT SUBJECTSDN(CN('Fred D. Client') -
  O('IBM') OU('LIBERTY')) WITHLABEL('FRED') -
  SIGNWITH(CERTAUTH LABEL('Liberty CA')) SIZE(2048) -
  NOTAFTER(DATE(2020/12/30))

RACDCERT ID(USER1) GENCERT SUBJECTSDN(CN('USER1 D. Client') -
  O('IBM') OU('LIBERTY')) WITHLABEL('USER1') -
  SIGNWITH(CERTAUTH LABEL('Liberty CA')) SIZE(2048) -
  NOTAFTER(DATE(2020/12/30))

RACDCERT ID(LIBSERV) ADDRING(Liberty.KeyRing)

RACDCERT CONNECT(ID(LIBSERV) -
  LABEL('Liberty Client Cert') RING(Liberty.KeyRing)) -
  ID(LIBSERV)

RACDCERT CONNECT(CERTAUTH LABEL('Liberty CA') -
  RING(Liberty.KeyRing)) ID(LIBSERV)

SETR RACLIST(DIGTCERT DIGTRING) REFRESH

PERMIT IRR.DIGTCERT.LISTRING -
  CLASS(FACILITY) ID(LIBSERV) ACCESS(READ)

PERMIT IRR.DIGTCERT.LIST -
  CLASS(FACILITY) ID(LIBSERV) ACCESS(READ)

SETR RACLIST(FACILITY) REFRESH
```

Tech-Tip: In summary, these commands create a server certificate and self-signed that certificate (RACF will be our certificate authority for this exercise). Personal certificates for identities LIBSERV, FRED and USER1 are created and signed by this RACF signing certificated. Then a key ring is created in RACF and the signing certificate and the personal certificate for LIBSERV is added to this key ring.

Mitch Johnson (mitchj@us.ibm.com)

Tech-Tip: The commands below are provided for reference only. If RACF is not the certificate authority, then the command for generating a personal certificate and importing the signed certificate provided by the certificate will be required instead. See an example flow below:

```
Generate a certificate for identity LIBSERV
racdcert id(LIBSERV) gencert subjectsdn(CN('wg31.washington.ibm.com')
 O('IBM') OU('LIBERTY')) withlabel('Liberty Server Cert')

Export a certificate request for certificate labeled 'Liberty Server Cert'
racdcert id(LIBSERV) genreq(label('Liberty Server Cert')) dsn(CERT.REQ)

Send the certificate (contents of CERT.REQ) to your Certificate Authority to be
signed. Store the return signed certificate in data set LIBSERV.P12. The public
certificate for the Certificate Authority probably already has been imported
into RACF. For the purposes of this example. The public certificate authority
certificate has been previously saved in data set LIBCA.PEM.

If necessary, import the public certificate for the certificate authority
racdcert CERTAUTH withlabel('Liberty CA') add(LIBCA.PEM)

Import the personal certificate (with a private key) for identity LIBSERV
racdcert id(LIBSERV) withlabel('Liberty Server Cert') add('LIBSERV.P12)
password('secret') TRUST

Create a key ring named 'Liberty.KeyRing' for identity LIBSERV
racdcert id(LIBSERV) addring(Liberty.KeyRing)

Connect LIBSERV's the certificate to this key ring as the default certificate
racdcert id(LIBSERV) connect(id(LIBSERV)
  label('Liberty Client Cert') ring(Liberty.KeyRing) default)

Connect the certificate authority certificate to the key ring
racdcert id(LIBSERV) connect(certauth label('Liberty CA')
  ring(Liberty.KeyRing))

Refresh the in-storage profiles for the DIGTCERT and DIGTRING resource classes.
setropts raclist(digtcert,digtring) refresh

Provide access to identity LIBSERV to LIBSERV's key rings
permit IRR.DIGTCERT.LISTRING class(FACILITY) id(LIBSERV) access(read)

Provide access to identity LIBSERV to LIBSERV's certificate
permit IRR.DIGTCERT.LIST class(FACILITY) id(LIBSERV) access(read)

Refresh the in-storage profiles for the FACILITY resource class
setropts raclist(FACILITY) refresh
```


Mitch Johnson (mitchj@us.ibm.com)

___3. Submit that job and look for the *MAXCC=0000* indicator of success.

___4. Go to the ISPF Edit Entry Panel (option *2)* by entering ISPF command ***=2*** on the command line and pressing **Enter**.

___5. Enter ***/var/zosconnect/servers/myServer/includes*** into the area beside *Name* under *Other Partitioned, Sequential or VSAM Data Set, or z/OS UNIX file:* and press **Enter**.

___6. Use the ***VA*** (View ASCII) line command to open the *keyring.xml* file in browse mode. You should see:

```
<?xml version="1.0" encoding="UTF-8"?>
<server description="ssl security">

    <!-- Enable features -->
    <featureManager>
        <feature>transportSecurity-1.0</feature>
    </featureManager>

     <sslDefault sslRef="DefaultSSLSettings" />
     <ssl id="DefaultSSLSettings"
        keyStoreRef="CellDefaultKeyStore"
        trustStoreRef="CellDefaultKeyStore" />
     <keyStore id="CellDefaultKeyStore"
        location="safkeyring:///Keyring.LIBERTY"
        password="password" type="JCERACFKS"
        fileBased="false" readOnly="true" />
</server>
```

1
2

Notes

1. The *transportSecurity-1.0* feature has been added
2. The basic *keystore* element specifying a trust store was replace by a keystore elements specify a SAF keyring.

___7. Change the *sever.xml* to replace include for the *keystore.xml* to an include for *keyring.xml*

<include location="${shared.config.dir}/keyring.xml" />

```
<?xml version="1.0" encoding="UTF-8"?>
<server description="new server">
<include location="${shared.config.dir}/ipic.xml" />
<include location="${shared.config.dir}/safSecurity.xml" />
<include location="${shared.config.dir}/keyring.xml" />
<include location="${shared.config.dir}/groupAccess.xml" />
```

___8. Start your server with MVS command ***S BAQSTRT***.

___9. Close all instances of your Firefox browser[1].

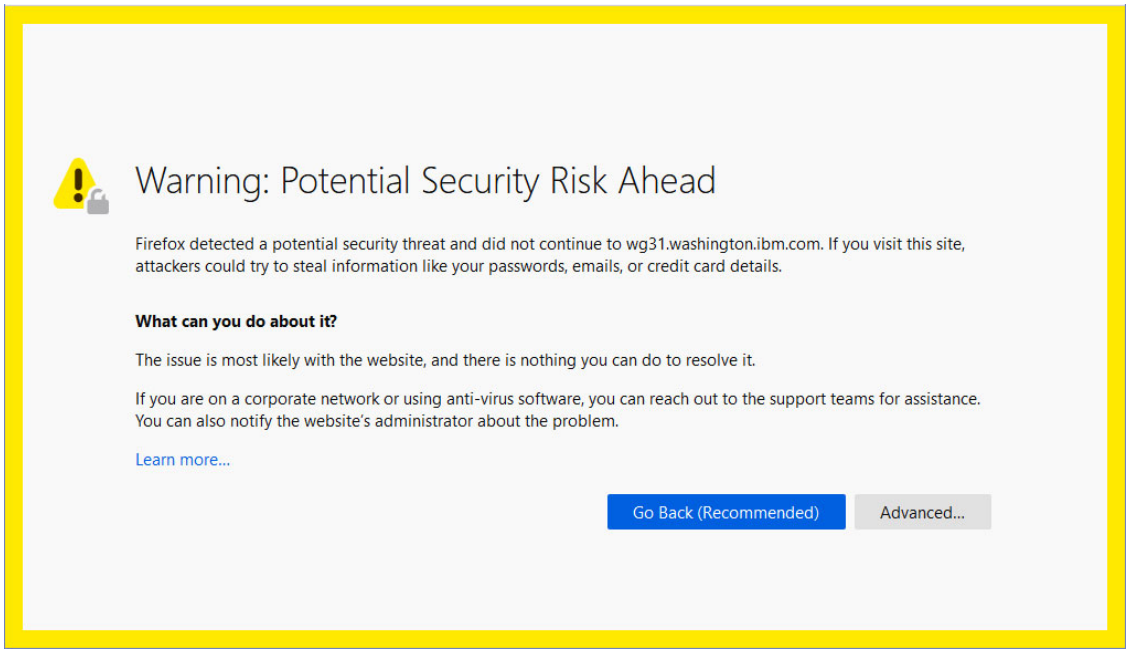
___10. Start Firefox and issue the following URL:

https://wg31.washington.ibm.com:9443/zosConnect/apis

1 So the certificate accepted earlier is cleared and you're forced to see the new SAF-created certificate.

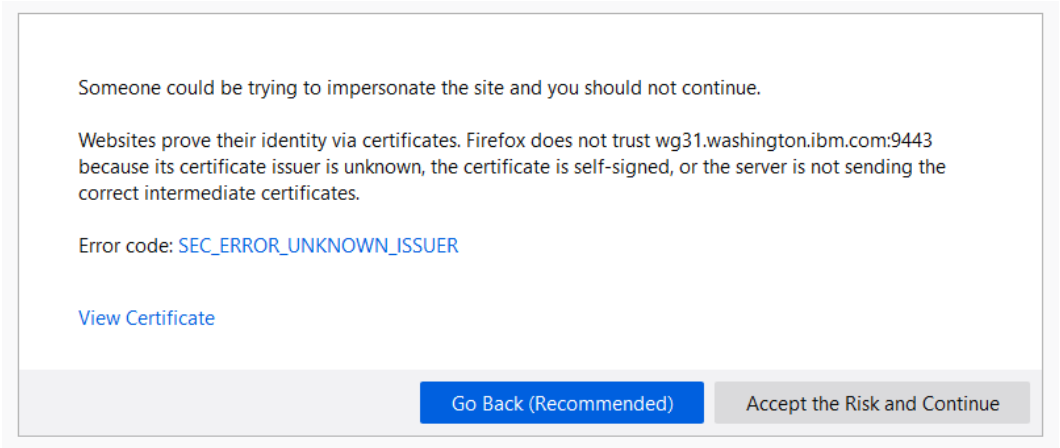
Mitch Johnson (mitchj@us.ibm.com)

___11. You will be challenged by Firefox because the digital certificate used by the Liberty z/OS server is a self-signed RACF certificate and the browser does not recognize RACF signed certificates. Click on the **Advanced** button to continue.



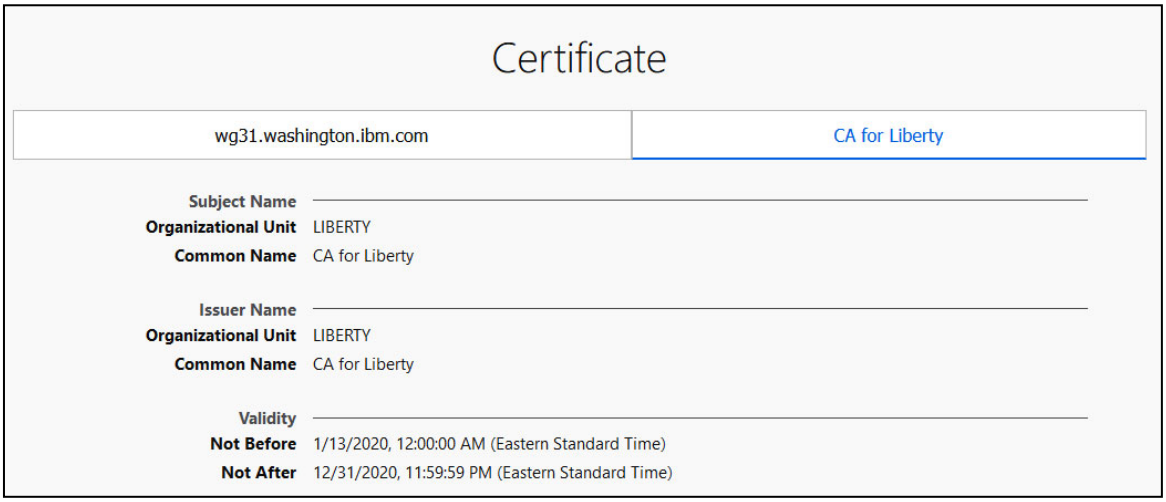
Tech-Tip: This warning is because the personal certificate being sent by the Liberty server was signed by a certificate authority the Firefox browser did not recognize. This was expected in this case.

___12. Additional information and options will be display at the bottom of the screen.



Mitch Johnson (mitchj@us.ibm.com)

___13. Click on *View Certificate* to display details about the certificate.



This is the new SAF-based certificate being presented. The values you see here are from the RACF job you ran:

