IBM z/OS Connect EE V3.0

zCEE Basic Configuration - Hands-on Lab

IBM WSC Wildfire Team
IBM z Systems

Lab Version Date: April 18, 2019

Table of Contents

Important: On the desktop there is a file named *Basic Configuration CopyPaste.txt*. This file contains commands and other text used in this workshop. Locate that file and open it. 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.

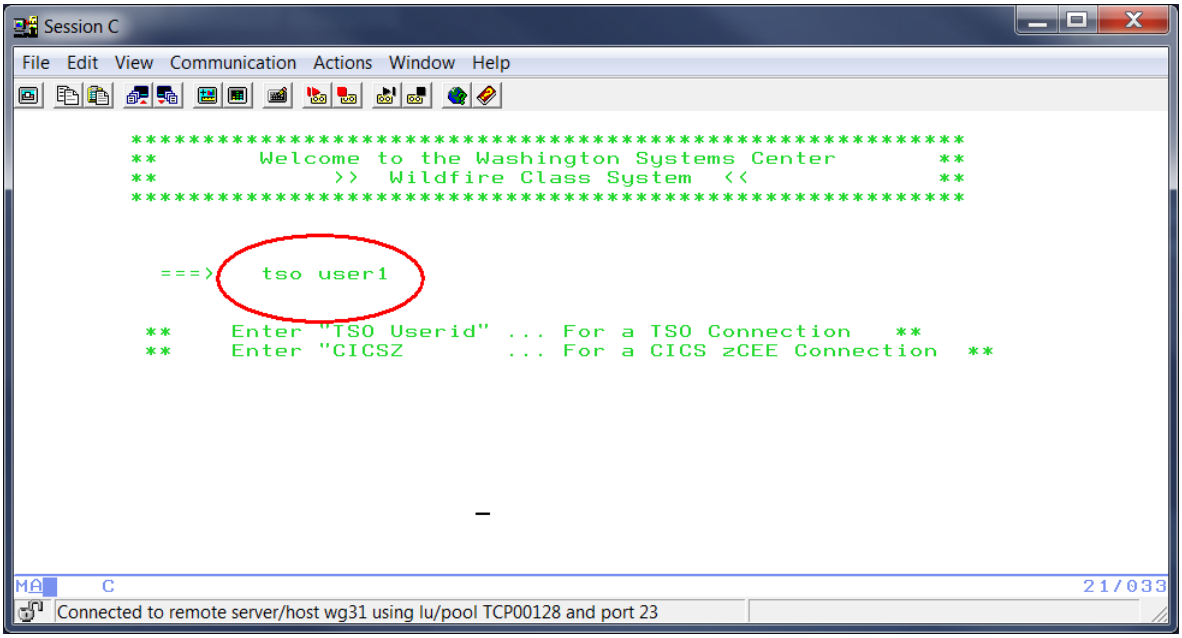
Mitch Johnson (mitchj@us.ibm.com)

Setup and Service Definitions

Run the jobs to setup RACF framework for server runtime

____ 1. Open the *WG31* icon on the workstation desktop. This will start a 3270-terminal session to your z/OS system.

> **Tech-Tip:** Desktop tools can be opened either by double clicking the icon or by selecting the icon and right mouse button clicking and then selecting the *Open* option.



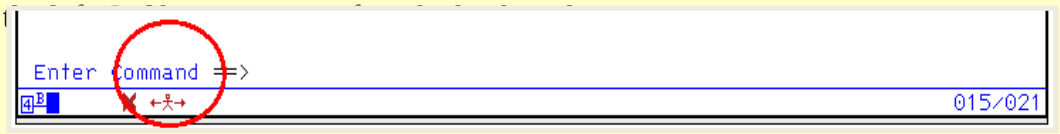
N.B., The 3270-terminal sessions and OMVS screen shots in the remainder of this exercise are shown in reverse video simply for printing purposes

____ 2. Enter ***TSO USER1*** (see below) and presss the 3270 **Enter** key (the **right-Ctrl** key sequence): The 3270-emulator used for this workshop (IBM Personal Communication) maps the 3270 enter key to the right **Ctrl** key (see below). Any references to the *Enter* key in non-3270 windows, OMVS terminal session, etc. refers to the key labeled *Enter* on the keyboard.

Mitch Johnson (mitchj@us.ibm.com)

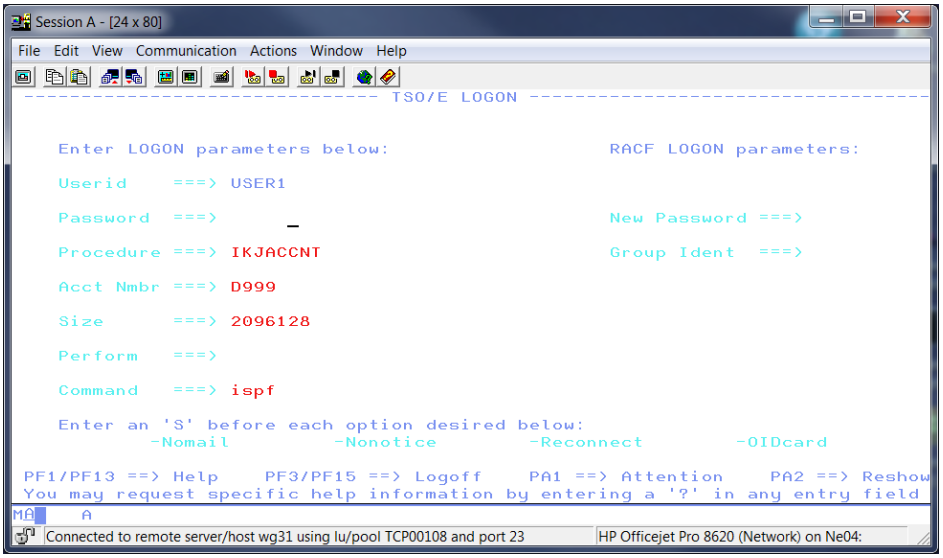
The instructions in this exercise will reference keyboard keys using a **bold** font. In the beginning explicit instructions regarding pressing keyboard keys will be provided. Eventually these explicit instructions to press the *Enter* key for example will not be included. If you are told to enter command then assume the appropriate *Enter* key should be pressed to have the command invoked or executed. Information which must be entered on a screen or panel will be in ***bold italics.*** References to text on a screen or panel will be in normal *italics*.

Tech-Tip: Different 3270-terminal emulators will display an icon similar to the *Personal Communications*®'s icon below at the bottom of the screen when the keyboard is locked. If this occurs use t



In this emulator the **Pause** key is mapped to the clear function. If your laptop has a **Pause** key use it to clear the screen. Newer laptops without a **Pause** key use the key sequence **Fn-P** to clear the screen. If none of these works, try a **Break** key or an **Alt-C** key sequence.

____ 3. On the *TSO/E LOGON* panel enter the password supplied by the instructors.



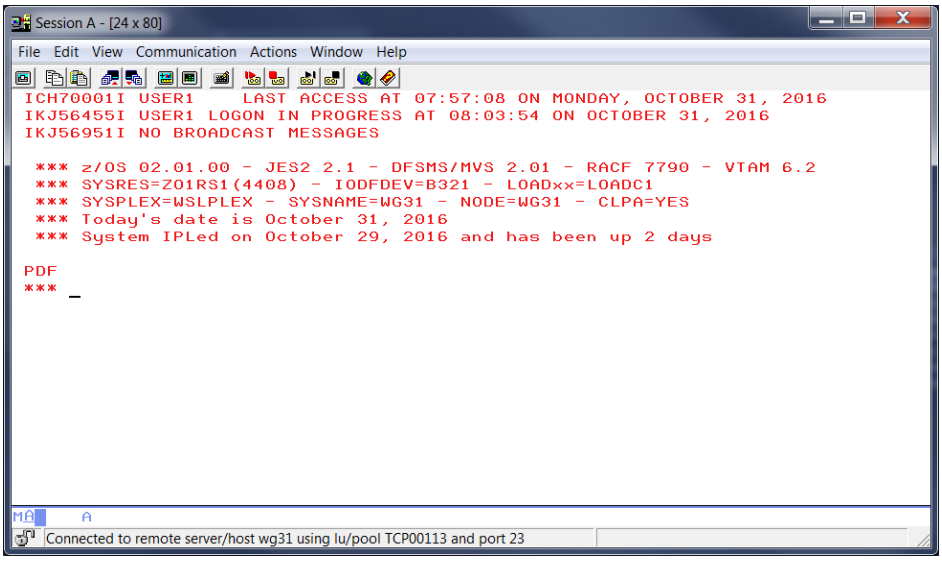
Tech-Tip: An **ISPF** or **PDF** command at the command prompt (===>) on this screen will automatically start ISPF. The copyright information displayed in Step 6 can be bypassed by entering ***ISPF NOLOGO*** or ***PDF NOLOGO*** at the command prompt.

Tech-Tip: If for some reason you are disconnected from your TSO session and cannot log in because your TSO session is still active you can enter an ***S*** beside *–Reconnect* near the bottom of the panel to reconnect to your session.