```
RACDCERT CERTAUTH GENCERT SUBJECTSDN(CN('CA for Liberty') -
  OU('LIBERTY')) WITHLABEL('Liberty CA') TRUST -
  SIZE(2048) NOTAFTER(DATE(2018/12/31))
```

Note: The certificate is still unverified, but it is different from before. That means it's no longer using the Liberty-generated certificate, but rather it is using the RACF-generated certificate in SAF.

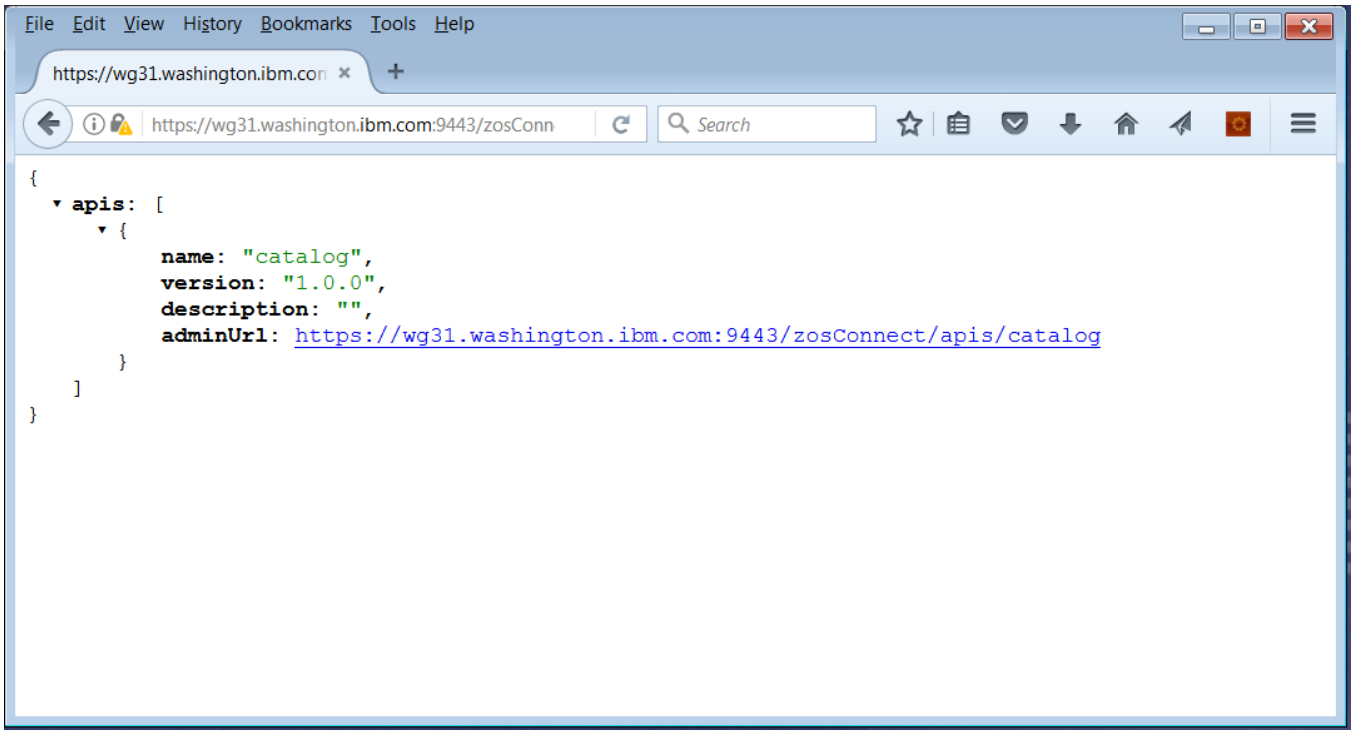
___14. Click the **Accept the Risk and Continue** button to continue.

___15. In the userid/password prompt window enter ***Fred*** and ***FRED***.

With SAF case does not matter. All userid and password values are stored in upper-case. Anything entered in lowercase or mixed is folded to uppercase and compared against the SAF registry.

Mitch Johnson (mitchj@us.ibm.com)

___16. You should see a familiar list of APIs:



Summary

One more element of the security infrastructure was moved from the "basic" Liberty implementation down into SAF. In this case it was the certificates for the establishment of the encrypted link.
In the "real world" a known Certificate Authority (such as VeriSign) would be used to sign the server certificate. In that case, the browser would trust the certificate based on the well-known CA and you would not get a challenge.

Mitch Johnson (mitchj@us.ibm.com)

Enabling mutual authentication using TLS

As the server is configured only the server is sending its certificate during the handshake. In this section the configuration will be changed to require the client to provide a personal certificate for authentication.

___1. Stop the the *BAQSTRT* z/OS Connect server, e.g. ***P BAQSTRT***.

___2. Use the ***VA*** (View ASCII) line command to open the *keyring*M*utual.xml* file in browse mode. You should see:

```
<?xml version="1.0" encoding="UTF-8"?>
<server description="mutual security">

    <!-- Enable features -->
    <featureManager>
         <feature>transportSecurity-1.0</feature>
    </featureManager>

    <sslDefault sslRef="DefaultSSLSettings"
      outboundSSLRef="DefaultSSLSettings" />

    <ssl id="DefaultSSLSettings"
       keyStoreRef="CellDefaultKeyStore"
       trustStoreRef="CellDefaultKeyStore"
       clientAuthenticationSupported="true"
       clientAuthentication="true"/>          1

    <keyStore id="CellDefaultKeyStore"
       location="safkeyring:///Keyring.LIBERTY"
       password="password" type="JCERACFKS"
       fileBased="false" readOnly="true" />

</server>
```

Notes

[1] Client authentication, e.g. mutual authentication is enabled.

Mitch Johnson (mitchj@us.ibm.com)

___3. Change the *sever.xml* by replacing the include for the file *keyring.xml* (see below) with an include for the file *keyring*M*utual.*

<include location="${shared.config.dir}/keyringMutual.xml"/>

```
<?xml version="1.0" encoding="UTF-8"?>
<server description="new server">
<include location="${shared.config.dir}/ipic.xml" />
<include location="${shared.config.dir}/safSecurity.xml" />
<include location="${shared.config.dir}/keyringMutual.xml" />
<include location="${shared.config.dir}/groupAccess.xml" />
```

When you ran the **ZCEETLSS** job earlier, one of the things it did was generate and export a client certificate for Fred.

___4. Restart the server with MVS command ***S BAQSTRT.***

___5. Go to data set *USER1.ZCEE30.CNTL* data set and browse the **ZCEETLSX** member. You will see something like this:

```
RACDCERT ID(FRED) EXPORT(LABEL('FRED')) -
  DSN('USER1.FRED.P12') FORMAT(PKCS12DER) PASSWORD('secret')

RACDCERT ID(USER1) EXPORT(LABEL('USER1')) -
  DSN('USER1.USER1.P12') FORMAT(PKCS12DER) PASSWORD('secret')