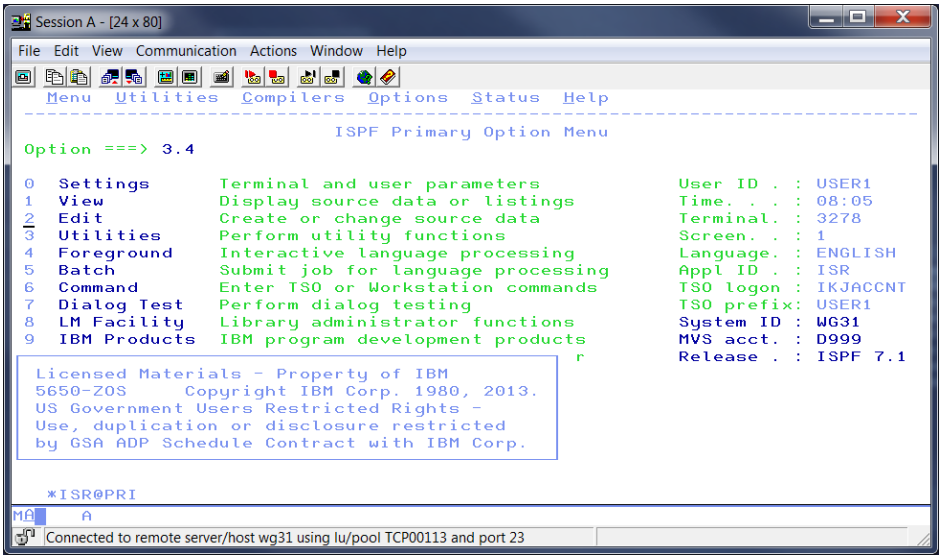
____ 4. In a TSO session whenever you see the string ***** (three asterisk)** appear (as below) as the last line in any terminal output there is more output is waiting to be displayed. Press the **Enter** key when you are

Mitch Johnson (mitchj@us.ibm.com)

ready to see this additional output. Also remember that notification messages such as jobs completing, etc. will not be displayed unless the **Enter** key is pressed



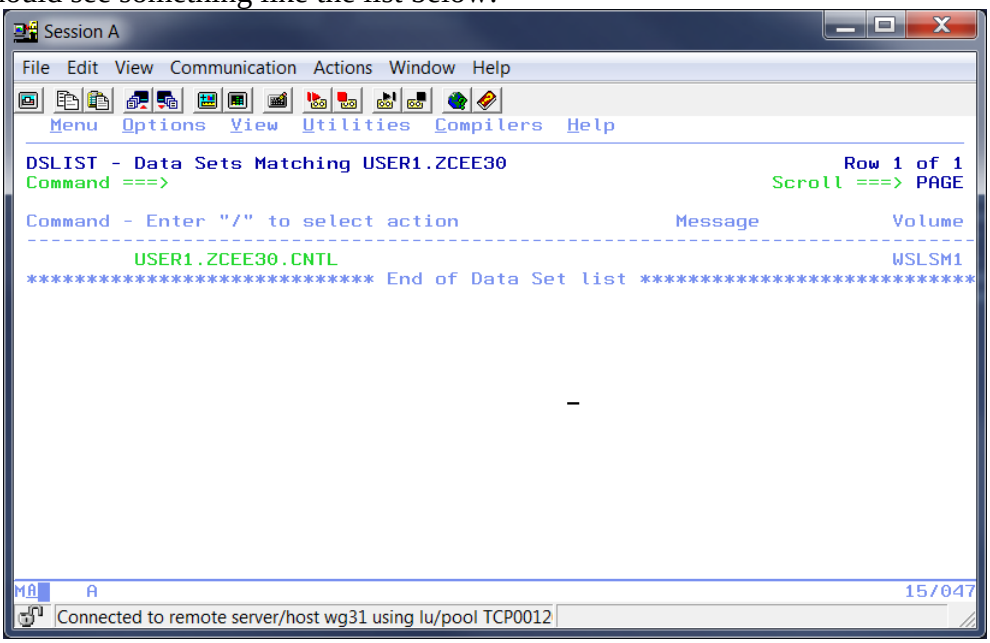
____ 5. You should now be at the main ISPF panel (see below). Press the **Enter** key to dismiss the *Copyright* statement.



____ 6. Enter ISPF command ***3.4*** in the area after the *Option*===> prompt in the upper left (see above) and press the **Enter** key to display the *Data Set List Utility* panel.

Mitch Johnson (mitchj@us.ibm.com)

____ 7. On the *Data Set List Utility* panel enter ***USER1.ZCEE30*** in the area beside *DSNAME level* and press the **Enter** key to display a list of data sets whose names begin with *USER1.ZCEE30*. You should see something like the list below:



____ 8. Enter ***E*** (for "edit") next to the data set *USER1.ZCEE30.CNTL*, and press **Enter**. You should see the a list of members in the partitioned data set.

Mitch Johnson (mitchj@us.ibm.com)

____ 9. Enter ***B*** (for "browse") next to member **ZC2RACF1** and press **Enter**: You should see a job with several RACF commands (see below):

```
ADDGROUP LIBGRP OMVS(AUTOGID) OWNER(SYS1)
ADDUSER LIBANGE DFLTGRP(LIBGRP) OMVS(AUTOUID HOME(/u/libange/) -
  PROGRAM(/bin/sh)) NAME('LIBERTY ANGEL')  NOPASSWORD NOOIDCARD
ADDUSER LIBSERV DFLTGRP(LIBGRP) OMVS(AUTOUID HOME(/u/libserv/) -
  PROGRAM(/bin/sh)) NAME('LIBERTY SERVER')
ALTUSER LIBSERV PASSWORD(LIBSERV) NOEXPIRED
RDEFINE STARTED BAQSTRT.* UACC(NONE)    -
  STDATA(USER(LIBSERV) GROUP(LIBGRP) -
  PRIVILEGED(NO) TRUSTED(NO) TRACE(YES))
RDEFINE STARTED BBGZANGL.* UACC(NONE) -
  STDATA(USER(LIBANGE) GROUP(LIBGRP) -
  PRIVILEGED(NO) TRUSTED(NO) TRACE(YES))
SETROPTS RACLIST(STARTED) REFRESH
RDEFINE SURROGAT BPX.SRV.LIBSERV
PERMIT BPX.SRV.LIBSERV CLASS(SURROGAT) ID(USER1) ACC(READ)
SETROPTS RACLIST(SURROGAT) REFRESH
```

Tech-Tip: This job creates the RACF group and IDs for the Angel process and z/OS Connect EE V3.0 server, and then it creates the two STARTED task profiles, so these IDs can be assigned to these started tasks. Finally, the RACF resources required for USER1 to act as a surrogate of user LIBSERV are defined.

____ 10. Enter command ***SUBMIT*** in the area after the command prompt (*Command ===>)* at the top of the screen, and press the **Enter** key to submit this job for execution. You should get a message indicating the job has been submitted, along with the three asterisk indicating that additional output is being held:

```
IKJ56250I JOB ZC2RACF1(JOB00060) SUBMITTED
***
```

____ 11. Press the **Enter** to display the additional output. You should then see either a message indicating the job has completed with MAXCC=0000 (which is good) or a redisplay of the ISPF browse panel. If the latter, keep pressing **Enter** until the job completes.

```
08.19.07 JOB00060 $HASP165 ZC2RACF1 ENDED AT WG31  MAXCC=0000 CN(INTERNAL)
***
```

____ 12. Press the **F3** key to return to the list of members.

Mitch Johnson (mitchj@us.ibm.com)

____ 13. Enter ***B*** next to the **ZC2RACF2** member, and press the **Enter** key. This is another set of RACF commands that define the RACF *SERVER* resources which allow the use of various z/OS authorized services. *WP102604 Getting Started Guide* describes these commands in more detail. ***Submit*** this job for execution (see Step 10 above). Allow this job to complete before continuing.

```
RDEFINE SERVER BBG.ANGEL UACC(NONE) OWNER(SYS1)
PERMIT  BBG.ANGEL CLASS(SERVER) ACCESS(READ) ID(LIBSERV)
RDEFINE SERVER BBG.AUTHMOD.BBGZSAFM UACC(NONE) OWNER(SYS1)
PERMIT  BBG.AUTHMOD.BBGZSAFM -
      CLASS(SERVER) ACCESS(READ) ID(LIBSERV)
RDEFINE SERVER BBG.AUTHMOD.BBGZSAFM.SAFCRED UACC(NONE)
PERMIT  BBG.AUTHMOD.BBGZSAFM.SAFCRED -
      CLASS(SERVER) ACCESS(READ) ID(LIBSERV)
RDEFINE SERVER BBG.AUTHMOD.BBGZSAFM.ZOSWLM UACC(NONE)
PERMIT  BBG.AUTHMOD.BBGZSAFM.ZOSWLM  -
      CLASS(SERVER) ACCESS(READ) ID(LIBSERV)
RDEFINE SERVER BBG.AUTHMOD.BBGZSAFM.TXRRS UACC(NONE)
PERMIT  BBG.AUTHMOD.BBGZSAFM.TXRRS   -
      CLASS(SERVER) ACCESS(READ) ID(LIBSERV)
RDEFINE SERVER BBG.AUTHMOD.BBGZSAFM.ZOSDUMP UACC(NONE)
PERMIT  BBG.AUTHMOD.BBGZSAFM.ZOSDUMP  -
      CLASS(SERVER) ACCESS(READ) ID(LIBSERV)
RDEFINE SERVER BBG.SECPFX.BBGZDFLT UACC(NONE)
PERMIT  BBG.SECPFX.BBGZDFLT  -
      CLASS(SERVER) ACCESS(READ) ID(LIBSERV)
RDEFINE SERVER BBG.AUTHMOD.BBGZSAFM.WOLA UACC(NONE) OWNER(SYS1)
PERMIT  BBG.AUTHMOD.BBGZSAFM.WOLA -
      CLASS(SERVER) ACCESS(READ) ID(LIBSERV)
RDEFINE SERVER BBG.AUTHMOD.BBGZSAFM.LOCALCOM UACC(NONE) OWNER(SYS1)
PERMIT  BBG.AUTHMOD.BBGZSAFM.LOCALCOM -
      CLASS(SERVER) ACCESS(READ) ID(LIBSERV)
RDEFINE SERVER BBG.AUTHMOD.BBGZSCFM UACC(NONE) OWNER(SYS1)
PERMIT  BBG.AUTHMOD.BBGZSCFM -
      CLASS(SERVER) ACCESS(READ) ID(LIBSERV)
RDEFINE SERVER BBG.AUTHMOD.BBGZSCFM.WOLA UACC(NONE) OWNER(SYS1)
PERMIT  BBG.AUTHMOD.BBGZSCFM.WOLA -
      CLASS(SERVER) ACCESS(READ) ID(LIBSERV)
PERMIT  BBG.AUTHMOD.BBGZSCFM.WOLA -
      CLASS(SERVER) ACCESS(READ) ID(LIBSERV)
RDEFINE SERVER BBG.AUTHMOD.BBGZSAFM.PRODMGR UACC(NONE) OWNER(SYS1)
PERMIT  BBG.AUTHMOD.BBGZSAFM.PRODMGR -
      CLASS(SERVER) ACCESS(READ) ID(LIBSERV)
RDEFINE SERVER BBG.AUTHMOD.BBGZSAFM.ZOSAIO UACC(NONE) OWNER(SYS1)
PERMIT  BBG.AUTHMOD.BBGZSAFM.ZOSAIO -
      CLASS(SERVER) ACCESS(READ) ID(LIBSERV)
SETROPTS RACLIST(SERVER) REFRESH
```

____ 14. Press **F3** to exit the browse session.

Summary

You just created a set of essential SAF profiles for z/OS Connect EE V3.0 to use. These are detailed in the *WP102604 Getting Started Guide*. The process was streamlined for this lab by coding them in a job so submitting this job would create what was needed.

Mitch Johnson (mitchj@us.ibm.com)

Create a z/OS Connect EE 3.0 server

z/OS Connect EE V3.0 was installed using SMP/E ahead of the workshop and the *zconsetup* command has been invoked to create a symbolic link to the */var/zosconnect/v3r0/extensions* directory. This directory contains property files which provide information required for locating z/OS Connect EE features and executables by Liberty at runtime.

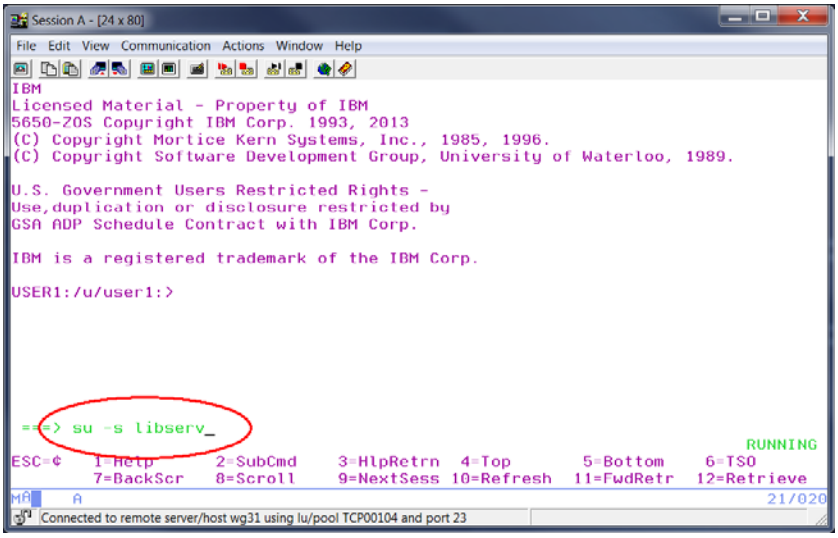
Now it's time to create a z/OS Connect server. This is done by executing a relatively simple shell script (*zosconnect*) provided with z/OS Connect EE. Remember that a z/OS Connect server runs in a Liberty runtime so occasionally there will be references to Liberty.

Tech-Tip: The OMVS directories and files created by this script must be owned by the same RACF identity associated with the z/OS Connect started task (e.g. BAQSTRT). This is problematic since this RACF identity is restricted and cannot be used to logon to TSO or submit jobs to create these directories and files.

There are various ways to ensure this ownership is set correctly. One is to create the directories and files invoking the OMVS commands using your regular RACF identity. Then use the Unix System Services (USS) command *chown* (change owner) to change the ownership of the directories and files to the RACF identity of the started task. Another way is to use Telnet clients like PuTTY or TeraTerm to access OMVS using the restricted RACF identity and then invoking OMVS commands with the restricted identity.

This exercise uses a mixture of all these techniques. In a z/OS OMVS shell the USS command *su* (switch user) is used to switch the current OMVS RACF identity to the RACF identity of the started task before invoking any commands or it uses the *chown* command to change ownership when executing commands in a MVS batch job. PuTTY is used start an OMVS shell with the restricted identity to update the z/OS Connect server configuration file with USS *cp* (copy) commands. All of these techniques are mostly interchangeable and are shown to demonstrate the options available.

___1. In the existing 3270-terminal session enter TSO command ***OMVS*** (e.g. ***TSO OMVS***) at the command prompt and press **Enter** to start an ISPF OMVS shell session. You should see the screen below



Mitch Johnson (mitchj@us.ibm.com)

___2. At the prompt, enter OMVS command ***su –s libserv*** (see above) and press the 3270-terminal session's **Enter** key. Note that *libserv* is the RACF identity created and associated with the *BAQSTRT* started task when you ran the **ZC2RACF1** job in the previous section.

> **Tech-Tip:** *LIBSERV* is the RACF identity created by the *ADDUSER* command executed when job ZC2RACF1 job was executed in the previous section. The *ALTUSER* command in the same job set the password for *LIBSERV* to *LIBSERV*.
>
> The ability to switch to this RACF identity without requiring a password in the *su* command was granted by defining the *BPX.SRV.LIBSERV* surrogate resource to RACF and then permitting user USER1 read access to this surrogate resource in *ZC2RACF1*.
>
> It is very important that this command or alternatives be considered when creating a z/OS Connect EE (zCEE) server, otherwise the server may not start or perform correctly. The RACF identity of the zCEE started task, e.g. *LIBSERV* and must be either be the owner or explicitly given read/write access to the directories and files in *the /var/zosconnect/server/myServer* directory structure

N.B. Instructions to press the **Enter** key will be omitted in subsequent steps; pressing **Enter** should be assumed whenever a command, script, etc. is to be executed.

___3. When you are logged on, enter the ***id*** command to confirm the user ID (*uid*) and group ID (*gid*) values are for LIBSERV and LIBGRP.

```
USER1:/u/user1:> su -s libserv
$ id
uid=200019(LIBSERV) gid=200017(LIBGRP)
$
```

___4. The z/OS Connect *zosconnect* shell script needs to be able to locate the Java executables. Use following command in the OMVS session to *export* the environment variable *JAVA_HOME*. This environment variable identifies the directory containing the location of the Java binaries.

```
export JAVA_HOME=/usr/lpp/java/J8.0_64
```

___5. Change to the directory containing the z/OS Connection script *zosconnect* using an OMVS *cd* command:

```
cd  /usr/lpp/IBM/zosconnect/v3r0/bin
```

> **Tech-Tip:** If you every have any questions about which directory your session is current using use the ***pwd*** (print working directory) to display the current directory path.

___6. Export environment variable (*WLP_USER_DIR*) to identify the directory where z/OS Connect EE V3.0 server configurations will reside.

```
export WLP_USER_DIR=/var/zosconnect
```

> **Tech-Tip:** The *WLP_USER_DIR* will be exported in the startup JCL. The value used in the JCL must be the same as the value used when the server was created.

___7. Invoke the zosconnect script to create the z/OS Connect server named ***myServer:***

```
./zosconnect  create  myServer  --template=zosconnect:default
```

You should see the following in response.

```
Server myServer created.
```

Tech-Tip: Other templates which can be specified with the *zosconnect create* command are *zosconnect:apiRequester*, *zosconnect:sampleCicsIpicCatalogManager*, *zosconnect:sampleWolaCatalogManager*, *zosconnect:sampleImsPhonebook* and *imsmobile:imsdefault.*

Tech-Tip: Member *ZCEESRVR* in *USER1.ZCEE30.CNTL* is an example of creating a server using these same commands in a MVS batch job. Again, this job runs under the authority of RACF identity LIBSERV by using RACF SURROGAT resources. If a surrogate is not used, then the ownership of the directories and files created by running this job must be changed so they are owned by the user associated with the server's started task.

___8. Enter the following OMVS command to display the contents of the */var/zosconnect/servers* subdirectory:

```
ls  -al  /var/zosconnect/servers
```

You should see:

```
$ ls -al /var/zosconnect/servers
total 96
drwxr-x--T   6 LIBSERV  SYS1        8192 Oct 31 11:57 .
drwxrwxrwx   4 BAGWELL  SYS1        8192 Oct 29 15:17 ..
drwxr-x--T   3 LIBSERV  SYS1        8192 Oct 29 15:17 .classCache
drwxr-x---   2 LIBSERV  SYS1        8192 Oct 31 11:57 .logs
drwxr-x---   3 LIBSERV  SYS1        8192 Oct 31 11:57 myServer
```

The *myServer* directory was created by *zosconnect* command in the previous step.