RACDCERT CERTAUTH EXPORT(LABEL('Liberty CA')) -
  DSN('USER1.CERTAUTH.PEM')
```

Tech-Tip: In summary, these commands export from RACF the personal certificates for identities FRED and USER1 and they export the certificate authority certificate used to sign these personal certificates (e.g. Liberty CA).
The personal certificates are exported encoded with their private keys included (PKCS12DER) and the exported files ae password protected. For the CA certificate, the certificate is exported in Privacy Enhanced Mail (PEM) format with no private key included.

___6. Submit that job and look for the *MAXCC=0000* indicator of success.

In the following steps you will download that certificate so you can import into Firefox and use them with cURL and Postman.

___7. On the Windows desktop, open a command prompt.

___8. Enter the command: ***cd c:\z\admin***

___9. Enter the command below to download the public Liberty CA certificate:

curl --user user1:user1 ftp:/wg31.washington.ibm.com/CERTAUTH.PEM --use-ascii -o certauth.pem

Mitch Johnson (mitchj@us.ibm.com)

___10. Enter the command below to download FRED's personal certificate:

curl --user user1:user1 ftp:/wg31.washington.ibm.com/fred.p12 -o fred.p12

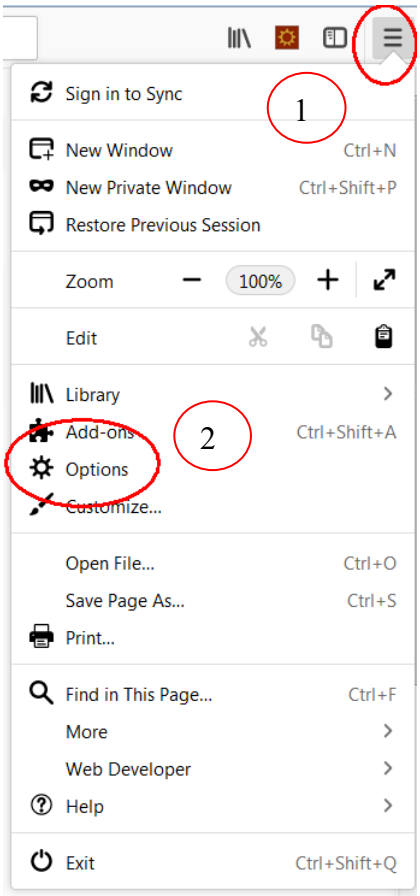
___11. Enter the command below to download USER1's personal certificate:

curl --user user1:user1 ftp:/wg31.washington.ibm.com/user1.p12 -o user1.p12

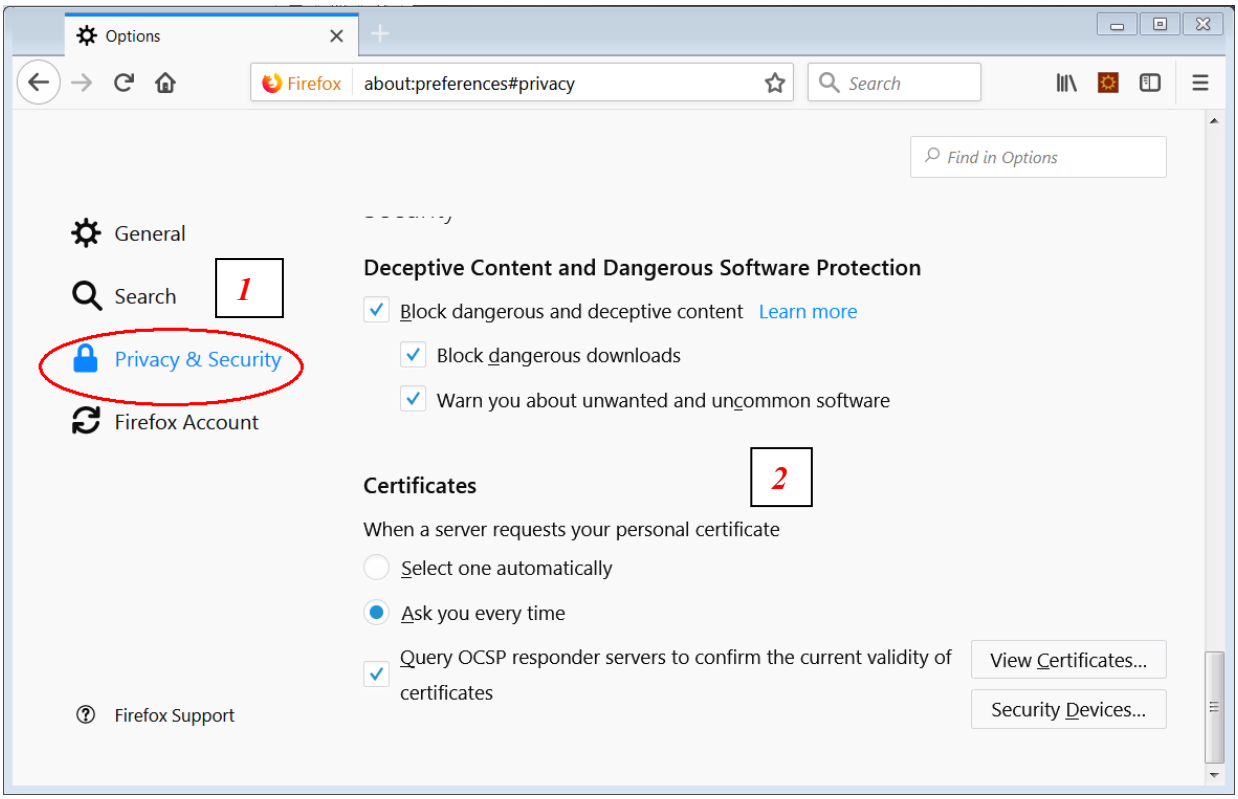
With the certificates downloaded, the next step is to import them into Firefox.

Using mutual authentication with Firefox

___1. In Firefox, click on the to the *Open Menu* (1) icon and select the *Options* (2) tool.

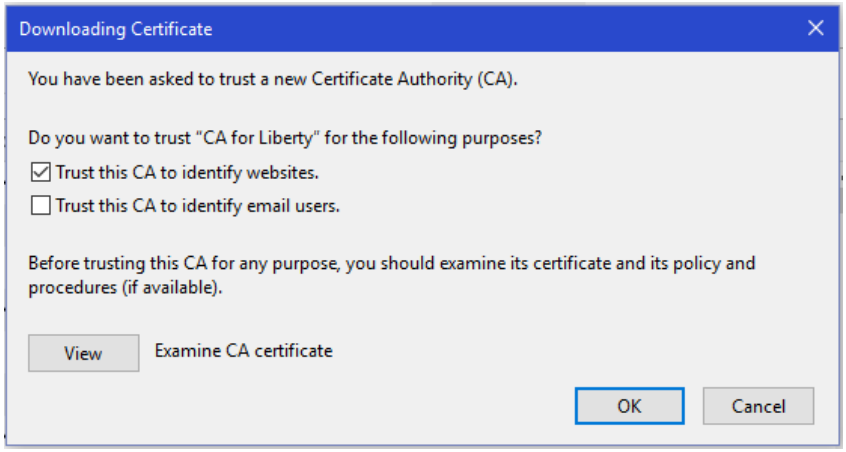


___2. Click on *Privacy & Security* (1) then scroll down to the *Certificates* (2) tab:



___3. Then click the **View Certificates** button.

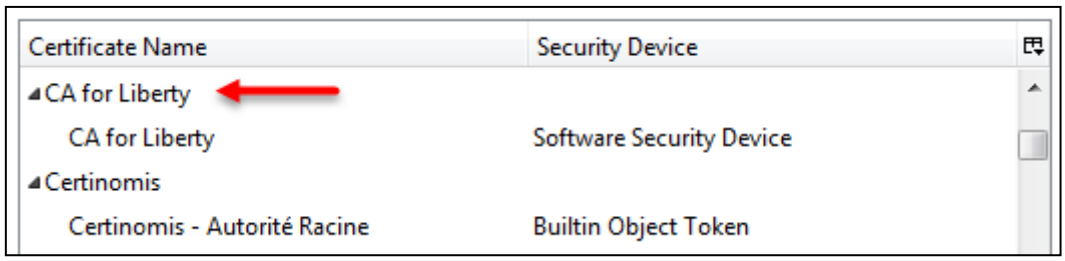
___4. Then click on the *Authorities* tab, and the **Import** button.



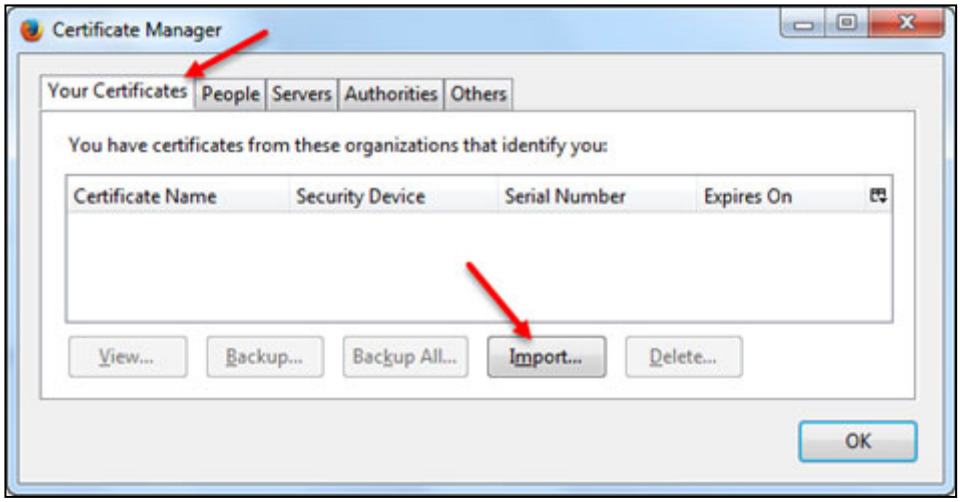
___5. Navigate to the *c:\z\admin directory* and double-click on the **certauth.pem** file.

___6. Then check the *Trust this CA to identify websites* box and click **OK:**

___7. Verify the certificate has been imported by scrolling down and looking for the "CA for Liberty" certificate in the list:

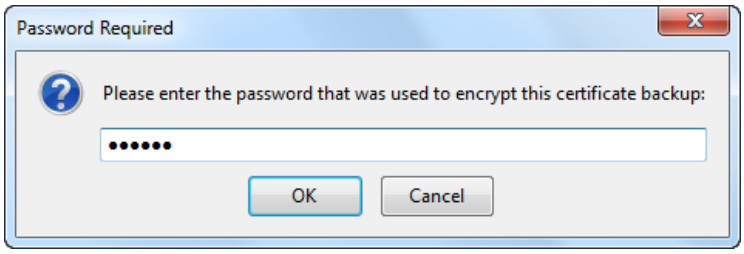


___8. Next, click the *Your certificates tab* and then the **Import** button:



___9. It should open in the *c:\z\admin* directory from before, but if not, navigate to that location. Locate the **fred.p12** certificate and double-click on it.

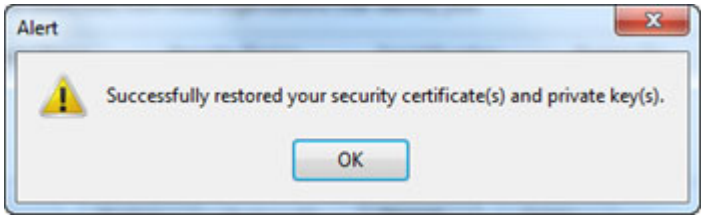
___10. A window will appear asking you to enter the password for the certificate:



Enter the value[2] ***secret*** and click **OK**. You should see confirmation:

2 If interested, look in the ZCEETLSS job to see where this value is specified on the certificate export.

Mitch Johnson (mitchj@us.ibm.com)

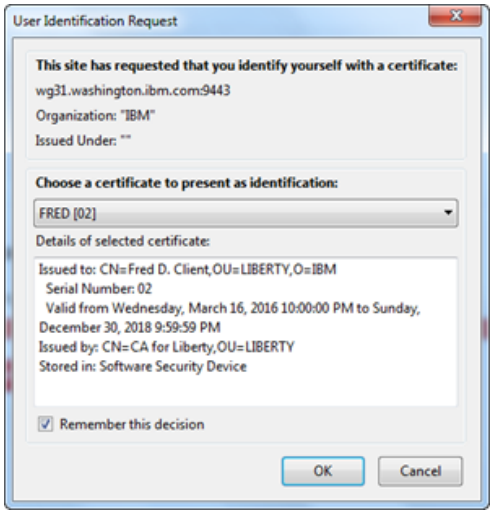


___11. Click **OK** to clear the confirmation, then click **OK** to close the certificate manager panel, **OK** to close the options panel, and then close *all instances* of your Firefox browser.

___12. Restart your server with ***S BAQSTRT***.

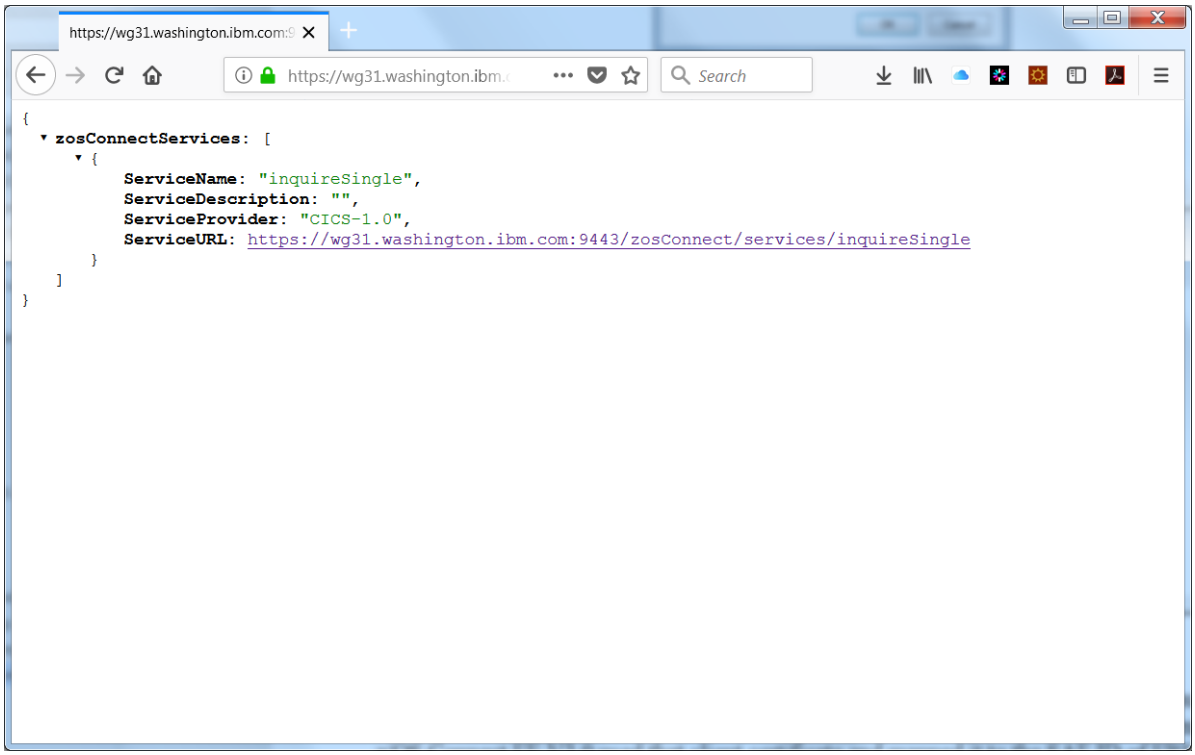
___13. Start Firefox and go to URL ***https://wg31.washington.ibm.com:9443/zosConnect/services***
Click the **Send** button.

___14. You will be prompted for which client certificate you wish to use:



You only have one, and it's selected ... so click **OK**.

___15. You should see the list of installed services:



Mitch Johnson (mitchj@us.ibm.com)

Using mutual authentication with cURL

___1. Use the *Command Prompt* icon on the desktop to open a DOS command prompt session.
___2. In the session use the change directory (cd) command to go to directory c:\z\admin, e.g.

cd c:\z\admin

___3. Paste the command below at the command prompt and press **Enter**.

curl -X put --cacert certauth.pem --cert user1.p12:secret --cert-type P12 https://wg31.washington.ibm.com:9443/zosConnect/services/inquireSingle?action=start

___4. You should see the response below:

```
{"errorMessage":"BAQR0406W: The zosConnectAuthorization interceptor encountered
an error while processing a request for service inquireSingle under request URL
https://wg31.washington.ibm.com:9443/zosConnect/services/inquireSingle.","errorD
etails":"BAQR0409W: User USER1 is not authorized to perform the request."}
```

The USER1 identity is determined by the client certificate specified in user1.p12.

___5. Paste the command below at the command prompt and press **Enter**.

curl -X put --cacert certauth.pem --cert fred.p12:secret --cert-type P12 https://wg31.washington.ibm.com:9443/zosConnect/services/inquireSingle?action=start

___6. You should see the response below:

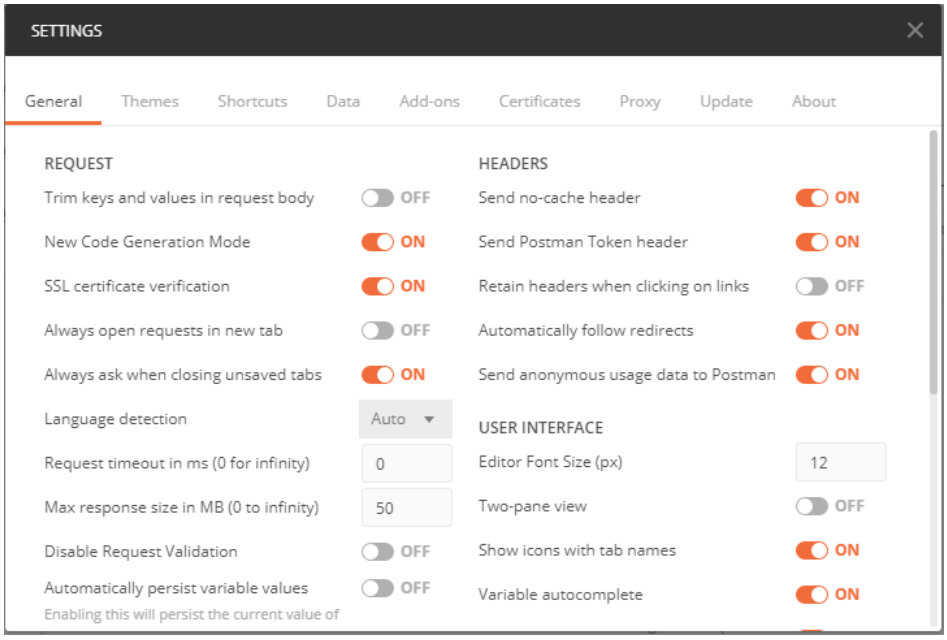
```
{"zosConnect":{"serviceName":"inquireSingle","serviceDescription":"","servicePro
vider":"CICS-1.0","serviceURL":"https://wg31.washington.ibm.com:9443/zosConnect/
services/inquireSingle","serviceInvokeURL":"https://wg31.washington.ibm.com:9443
/zosConnect/services/inquireSingle?action=invoke","dataXformProvider":"zosConnec
tWVXform-1.0","serviceStatus":"Started"}}
```

The FRED identity is determined by the client certificate specified in fred.p12 and FRED has administrator authority.

Mitch Johnson (mitchj@us.ibm.com)

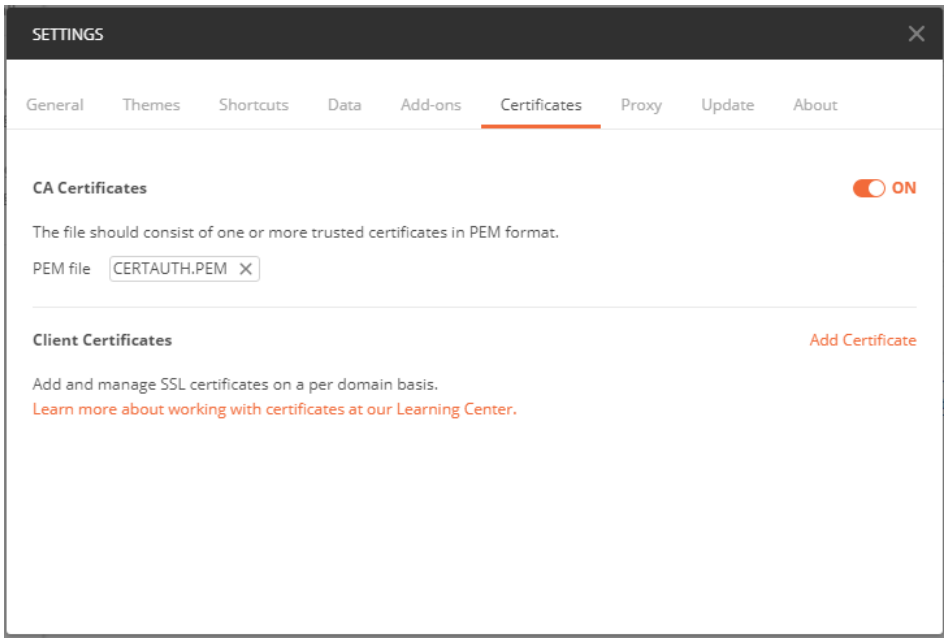
Using mutual authentication with Postman

___1. To test with Postman, do the following. Go into Postman and select the settings icon (the wrench on the tool bar) to open the *Settings* window.



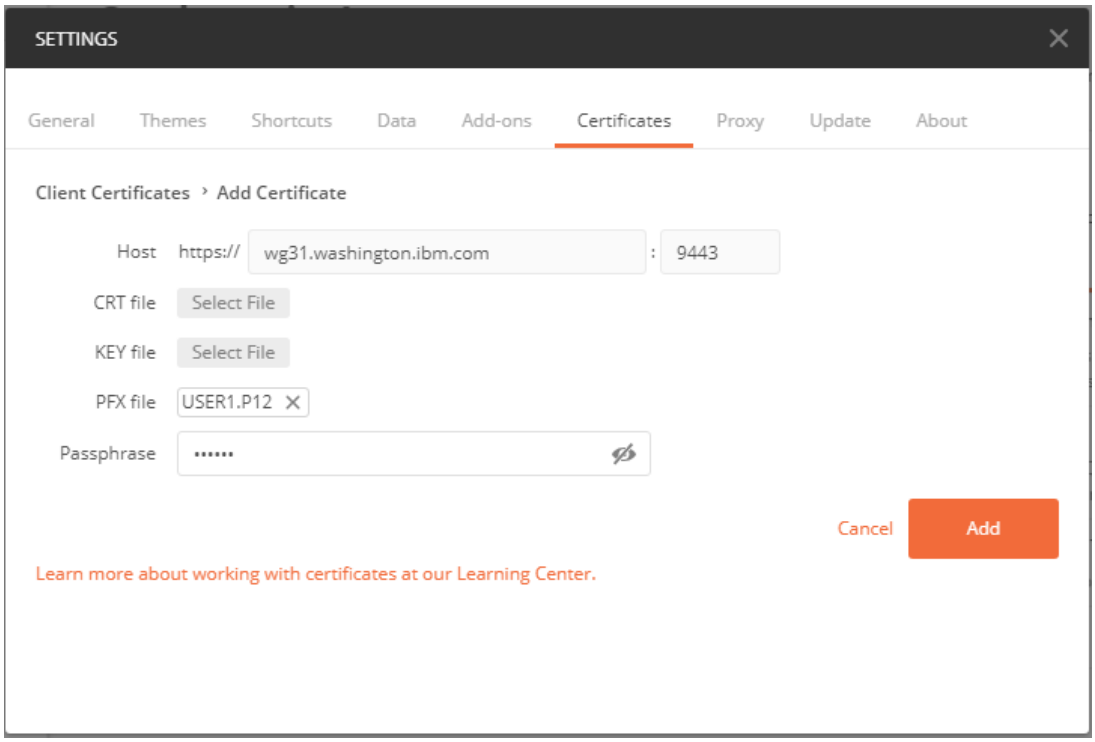
___2. In the *General* tab, turn SSL certificate verification on (see above).

___3. Go to the *Certificates* tab and turn on the **On** radio button. Use the *Select File* button to add the CA certificate to be *CERTAUTH.PEM* file downloaded earlier.

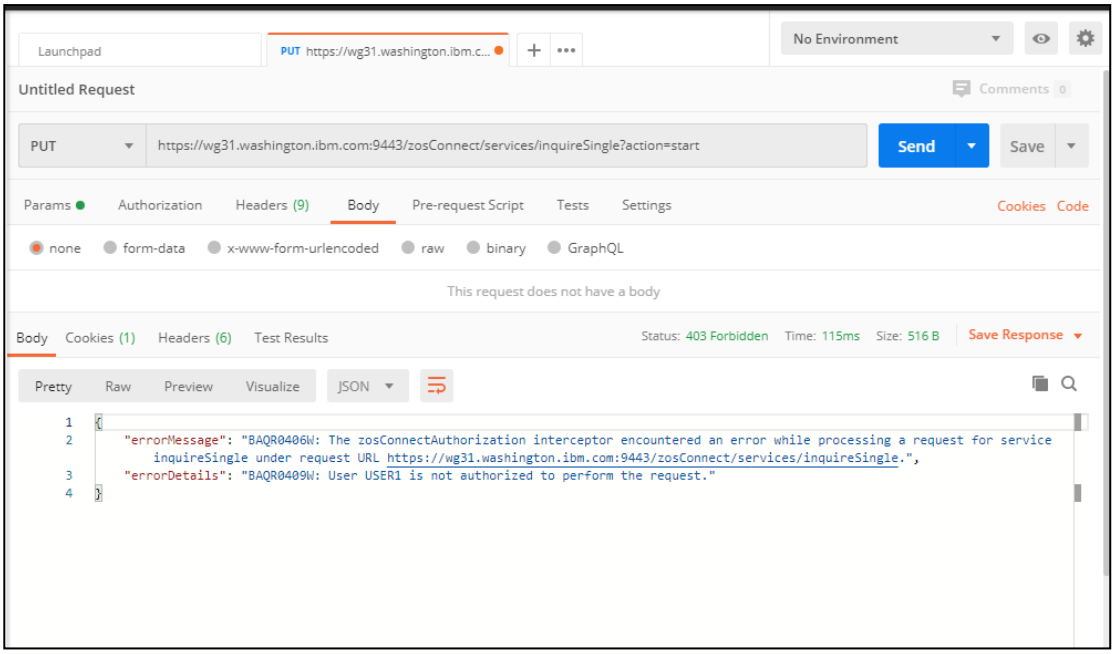


Mitch Johnson (mitchj@us.ibm.com)