___9. Now display the contents of the */var/zosconnect/servers/myServer* directory by entering OMVS command:

```
ls  -al  /var/zosconnect/servers/myServer
```

You should see:

```
$ ls -al /var/zosconnect/servers/myServer
total 66
drwxr-x---   3 LIBSERV  SYS1        8192 Oct 31 11:57 .
drwxr-x--T   6 LIBSERV  SYS1        8192 Oct 31 11:57 ..
drwxr-x---   4 LIBSERV  SYS1        8192 Aug  1 16:23 resources
-rw-r-----   1 LIBSERV  SYS1          25 Oct 31 11:57 server.env
-rw-r-----   1 LIBSERV  SYS1         442 Oct 31 11:57 server.xml
drwxr-x---   3 LIBSERV  SYS1        8192 Oct 31 11:57 workarea
```

You just created a server, but right now it's just a skeleton without any service definitions or deployed APIs. Next you will customize the JCL start procedures for the Angel process and the server.

Mitch Johnson (mitchj@us.ibm.com)

___10. Terminate the OMVS session by entering the ***exit*** command twice to redisplay an ISPF panel. The first exit terminates the *libserv* session and the second exit terminates the *user1* session.

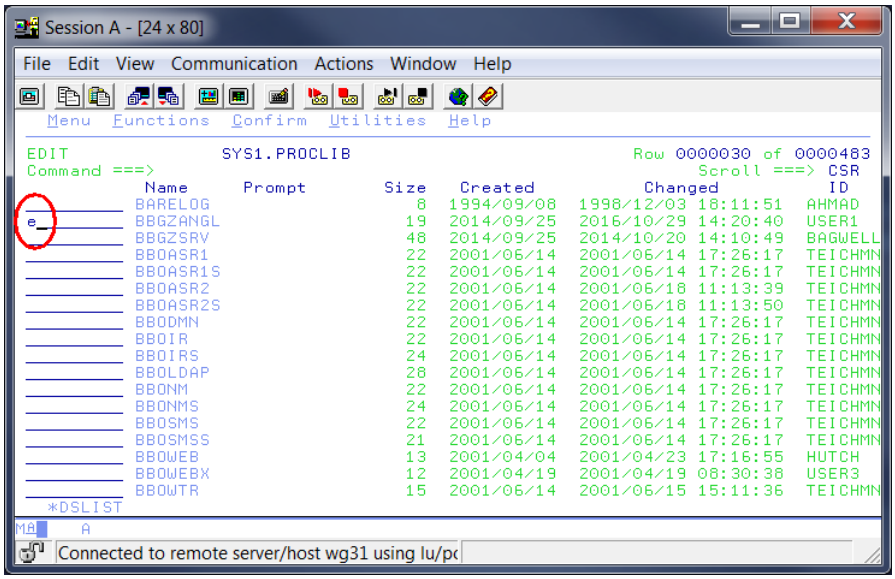
Customize the JCL start procedures and start the server

___1. In your 3270-terminal session enter ISPF command **=3.4** in the command prompt area and press **Enter**.

___2. Enter ***SYS1.PROCLIB*** in the area beside *Dsname Level* and press **Enter.**

___3. Type an **E** (edit) next to the *SYS1.PROCLIB* data set and press **Enter**.

___4. You will now see a long list of members in the *SYS1.PROCLIB* data set. Enter command ***L BBG*** after the command prompt (Command ===>) and press **Enter**. Locate the **BBGZANGL** member and place an **E** (edit) next to **BBGZANGL** to open this member in edit mode so changes can be made to the member.



This JCL procedure was copied this member to *SYS1.PROCLIB* prior to the workshop. This JCL is supplied in the z/OS Connect EE V3.0 SBAQSAMP target data set.

N.B. Instructions to press the **Enter** key should be assumed in subsequent steps as appropriate in order to execute commands, etc.

___5. Update the BBGZANGL JCL procedure to change the value of the *SET ROOT* parameter on line 3 to the location of the Liberty code shipped with z/OS Connect, e.g. */usr/lpp/IBM/zosconnect/v3r0/wlp*.

```
//  SET ROOT='/usr/lpp/IBM/zosconnect/v3r0/wlp'
```

```
****** ******************************** Top of Data ********************************
000001 //BBGZANGL PROC PARMS='',COLD=N, NAME=''
000002 //*--------------------------------------------------------------------
000003 //  SET ROOT='/usr/lpp/IBM/zosconnect/v3r0/wlp'
000004 //*--------------------------------------------------------------------
```

___6. Press **F3** to save the member and exit the edit session.

___7. Scroll up (**F7**) and locate member **BAQSTRT**. Place and **E** next to the member in order to open the member so changes can be made to the member.
This is the JCL procedure used to start the z/OS Connect EE V3.0 server. This JCL was copied to *SYS1.PROCLIB* prior to the workshop as well.

___8. You are going to make three changes to this JCL procedure The first is to replace the first line so the PARMS= specifies your server's name[1].

___9. Clear line 1 by placing your cursor at the start of the line (on the first slash in column 1):

```
000001 //BAQSTRT  PROC PARMS='defaultServer'
000002 //*
000003 //* Licensed Materials - Property of IBM
```

___10. Then press the **End** key. That should clear the line of all characters.

___11. Copy/Paste the JCL statement below on line 1 by using **Ctrl-C** of the text from the copy/paste file and then selecting *Edit* → *Paste* or **Ctrl-V** in the 3270-terminal session.

//BAQSTRT PROC PARMS='myServer'

The result should be:

```
000001 //BAQSTRT  PROC PARMS='myServer'
000002 //*
000003 //* Licensed Materials - Property of IBM
```

Important: In the above instructions the *Edit* → *Paste* translates to click the *Edit* tool in the tool bar of the 3270-terminal session and then select the *Paste* option. The → arrow will be used in later steps for selecting other tools in a tool bar and selecting an option in the 3270-terminal session and other windows.

You could have just overtyped *defaultServe*r with *myServer*. We chose to go with copy-and-paste to eliminate any chance of a case mis-match or typo. The field was cleared first because the original text in the line was longer than its replacement.

___12. The next two lines to be updated are the *// SET ZCONHOME=* (line 38) and the *JAVA_HOME=* (line 47) environment variables. Do the following:

- Scroll down (**F8**) until you see the *SET ZCONHOME=* line:
- Place your cursor at the start of the line (the first slash):
- From the copy-and-paste file, copy the line:

// SET ZCONHOME='/usr/lpp/IBM/zosconnect/v3r0'

```
000037 //*
000038 // SET ZCONHOME='/usr/lpp/IBM/zosconnect/v3r0'
000039 //*
```

1 You could start the server with /S BAQSTRT,PARMS='myServer' and avoid updating this line in the JCL. We chose to hard-code the server name for this workshop so the start is a simple /S BAQSTRT.

Mitch Johnson (mitchj@us.ibm.com)

- Scroll down (**F8**) and locate the *JAVA_HOME*= line. It should be line 47:
- Place your cursor at the start of that line and press the **End** key to clear the line.
- From the copy-and-paste file, copy the line:

  JAVA_HOME=/usr/lpp/java/J8.0_64

- Place your cursor at the start of that line and press the **End** key to clear the line.
- Locate the *WLP_USER_DIR*= line. It should be line 48:
- Place your cursor at the start of that line and press the **End** key to clear the line.
- From the copy-and-paste file, copy the line:

  WLP_USER_DIR=/var/zosconnect

- In your 3270-terminal session, select *Edit* → *Paste* (you may need to remove a trailing > character). The result should be:

```
000045 //STDENV   DD   *
000046 _BPX_SHAREAS=YES
000047 JAVA_HOME=/usr/lpp/java/J8.0_64
000048 WLP_USER_DIR=/var/zosconnect
000049 #JVM_OPTIONS=<Optional JVM parameters>
```

Now you're ready to start the *Angel* process and the server.

Tech Tip: An additional DD statement can be added to the JCL which will provide useful information regarding the status of the z/OS Connect. Normally detailed Liberty messages are written to a OMVS file in ASCII but if you add a DD statement for DD name MSGLOG and specify SYSOUT=* a subset of these messages will be included in the SPOOL of the task.

___13. Go to SDSF in your 3270-terminal session by entering ISPF command ***=sdsf.da*** in the command field and pressing **Enter.** The *da* stands for *display active* and it is the z/OS facility for show running tasks and completed jobs.

___14. Next, enter ***PRE BBG**** in the command field and press **Enter.** This sets a "prefix" so only tasks starting with string *BBG* are shown. The Angel task starts with *BBG*. After entering that command, you should see the new prefix is in effect:

___15. Enter the MVS command ***/S BBGZANGL*** (note the leading slash) after the command prompt.

___16. Keep pressing the **Enter** key until you should then see the Angel process is running.

```
  Display  Filter  View  Print  Options  Search  Help
-------------------------------------------------------------------------------
SDSF DA WG31     WG31     PAG  0  CPU   0             LINE 1-1 (1)
COMMAND INPUT ===>                                      SCROLL ===> CSR
NP   JOBNAME  StepName ProcStep JobID    Owner    C Pos DP Real Paging    SIO
     BBGZANGL BBGZANGL STEP1    STC00045 LIBANGE    NS  FE  352   0.00   0.00
```


Mitch Johnson (mitchj@us.ibm.com)

___17. You should see *LIBANGE* is the ID that owns the started task. That is a result of the RACF generated by the *ZC2RACF1* job you ran earlier. The STARTED profile matched to the *BBGZANGL* start procedure and assigned the ID *LIBANGE* to this task.

```
  Display  Filter  View  Print  Options  Search  Help
-------------------------------------------------------------------------------
SDSF DA WG31     WG31     PAG  0  CPU   0              LINE 1-1 (1)
COMMAND INPUT ===>                                     SCROLL ===> CSR
PREFIX=BBG*  DEST=(ALL)  OWNER=*  SYSNAME=
NP   JOBNAME    CPU-Time ProcStep SR DP Pos C Owner    Status SysName  SPag SCP
     BBGZANGL       0.00 STEP1       FE NS    LIBANGE         WG31        0
```

___18. Change the prefix again by entering ***PRE BAQ**** at the command line. The *BBGZANGL* task no longer shows (it's still running, but the prefix no longer matches).

___19. Enter the command MVS command ***/S BAQSTRT*** (note the leading slash) after the command prompt press **Enter** until you see an active BAQSTRT task.

```
  Display  Filter  View  Print  Options  Search  Help
-------------------------------------------------------------------------------
SDSF DA WG31     WG31     PAG  2  CPU   8              LINE 1-1 (1)
COMMAND INPUT ===>                                     SCROLL ===> CSR  =
NP   JOBNAME  StepName ProcStep JobID    Owner    C Pos DP Real Paging    SIO
     BAOSTRT  BAOSTRT  ZCON     STC00048 LIBSERV    IN  F8  10T   0.00 21640
```

You should see that *LIBSERV* is the ID that owns the started task. That is a result of the RACF generated by the *ZC2RACF1* job you ran earlier. The STARTED profile matched to the BAQSTRT start procedure and assigned the ID *LIBSERV* to it.).

Note: The z/OS Connect EE server is running with *LIBSERV*'s RACF authority. This is why it was important to run the ***zoconnect create*** command as LIBSERV and why it is important in subsequence steps to use the *LIBSERV* identity when deploying Service ARhive (SAR) files. Also note that in later steps when files are included into the server.xml configuration file, the files that are included are owned by *LIBSERV*. All of this is done to avoid OMVS permission issues.

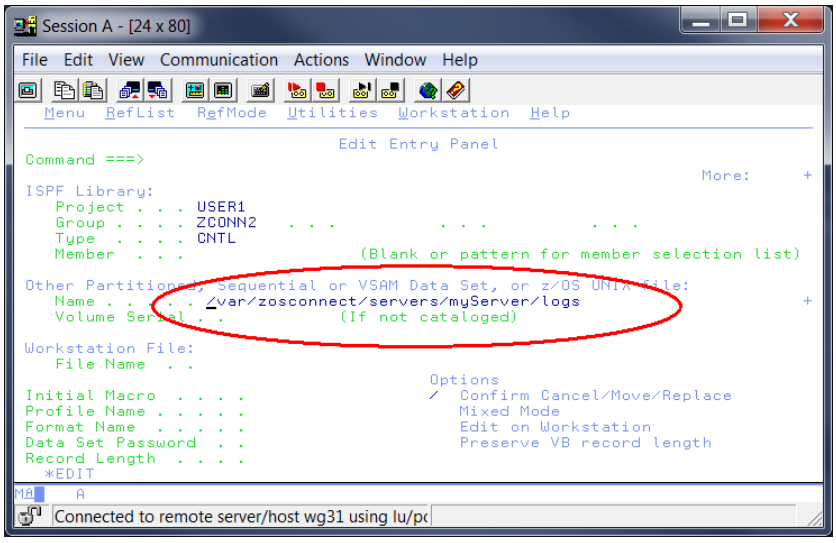
Both the Angel process and the z/OS Connect EE V3.0 server are started. Let's go look at the *messages.log* file for the server.

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 enter 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 and you may enter the command at the prompt by using the / (slash) prefix or using the *SDSF – System Command Extension* panel.

___20. Go to the ISPF Edit Entry Panel (option *2)* by entering ISPF command ***=2*** on the command line and pressing **Enter**.

Tech Tip: Most of the subsequent steps in this section can be performed using the IBM z/OS Explorer. If you are interested in doing these steps using this Eclipse tool contact the instructor.

___21. Enter ***/var/zosconnect/servers/myServer/logs*** into the area beside *Name* under *Other Partitioned, Sequential or VSAM Data Set, or z/OS UNIX file:* and press **Enter**.



___22. This will display the contents of the OMVS directory where the z/OS Connection messages are written to file *messages.log*.

```
------------------------------------------------------------------------------
                          z/OS UNIX Directory List            Row 1 to 4 of 4
Command ===>                                                 Scroll ===> PAGE
Time zone EST5EDT is used to calculate the displayed date and time values.
Pathname . : /SYSTEM/var/zosconnect/servers/myServer/logs
EUID . . . : 8470391
Command  Filename                      Message          Type Permission
------------------------------------------------------------------------------
         .                                              Dir  rwxrwxrwx
         ..                                             Dir  rwxr-x---
         messages.log                                   File rw-rw-rw-
         state                                          Dir  rwxrwxrwx
****************************** Bottom of data ******************************
```

___23. Messages are written in ASCII so to view them in a 3270-terminal session use the ***VA*** (View ASCII) line command. Enter line command ***VA*** beside *messages.log* and press **Enter** twice.

___24. You should see the *View* panel open and the *messages.log* file contents displayed in EBCDIC:

```
  File  Edit  Edit_Settings  Menu  Utilities  Compilers  Test  Help
sssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssss
VIEW       /SYSTEM/var/zosconnect/servers/myServer/logs/me Columns 00001 00072
Command ===>                                                  Scroll ===> PAGE
****** ***************************** Top of Data ******************************
000001 ********************************************************************************
000002 product = WAS FOR Z/OS 18.0.0.3, z/OS Connect 03.00.15 (wlp-1.0.22.cl180
000003 wlp.install.dir = /shared/IBM/zosconnect/v3r0/wlp/
000004 server.config.dir = /var/zosconnect/servers/myServer/
000005 java.home = /shared/java/J8.0_64
000006 java.version = 1.8.0
000007 java.runtime = Java(TM) SE Runtime Environment (pmz6480sr3fp20-20161019_
000008 os = z/OS (02.01.00; s390x) (en_US)
000009 process = 16777379@WG31
000010 ********************************************************************************
000011 [11/29/18 19:49:24:632 GMT] 0000000d com.ibm.ws.kernel.launch.internal.F
000012 [11/29/18 19:49:26:483 GMT] 0000001b com.ibm.ws.config.xml.internal.XMLC
000013 [11/29/18 19:49:26:527 GMT] 0000001b com.ibm.ws.config.xml.internal.XMLC
000014 [11/29/18 19:49:26:529 GMT] 0000001b com.ibm.ws.config.xml.internal.XMLC
000015 [11/29/18 19:49:26:530 GMT] 0000001b com.ibm.ws.config.xml.internal.XMLC
000016 [11/29/18 19:49:26:532 GMT] 0000001b com.ibm.ws.config.xml.internal.XMLC
000017 [11/29/18 19:49:26:777 GMT] 00000024 com.ibm.ws.logging.internal.TraceSp
000018 [11/29/18 19:49:27:228 GMT] 0000001b com.ibm.ws.zos.core.internal.Native
000019 [11/29/18 19:49:27:230 GMT] 0000001b com.ibm.ws.zos.core.internal.Native
000020 [11/29/18 19:49:27:244 GMT] 0000001b com.ibm.ws.zos.core.internal.Native
```

___25. You need to scroll to the right to see more message details. Enter ***100*** after the command prompt (===>) and press the **F11** key.

```
  File  Edit  Edit_Settings  Menu  Utilities  Compilers  Test  Help
--------------------------------------------------------------------------------
VIEW       /SYSTEM/var/zosconnect/servers/myServer/logs/me Columns 00101 00172
Command ===>                                                  Scroll ===> CSR
****** ***************************** Top of Data ******************************

0.22.cl180320180905-2337)









WKE0001I: The server myServer has been launched.
WKB0103I: Authorized service group KERNEL is available.
WKB0103I: Authorized service group LOCALCOM is available.
WKB0103I: Authorized service group PRODMGR is available.
WKB0103I: Authorized service group SAFCRED is available.
WKB0103I: Authorized service group TXRRS is available.
WKB0103I: Authorized service group WOLA is available.
WKB0103I: Authorized service group ZOSAIO is available.
WKB0103I: Authorized service group ZOSDUMP is available.    .
```

Note that the **F8** key can be used to scroll forward and the **F7** key can be used to scroll backward. **F10** can be used to scroll leftward and F**11** key can be used to scroll rightward. A numeric value on the command prompt will be used to determine the number of lines to scroll forward or backward or the number of columns for scrolling leftward or rightward.