___4. Click the *Add Certificate* and enter ***wg31.washington.ibm.com*** and port ***9443*** for the *Host* as shown below. Use the Select File button beside PFX file to select file *USER1.P12* downloaded earlier. Note the password is entered as the value in the *Passphrase* field.

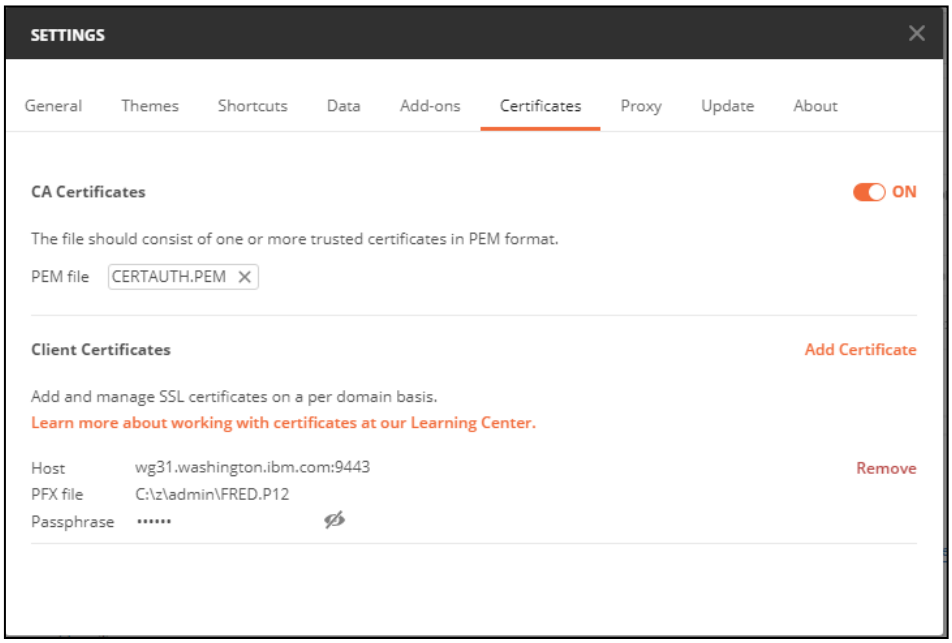


___5. Send a *PUT* request for URL ***https://wg31.washington.ibm.com:9443/zosConnect/services/inquireSingle?action=start*** and you should see the same results and as when using the cURL command.

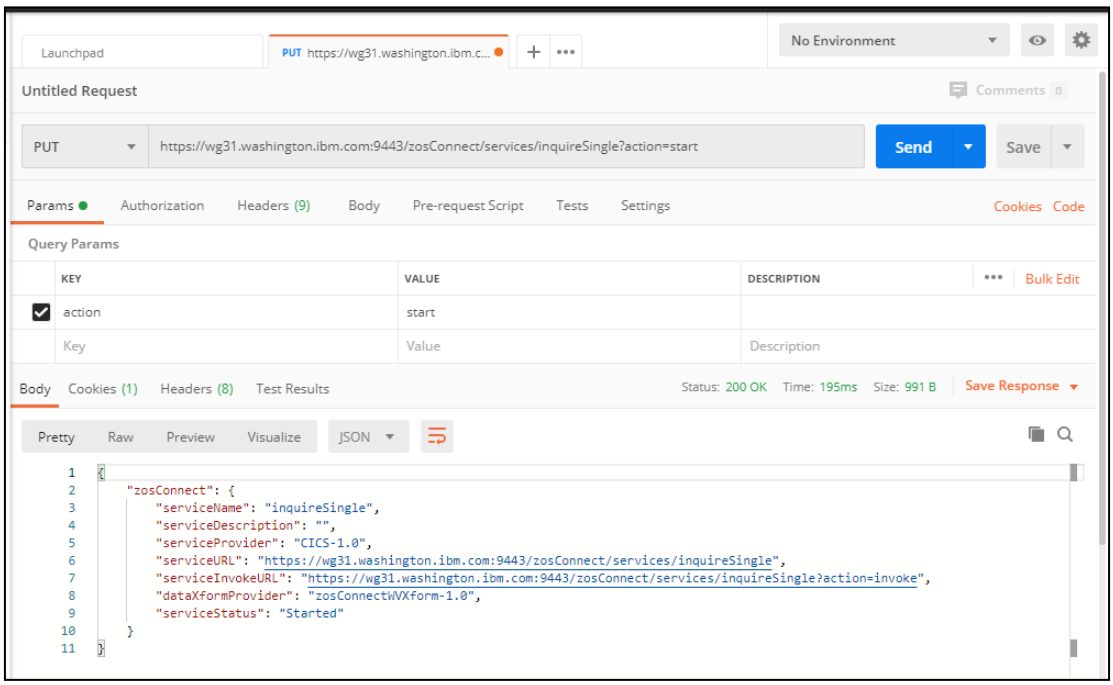


Mitch Johnson (mitchj@us.ibm.com)

___6. You cannot easily change personal certificates in Postman from one to another. Switching to a different personal certificate requires removing the current certificate and then adding the new certificate in the *Certificates* tab of the *Settings*.



___7. This will require starting Postman after the switch is made.

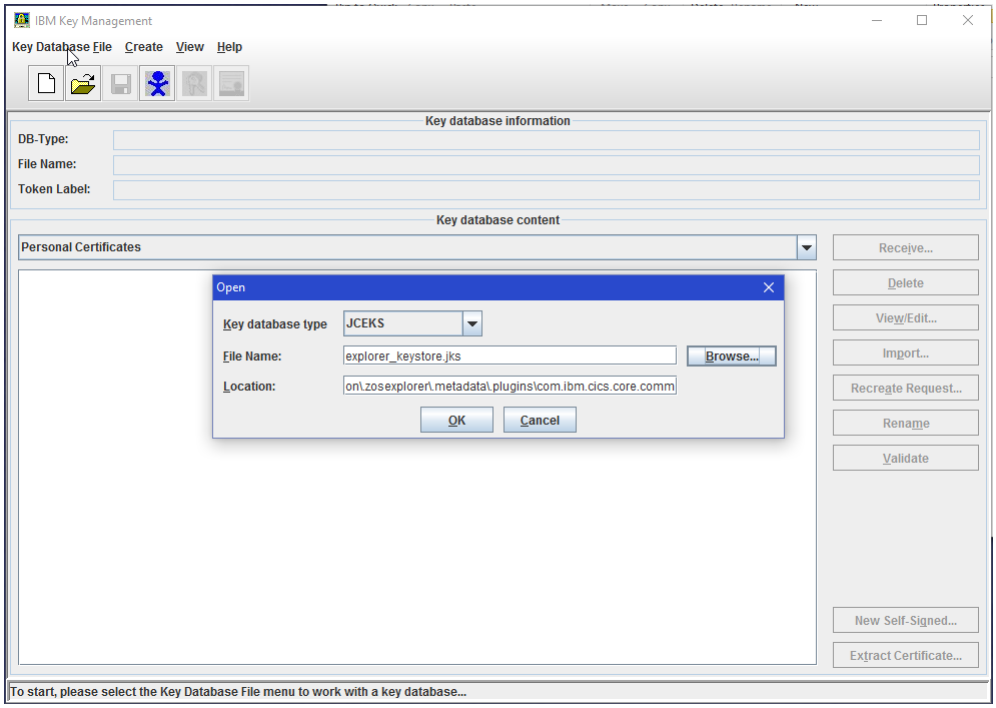


Using mutual authentication with the API Toolkit

Tech-Tip: The IBM Key Management tool is shipped with the IBM Java package which is shipped with most if not all IBM workstation products that uses Java. The executable is usually in subdirectory …/jdk/jre/bin. For example, on Windows with the IBM z/OS Explorer install, the executable can be found in directory *c:/Program Files/IBM/zOS Explorer/jdk/jre/bin*. This image has a shortcut for this executable on the desktop.

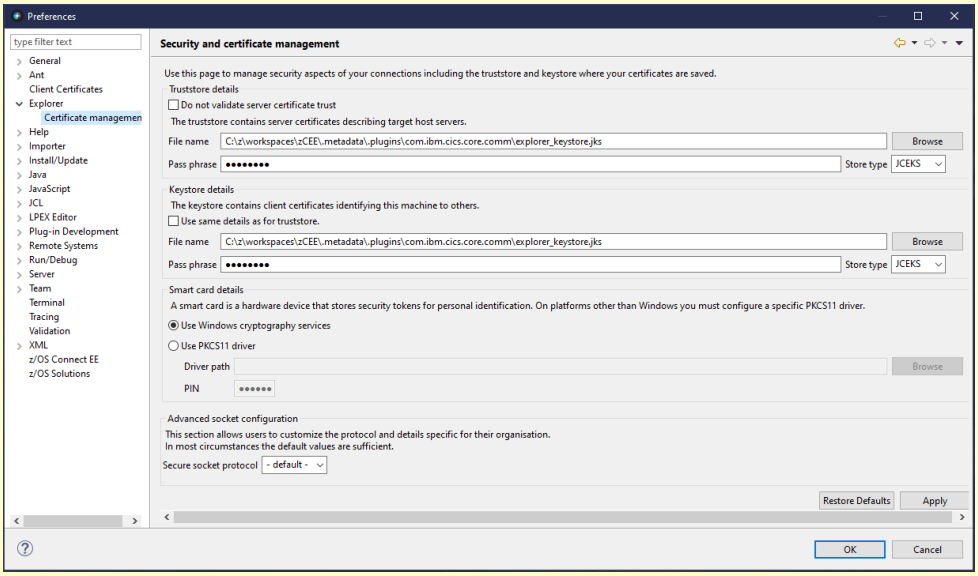
Eclipse tooling like the API Toolkit maintains their own trust store file in the workspace directory structure (e.g., each workspace has a different trust store). In this section, the IBM Key Management tool (iKeyman) will be used to import these same CA and personal certificates into the trust store file used by Eclipse. This will allow the use of TLS and mutual authentication between the toolkit and a server.

___1. Open the *iKeyman* tool on the desktop. On the tool bar, select *Key Database File* and select the *Open* option. Use the pull-down arrow beside *Key database type* to select a type of *JCEKS* and then use **Browse** button to select the *explorer_keystore.jks* file (see below).



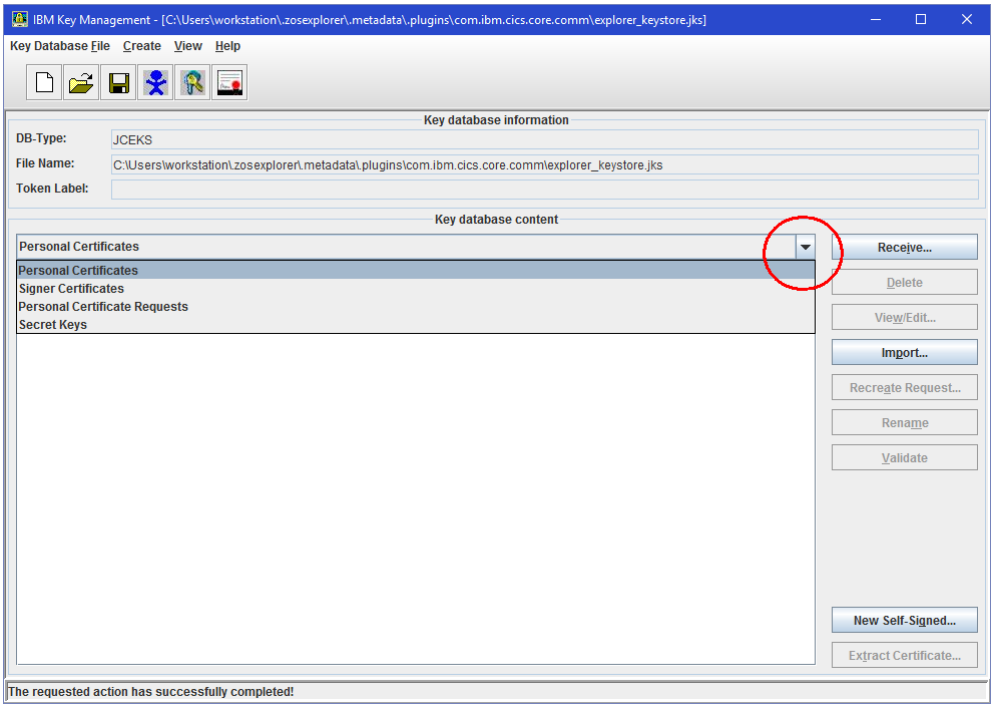
Mitch Johnson (mitchj@us.ibm.com)

Tech-Tip: For the z/OS Connect API Toolkit, the trust store file is the file *explorer_keystore.jks* in the Eclipse workspace directory *..\.metadata\.plugins\com.ibm.cics.core.comm (the ellipses represent the workspace directory)*. The full path and file name of the key stores can be obtained by selecting the *Window* tool on the toolbar then the *Preferences -> Explorer -> Certificate management* to display the current *Truststore* and *Keystore* file names.



___2. Click the OK button and enter ***changeit*** as the password.

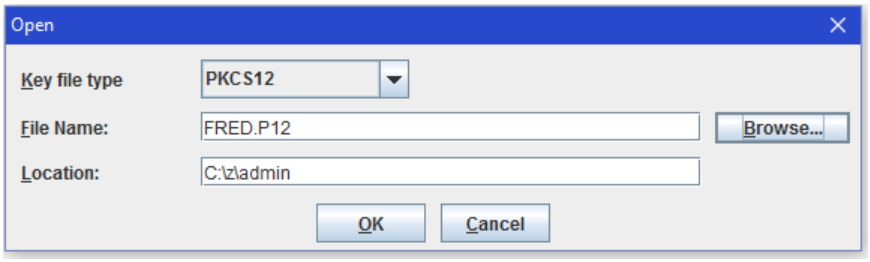
___3. This will display the screen below. The drop down (circle in red below) can be used to switch from *Personal Certificates* to *Signer Certificates* (as known as CA certificate)



Mitch Johnson (mitchj@us.ibm.com)

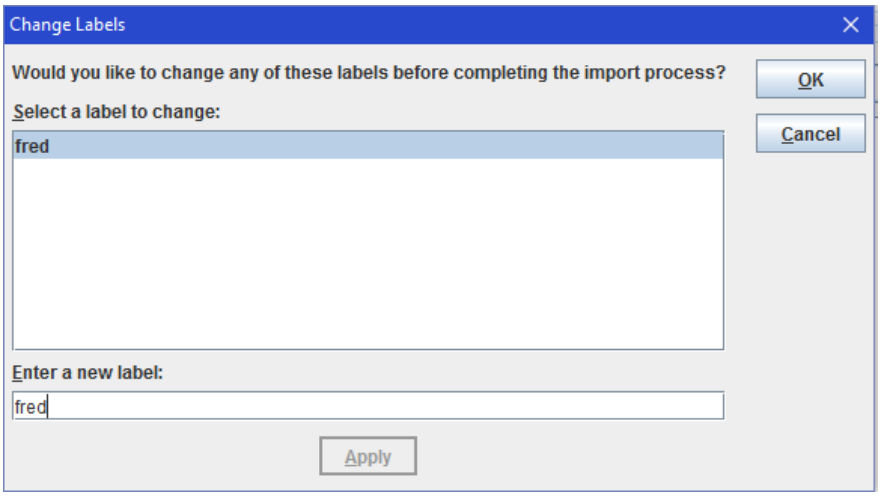
___4. There are no personal certificates currently in this key store, so click the **Import** button to start the import process.

___5. On the next window, use the pull-down arrow and select *PKCS12* for the *Key file type*. Use the Browse button to navigate to directory c:\z\admin (where the certificates were previously downloaded) and select file *FRED.P12*. Click **OK** to continue.



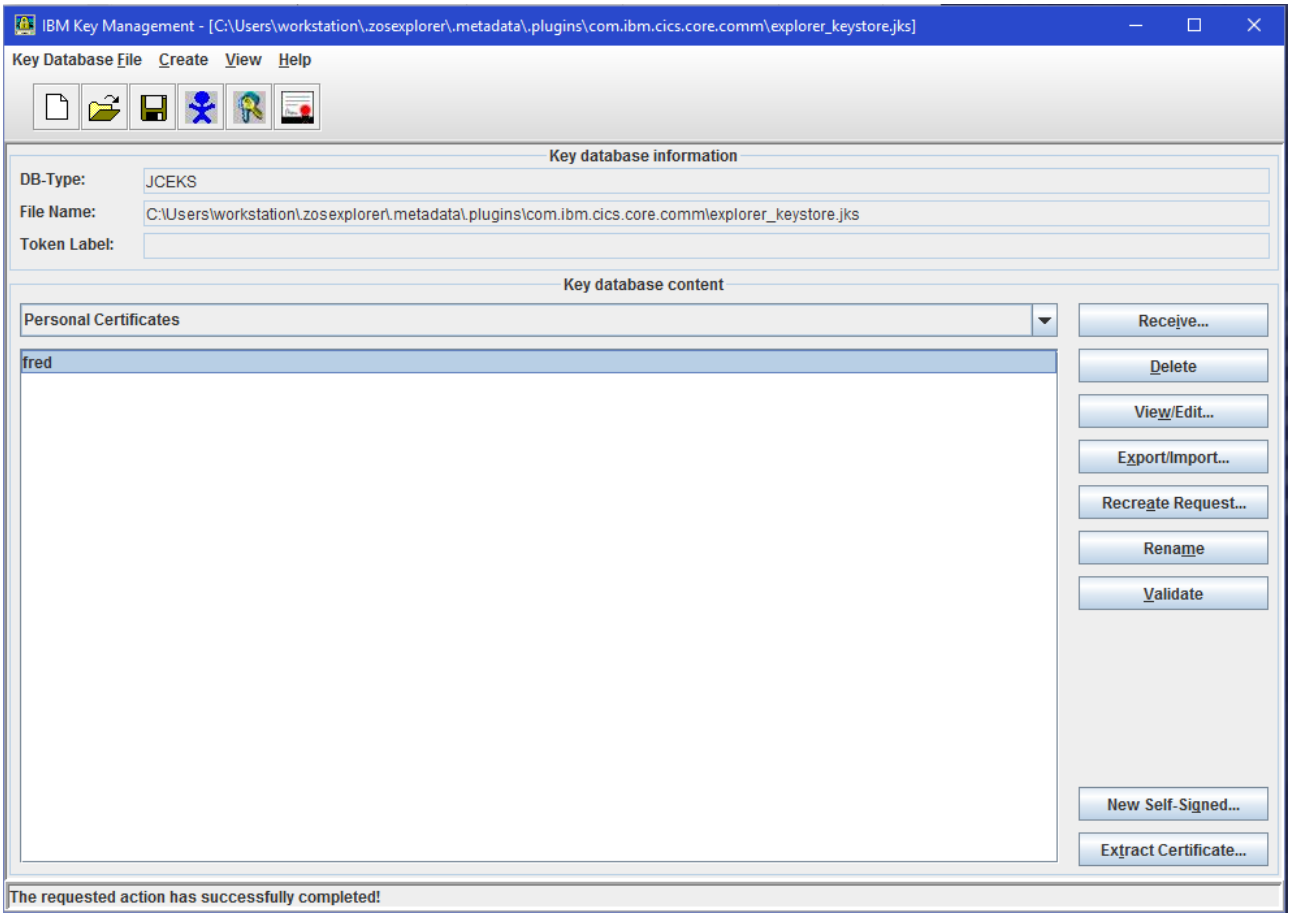
___6. This file is password protected (it contains a private key), so enter the password used when the certificate was exported from RACF(e.g. **secret**).

___7. You will be given an opportunity to change the certificate label. This will not be done in the case so click **OK** to continue.



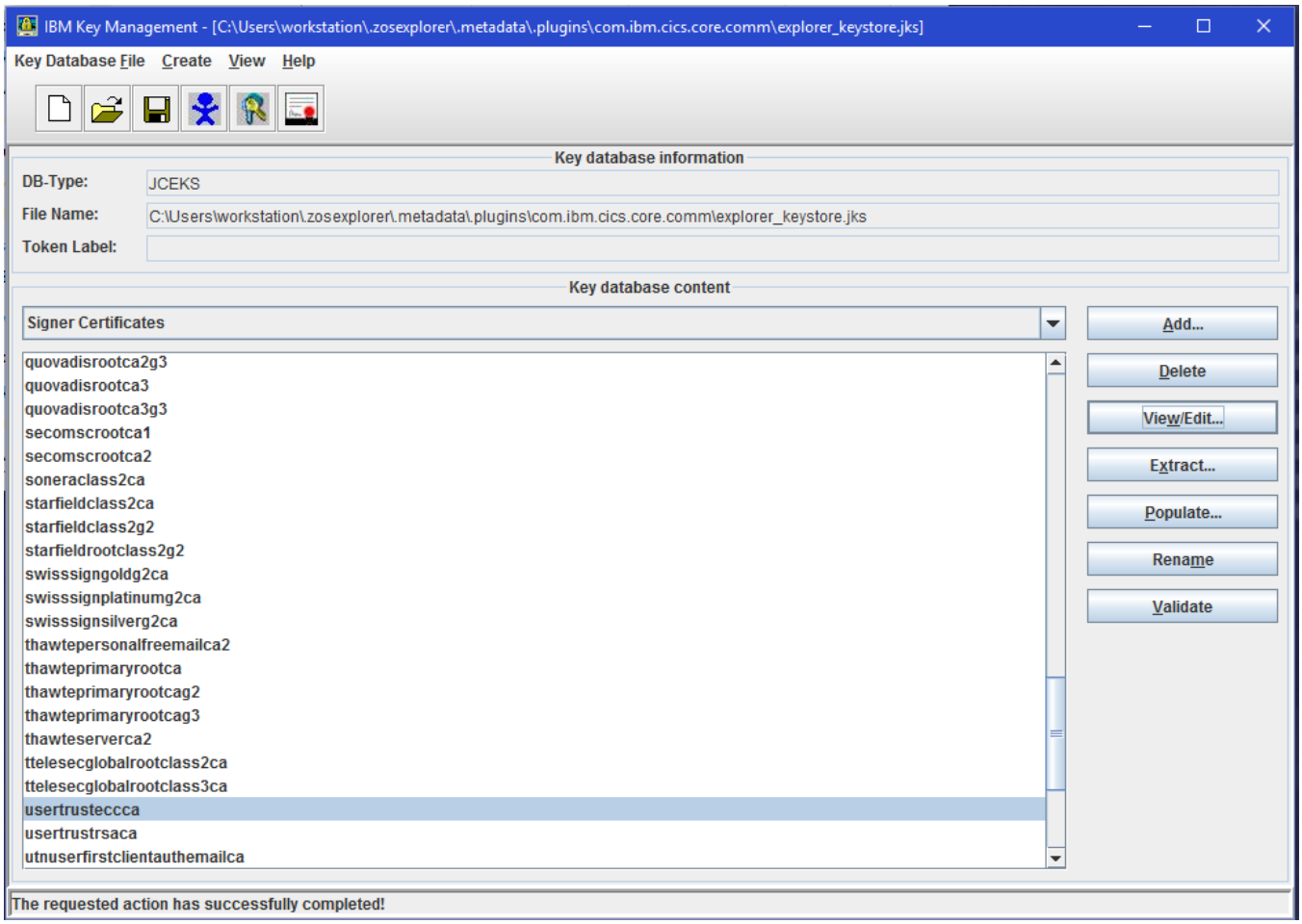