Mitch Johnson (mitchj@us.ibm.com)

___26. Look at the messages and note the following:

```
000012 WKB0103I: Authorized service group KERNEL is available.
000013 WKB0103I: Authorized service group LOCALCOM is available.
000014 WKB0103I: Authorized service group PRODMGR is available.
000015 WKB0103I: Authorized service group SAFCRED is available.
000016 WKB0103I: Authorized service group TXRRS is available.
000017 WKB0103I: Authorized service group WOLA is available.
000018 WKB0103I: Authorized service group ZOSAIO is available.
000019 WKB0103I: Authorized service group ZOSDUMP is available.
000020 WKB0103I: Authorized service group ZOSWLM is available.
000021 WKB0103I: Authorized service group CLIENT.WOLA is available.
000022 WKB0108I: IBM Corp product z/OS Connect version 03.00 successfully regis
000023 WKB0112I: The number of successfully registered products with z/OS is 1.
000024 WKE0002I: The kernel started after 3.05 seconds
000025 WKF0007I: Feature update started.
000026 WKS0007I: The security service is starting...
000027 NA1001I: WebSphere Dynamic Cache instance named baseCache initialized su
000028 NA1071I: The cache provider default is being used.
000029 NA1056I: Dynamic Cache (object cache) initialized successfully.
000030 WKO0229I: Native Asynchronous I/O support for z/OS has been activated.
000031 WKS4103I: Creating the LTPA keys. This may take a few seconds.
000032 WKS1123I: The collective authentication plugin with class name NullColle
000033 QR0000I: z/OS Connect Enterprise Edition version 3.0.15. (20181120-1404)
000034 WKO0219I: TCP Channel defaultHttpEndpoint has been started and is now li
000035 WKS4104A: LTPA keys created in 1.513 seconds. LTPA key file: /var/zoscon
000036 WKS4105I: LTPA configuration is ready after 1.527 seconds.
000037 WKF0015I: The server has the following interim fixes active in the runti
000038 WKF0012I: The server installed the following features: [servlet-3.1, ssl
000039 WKF0008I: Feature update completed in 5.539 seconds.
000040 WKF0011I: The server myServer is ready to run a smarter planet.
000041 VE0169I: Loading Web Module: z/OS Connect.
000042 VE0250I: Web Module z/OS Connect has been bound to default_host.
000043 WKT0016I: Web application available (default_host): http://wg31.washingt
000044 SN8501I: The session manager did not find a persistent storage location;
000045 SN0176I: A new session context will be created for application key defau
000046 SN0172I: The session manager is using the Java default SecureRandom impl
000047 NA1056I: Dynamic Cache (object cache) initialized successfully.
000048 VE0242I: [com.ibm.zosconnect] [/] [com.ibm.zosconnect.internal.web.Servi
000049 WKS9122I:  For URL /* in application com.ibm.zosconnect, the following H
```

1. The "Authorized service …. is available" messages indicate the SERVER profiles (**ZC2RACF2** job) are in effect for this server and the ID under which it runs. Of particular interest is SAFCRED (for later when we use SAF for security).
2. Version 3.0.15.0 (20181120-1404) indicates the maintenance made available in November 20th of 2018 is in effect. The values you see may be different.
3. The server is started and ready to run.

___25. Use the **F3** key to go back to the *Edit Entry Panel* and use the space bar or **Backspace** key to remove the */logs* subdirectory from the directory name field. Press **Enter** to display the contents of */var/zosconnect/servers/myServer*. You should see:

```
   Menu  Utilities  View  Options  Help
------------------------------------------------------------------------------
                           z/OS UNIX Directory List            Row 1 to 8 of 8
 Command ===>                                                 Scroll ===> PAGE
 Time zone EST5EDT is used to calculate the displayed date and time values.
 Pathname . : /SYSTEM/var/zosconnect/servers/myServer
 EUID . . . : 8470391
 Command  Filename                      Message          Type Permission
 ------------------------------------------------------------------------------
          .                                              Dir  rwxr-x---
          ..                                             Dir  rwxr-x--T
          logs                                           Dir  rwxrwxrwx
          resources                                      Dir  rwxrwxrwx
          server.env                                     File rw-r-----
          server.xml                                     File rw-r-----
          workarea                                       Dir  rwxr-x---
 ****************************** Bottom of data ********************************
```

___26. Use the ***EA*** (Edit ASCII) line command to open the *server.xml* file in EBCDIC mode. You should see:

```
000001 <?xml version="1.0" encoding="UTF-8"?>
000002 <server description="new server">
000003
000004     <!-- Enable features -->
000005     <featureManager>
000006         <feature>zosconnect:zosConnect-2.0</feature>
000007         <feature>zosconnect:zosConnectCommands-1.0</feature>
000008     </featureManager>
000009
000010     <!-- To access this server from a remote client add a host attribute
000011     <httpEndpoint id="defaultHttpEndpoint"
000012                   host="*"
000013                   httpPort="9080"
000014                   httpsPort="9443" />
000015
000016     <!-- add cors to allow cross origin access, e.g. when using swagger
000017     <cors id="defaultCORSConfig"
000018           domain="/"
000019           allowedOrigins="*"
000020           allowedMethods="GET, POST, PUT, DELETE, OPTIONS"
000021           allowedHeaders="Origin, Content-Type, Authorization"
000022           allowCredentials="true"
000023           maxAge="3600"/>
000024
000025  <!-- NOTE: Disabling automatic polling for changes to configuration fil
000026       deployed services and APIs is a prudent option for z/OS Connect EE
000027    Polling might be convenient for iterative development and test
000028    systems, but not for production.
000029
000030       Configuration elements that can drive significant polling activity
000031    default are specified below to explicitly disable automatic polling.
000032    Further element types to consider for polling interval include
000033    zosconnect_zosConnectDataXform (default 2 seconds) and
000034    keyStore (default 500 milliseconds).
```


Mitch Johnson (mitchj@us.ibm.com)

```
000035
000036    Consider setting the updateTrigger attribute to "polled" if changes
000037    to associated resources need to be picked up automatically, and tune
000038    the polling interval accordingly. The attribute that controls polling
000039    frequency for each of these elements is included, together with its
000040    associated default value.
000041     -->
000042
000043  <!-- config requires updateTrigger="mbean" for REFRESH command support
000044  <config updateTrigger="mbean" monitorInterval="500"/>
000045
000046     <!-- zosConnect APIs -->
000047     <zosconnect_zosConnectAPIs updateTrigger="disabled" pollingRate="5s"
000048
000049     <!-- zosConnect Services -->
000050  <zosconnect_services updateTrigger="disabled" pollingRate="5s"/>
000051
000052  <!-- applicationMonitor is not applicable for zCEE servers -->
000053  <applicationMonitor updateTrigger="disabled" dropinsEnabled="false"/>
000054
000055 </server>
```

That is a *very* minimal server.xml file created from the default template. Other templates will generate server.xml files based on their parameters.
In the next step we will have you copy in a new server.xml file that will add quite a few more XML elements. The dynamic nature of Liberty will incorporate the new configuration without requiring a server restart.

. **Tech Tip:** Note that by default the *updateTrigger* attributes are disabled for APIs and services. These will be enable later in this exercise.

___27. Add the following include after the ***<server description=<new server”>*** line.

<include location="/wasetc/zc3lab/basic.xml" optional="true"/>

Mitch Johnson (mitchj@us.ibm.com)

___28. Including this file add a new feature to the server, *appSecurity-2.0* which enables application security and some of the other elements required for basic security.

```
<server description="basic security">

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

    <keyStore id="defaultKeyStore" password="Liberty"/>

    <webAppSecurity allowFailOverToBasicAuth="true" />

    <basicRegistry id="basic1" realm="zosConnect">
        <user name="Fred" password="fredpwd" />
    </basicRegistry>

    <authorization-roles id="zos.connect.access.roles">
        <security-role name="zosConnectAccess">
             <user name="Fred"/>
         </security-role>
     </authorization-roles>

</server>
```

Figure 1: Contents of basic.xml

"Basic" means a user is authentication by providing a user identity and password. In this case the user registry will be provided by the Liberty server itself. For now it's a simple way to satisfy the security requirements of z/OS Connect EE V3.0. In subsequent section of this exercise, SAF (e.g. RACF) will be implemented.

___29. Use the **F3** key to end the edit session.

___30. Adding this file using an *include* statement refreshing the z/OS Connect EE V3.0 server. Enter the following MVS command to refresh the configuration ***F BAQSTRT,ZCON,REFRESH***

Next, we're going to do a preliminary test of your z/OS Connect EE V30 server, even though you do not yet have any services or APIs defined.

. **Tech Tip:** MVS commands can be entered using SDSF. In ISPF, enter the command ***=SDSF.LOG*** and add the / (slash) prefix to the command, e.g. /F BAQSTRT,ZCON,REFRESH

___31. When the message that the server configuration has been successfully updated appear in the *messages.log* file (see below), open the Firefox browser on your desktop.

```
CWPKI0803A: SSL certificate created in 5.313 seconds. SSL key file: /var/zosconnect/servers/myServer/resources/security/key.jks
CWWKS9112A: The web application security settings have changed. The following properties were modified: allowFailOverToBasicAuth=true
CWWKS9120I: Authorization roles with id="zos.connect.access.roles" have been successfully processed
CWWKG0017I: The server configuration was successfully updated in 5.481 seconds.
CWWKO0219I: TCP Channel defaultHttpEndpoint-ssl has been started and is now listening for requests on host *  (IPv4) port 9443.
```

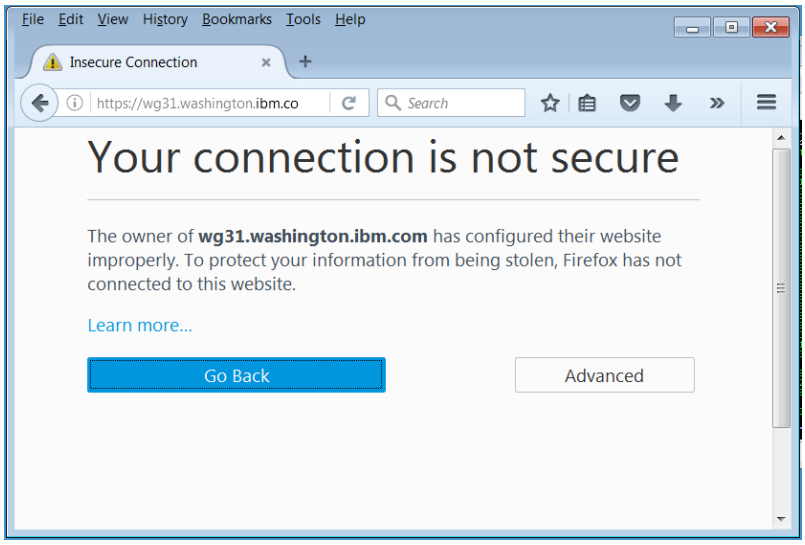

Mitch Johnson (mitchj@us.ibm.com)

Enter the following as the URL:

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

This will query the server for configured APIs. Even thought you will receive a response indicating no APIs are configured this is still a good test of connectivity to the server. But before we see the screen showing no APIs are defined we first must address an issue where the z/OS Connect EE server is using a self-signed certificate that is not recognized by the browser and we must authenticate to the z/OS Connect EE server with a user identity and password.

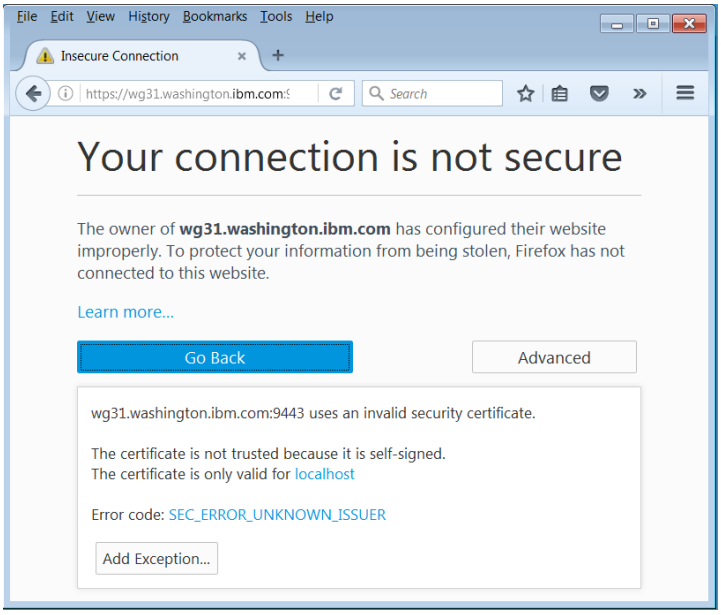
___32. Initially you will be challenged by Firefox because the digital certificate used by the Liberty z/OS server is self-signed (recall that we're using simple basic security for the time being). Click on the **Advanced** button to continue.



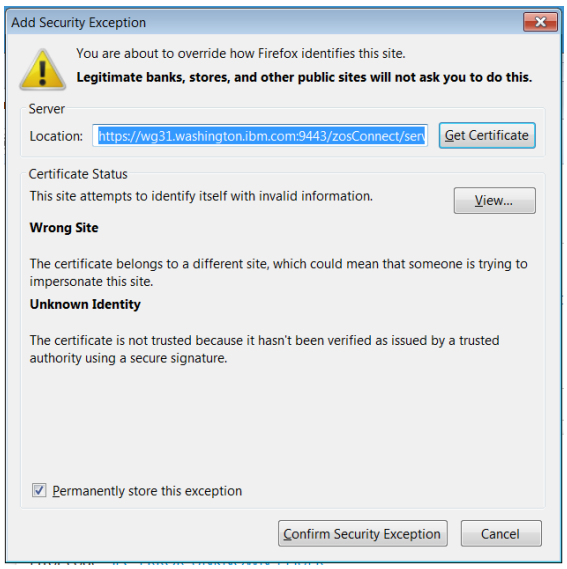
Tech Tip: It is very important to access the z/OS Connect server from a browser prior to any testing using the Swagger UI. Accessing a z/OS Connect URL from a browser starts an SSL handshake between the browser and the server. If this handshake has not performed prior to performing any test the test will fail with no message in the browser and no explanation. Ensuring this handshake has been performed is why you may be directed to access a z/OS Connect URL prior to using the Swagger UI during this exercise.

Mitch Johnson (mitchj@us.ibm.com)

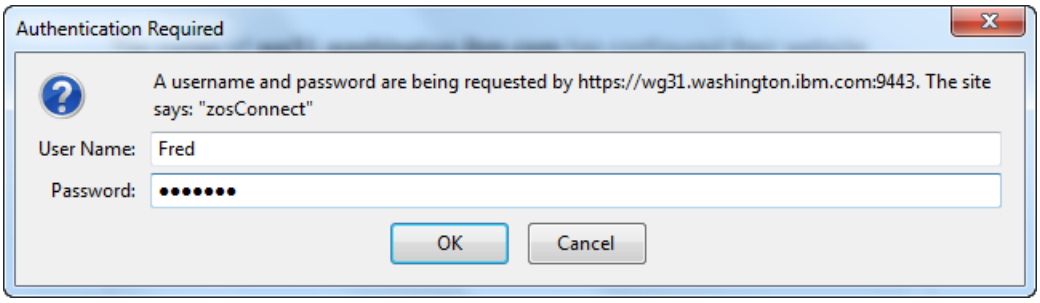
___33. Click the **Add Exception** button to continue.



___34. Then click on the **Confirm Security Exception** button".



___35. Next you will see a prompt you for a userid and password:



Enter the username ***Fred*** and password **fredpwd** (case matters) and click **OK**.

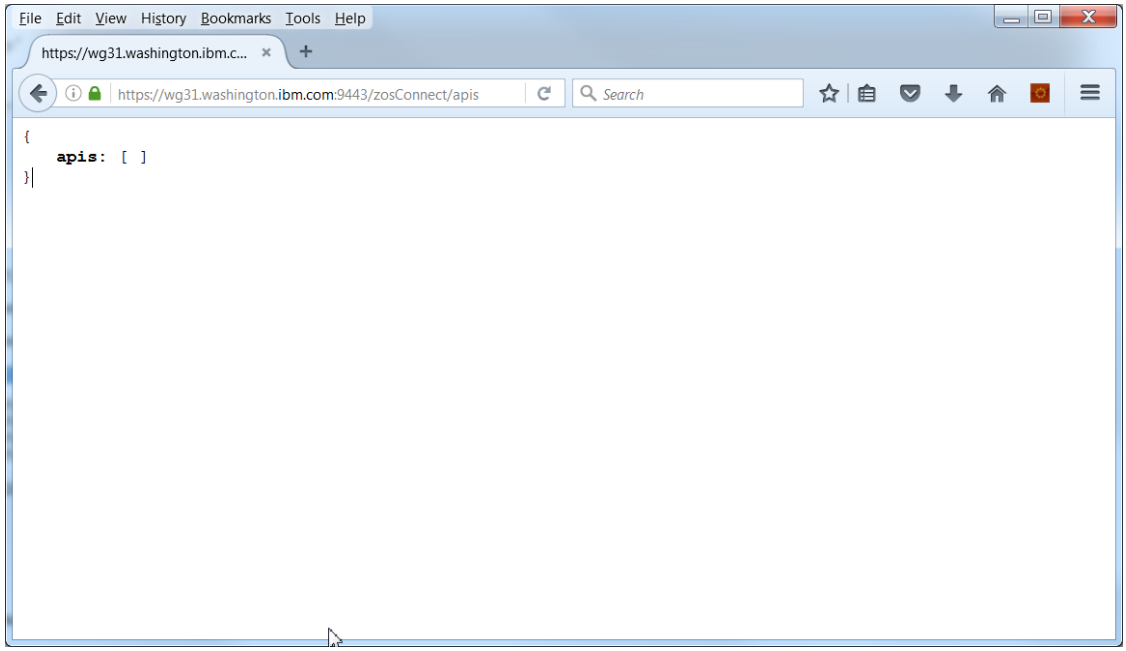
Note: The ID and password is defined in the server.xml file:

```
<basicRegistry id="basic1" realm="zosConnect">
   <user name="Fred" password="fredpwd" />
</basicRegistry>
```

Our objective for security at this point in the workshop is simplicity. That's why we're using the security elements coded in the server.xml. Later we will use RACF.

Mitch Johnson (mitchj@us.ibm.com)

___36. You should now see the following. That is the expected result since no APIs have been installed.



___37. Enter the following as the URL:

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

That will query the server for configured services. You should also get a response indicating no services are configured. This is expected.

Summary

You customized the JCL start procedures and started both the Angel process and the server. You have added basic security for z/OS Connect EE V3.0. Finally, you verified basic operations with the browser.

Mitch Johnson (mitchj@us.ibm.com)

Composing and Deploying Services and APIs

In this lab you will use the Eclipse-based z/OS Explorer tool with the z/OS Connect EE API Tookkit installed. When the SAR is completed, you will upload the SAR (Service Archive) file and then use the SAR to develop an API. Once the API is completed it will be deployed to a z/OS Connect EE server and tested.

Update the z/OS Connect EE Server Configuration

Before we begin developing the Services and APIs for the application we should add the configuration information for the application to the server.xml of the z/OS Connect EE server where the services and APIs will be running.