Mitch Johnson (mitchj@us.ibm.com)

___8. The imported personal certificate will now show up in the list of *Personal Certificates*. On the right-hand side of the screen, there are various tools available for managing personal certificate. Explore the *View/Edit* and *Validate* tools. The former provides access to the details of the certificate while the latter performs a simple validation of the certificate.

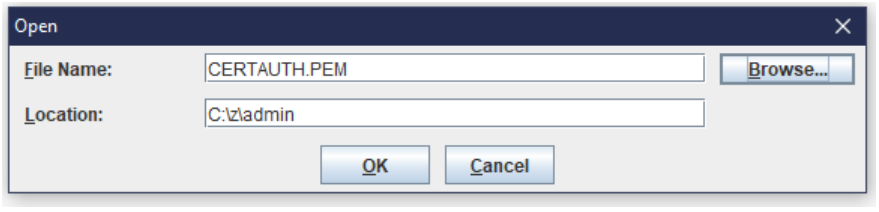


Mitch Johnson (mitchj@us.ibm.com)

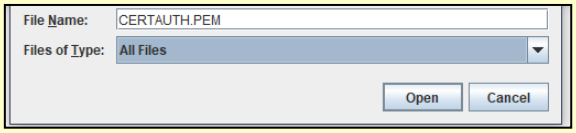
___9. Switch to the *Signer Certificates* view and you will see an extensive list of well know certificate authorities already present in the key store. Well known certificates authority certificates were automatically added when the trust store was created.



___10. The next step is to add the signer certificate used to sign the server certificate sent by the z/OS Connect server during the handshake. Click the **Add** button. Use the **Browse** button to navigate to directory c:\z\admin and select the certificate authority certificate exported earlier from RACF and downloaded to this directory. Click **OK** to continue.



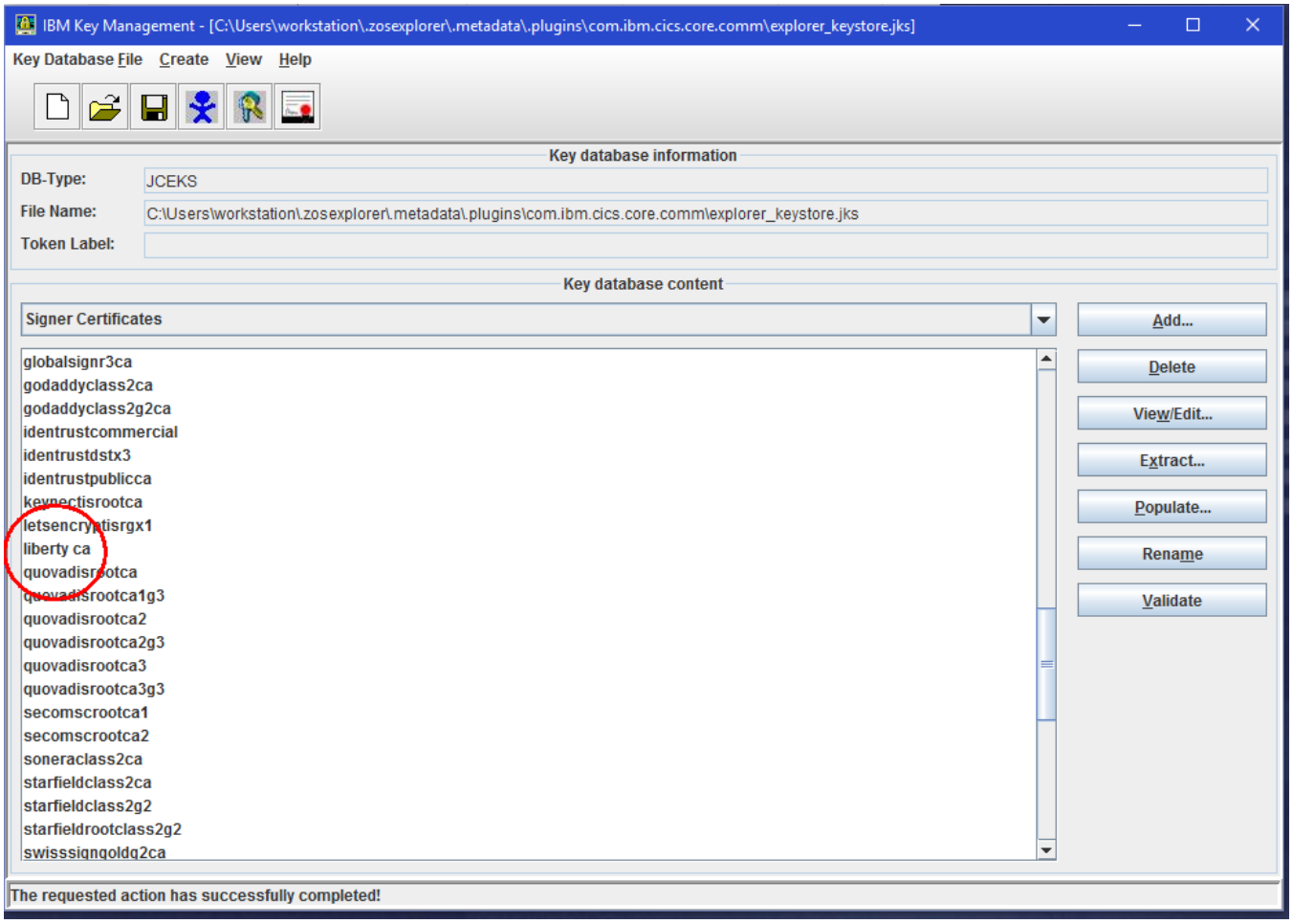
Tech-Tip: After clicking the **Browse** button, you may have to use the pull-down arrow to select *All Files*.



Mitch Johnson (mitchj@us.ibm.com)

___11. Enter a meaningful label and click OK to continue.

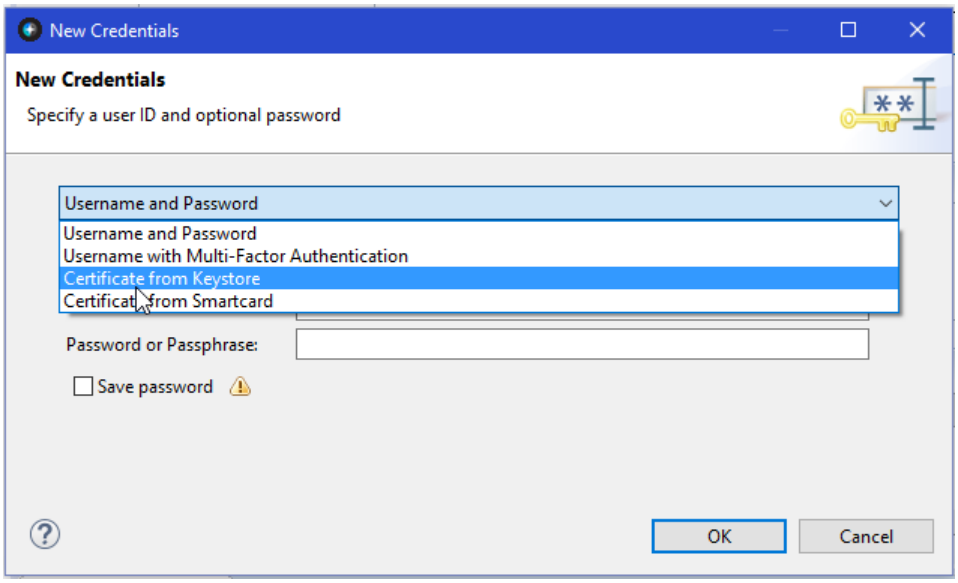
___12. The imported certificate should now show up in the list. Use the tools on the right-hand side to explore this certificate. Now when the zCEE server sends a server certificate, this Eclipse instance will automatically validate the server certificate and accept it as valid.



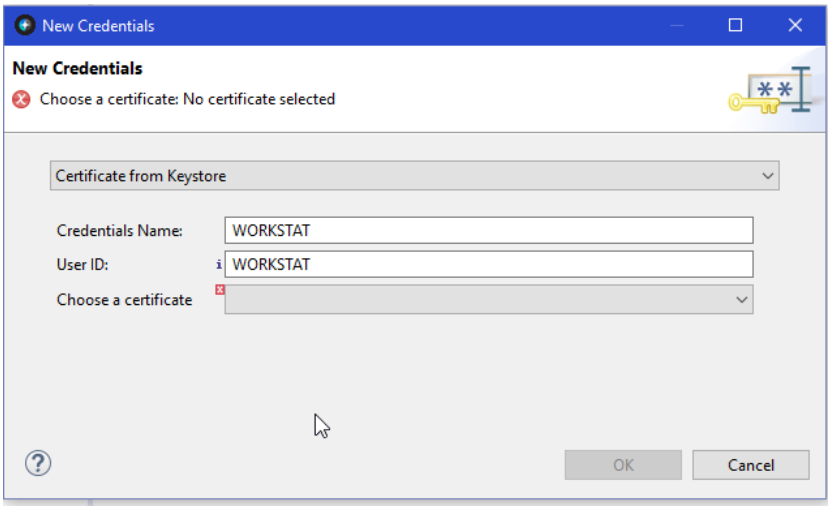
___13. Close the *IBM Key Management* tool and go back to *IBM Explorer for z/OS*

Mitch Johnson (mitchj@us.ibm.com)

___14. The new credentials for the personal certificate added to the key store by IBM Key Management now need to be added in the tool kit. Click the **Add** button in the credentials pane and use the pull-down arrow to select *Certificate from Keystore*.

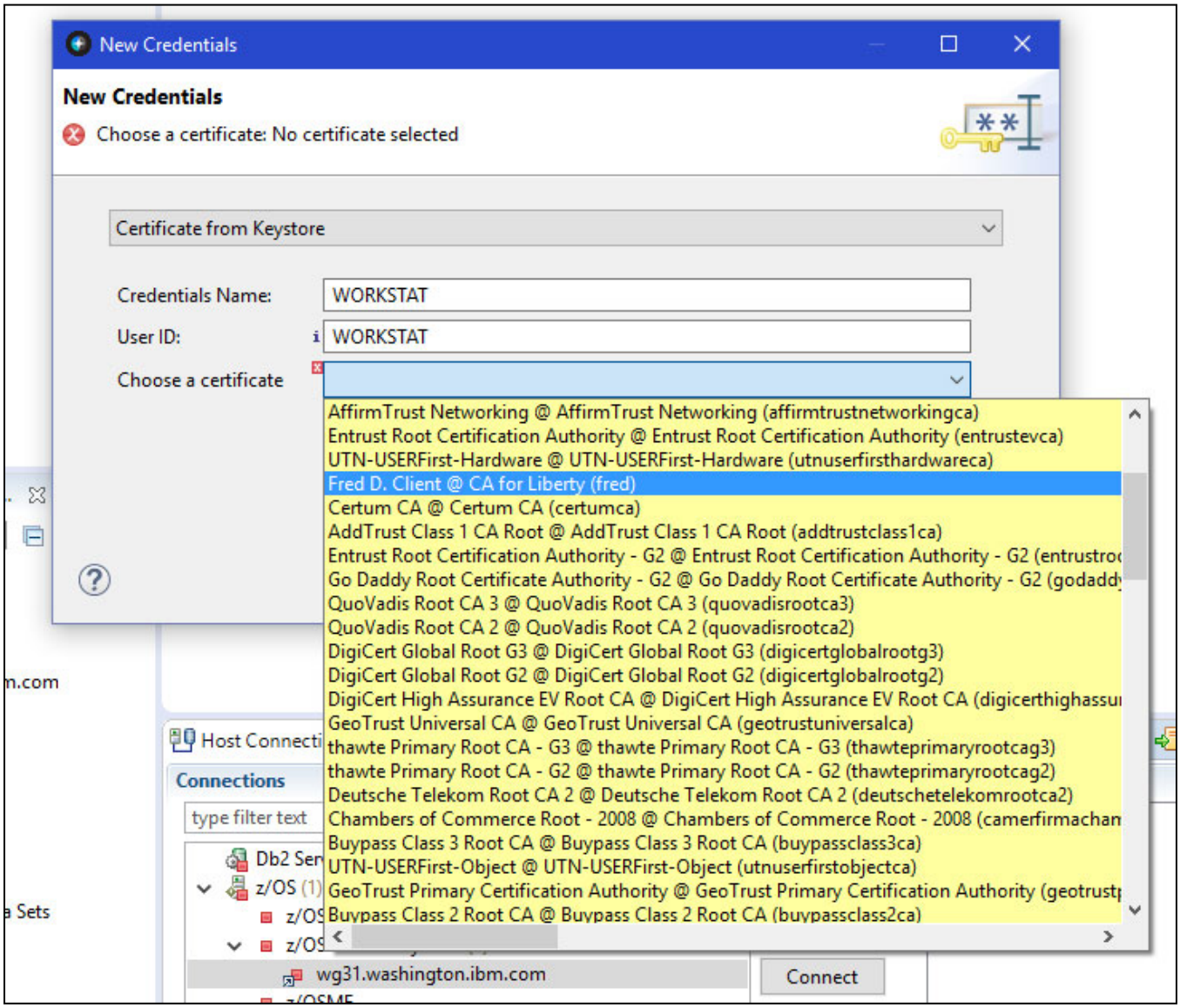


___15. Use the pull-down arrow beside *Choose a certificate* to display all the certificates in the Eclipse key store.



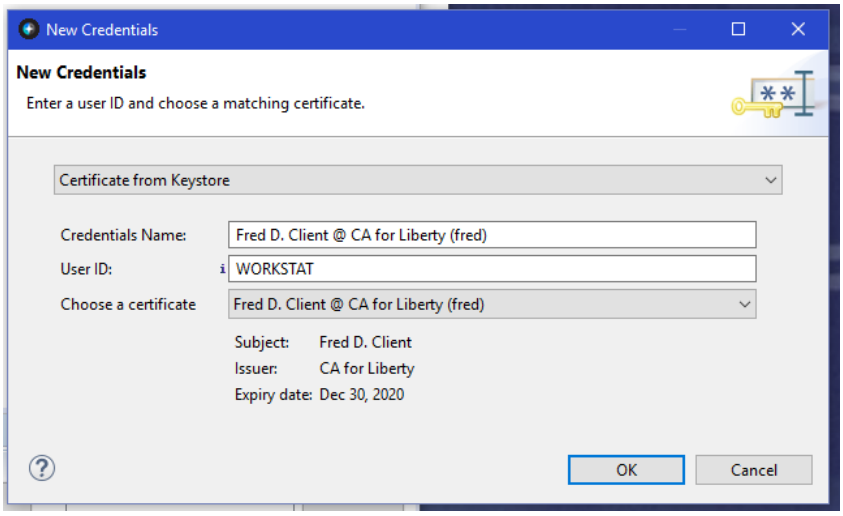
Mitch Johnson (mitchj@us.ibm.com)

___16. Next select the personal certificate to be used for mutual authentication.

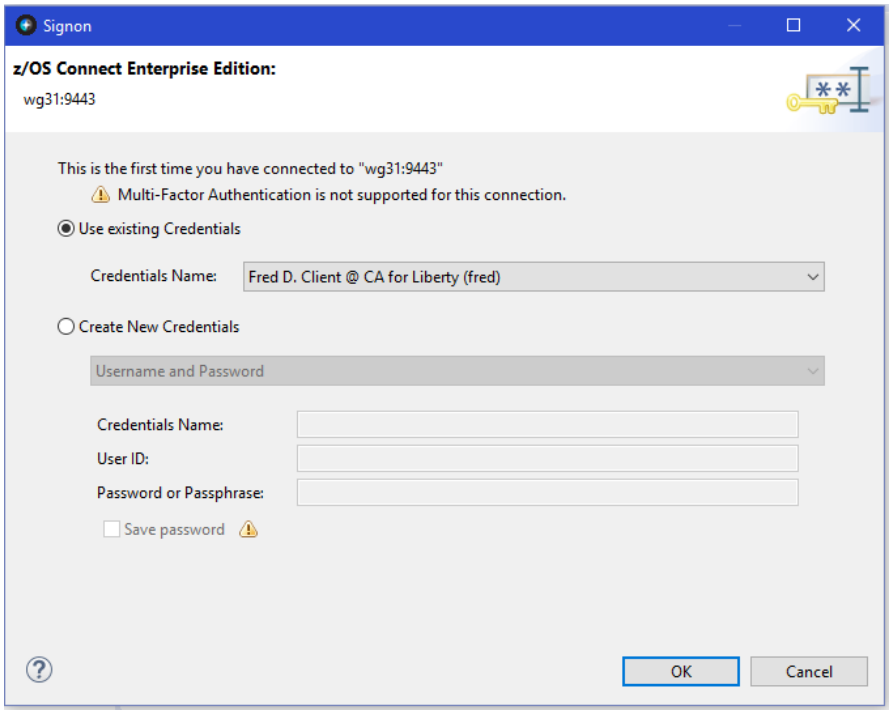


Mitch Johnson (mitchj@us.ibm.com)

___17. Click **OK** to save these credentials

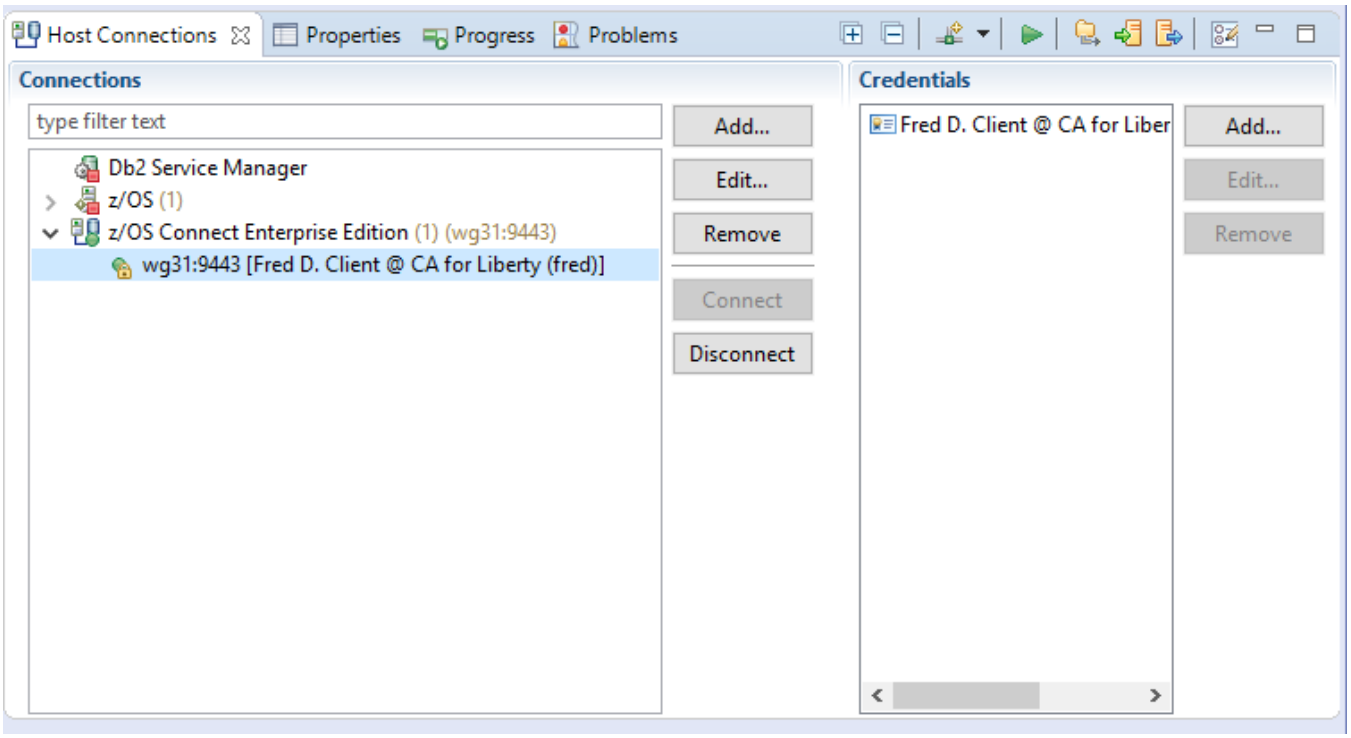


___18. And use the new credentials to connect to the z/OS Connect server



Mitch Johnson (mitchj@us.ibm.com)

___19. The connection under Host Connection should reflect the use of the client certificate as shown below.



Summary

In that exercise you did not see the basic authentication panel like you did before. In the web browser you were prompted for a client certificate (because of an option that defaulted when you imported the client certificate). The z/OS Connect server used that client certificate and mapped it to the SAF ID of FRED. That's what allowed you to invoke the */zosConnect/services* API and get the list of services. In the cURL and Postman examples, the client certificate specified by the *–cert* flag or via the settings determined which identity was used for authorization checking in z/OS Connect because *clientAuthenication* was enabled.

Mitch Johnson (mitchj@us.ibm.com)