___1. Edit the *server.xml* in */var/zosconnect/servers/myServer* and add the following *include* after the ***<server description=<new server”>*** line.

<include location="/wasetc/zc3lab/ipic.xml" optional="true"/>

```
<?xml version="1.0" encoding="UTF-8"?>
<server description="new server">
<include location="/wasetc/zc3lab/ipic.xml" optional="true"/>
<include location="/wasetc/zc3lab/basic.xml" optional="true"/>
```

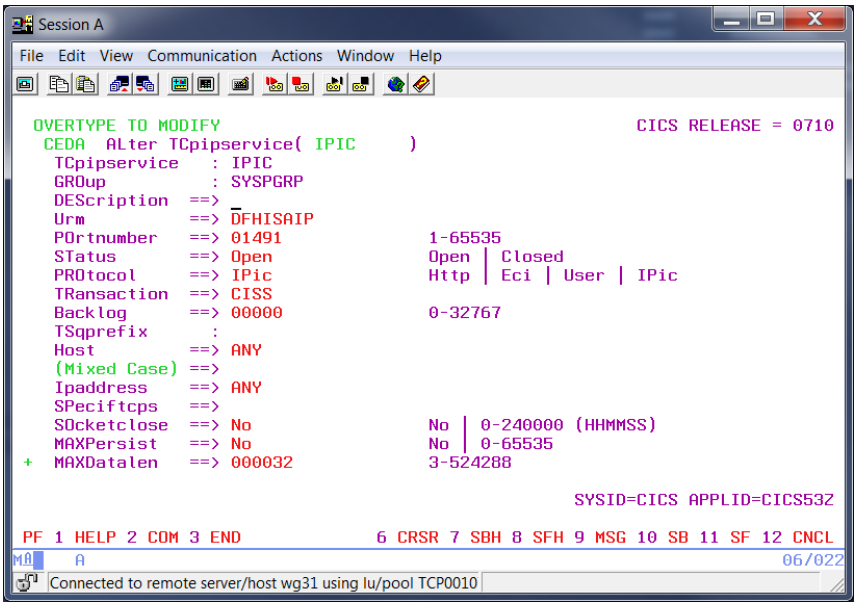
___2. Including this file adds a new feature to the server, *cicsService-1.0* which enables add support for connecting to CICS regions using IP interconnectivity (IPIC). In this scenario a CICS IPIC *TCPIPService* (see below) has been defined in the target CICS region to listen for input TCP/IP request on port 1491.

```
<server description="CICS IPIC - Catalog">

    <featureManager>
      <feature>zosconnect:cicsService-1.0</feature>
    </featureManager>

    <zosconnect_cicsIpicConnection id="catalog"
      host="wg31.washington.ibm.com"
      port="1491"/>

</server>
```

Figure 2: Contents of ipic.xml

Tech-Tip: Service names do not have to be explicitly identified in a *zosconnect_services* element. Services defined in SAR files in the services directory will be automatically installed either a server restart or based on the setting of the *updateTrigger* property (see *server.xml* above). One reason for adding explicit definitions for services (*zosconnect_services*) is when security attributes are required.

Note, the corresponding CICS TCPIPService definition.



___3. Use the **F3** key to end the edit session

___4. Adding this file using an *include* statement refreshing the z/OS Connect EE V3.0 server. Enter the following MVS command to refresh the configuration ***F BAQSTRT,ZCON,REFRESH***

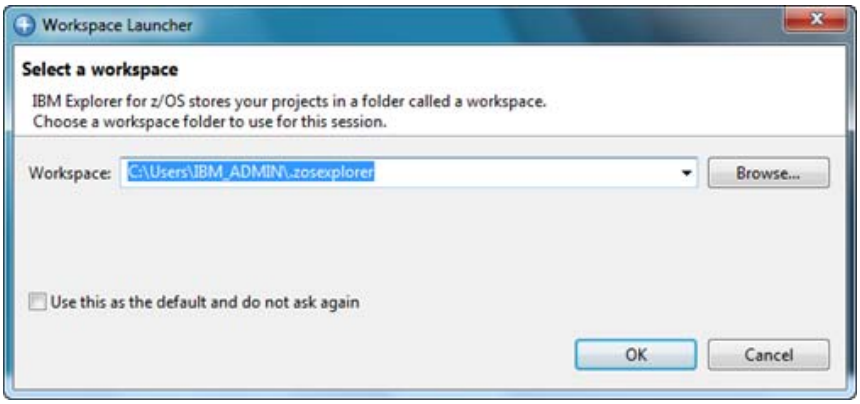
> **Tech Tip:** MVS commands can be entered using SDSF. In ISPF, enter the ISPF command ***=SDSF.LOG*** to go to SDSF and add the / (slash) prefix to the command before entering, e.g. /F BAQSTRT,ZCON,REFRESH

Mitch Johnson (mitchj@us.ibm.com)

Open z/OS Explorer and a create SAR project

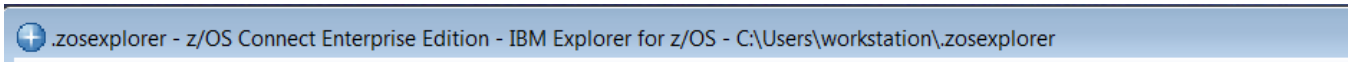
___1. On the workstation desktop, locate the *z/OS Explorer* icon and double click on it to open the tool.

___2. You will be prompted for a workspace:

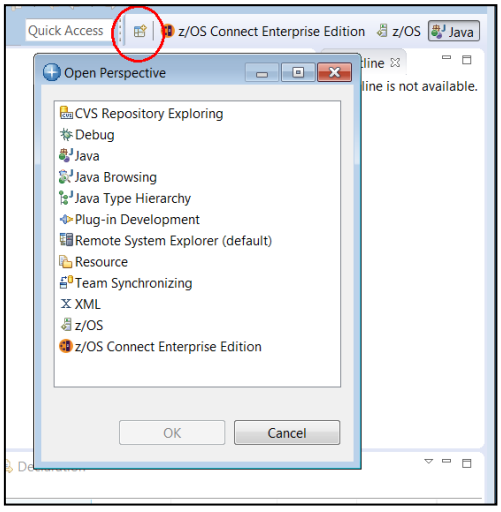


Take whatever default value is seen and click **OK**. If you see a *Welcome* tab close it by click on the white X in the tab.

___3. We have set up the image to open by default to the *z/OS Connect Enterprise Edition* perspective. Verify this by looking in the upper left corner. You should see:

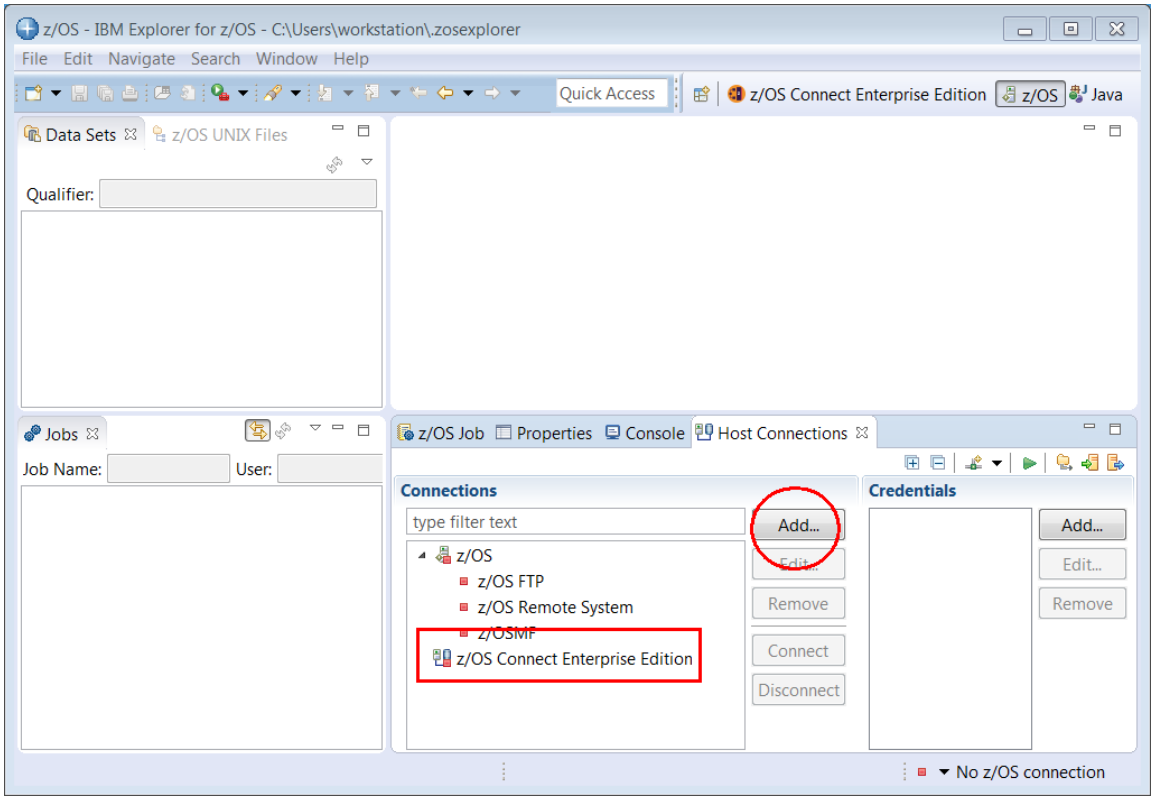


First we want to establish a connection to your z/OS Connect server. Select the *Open Perspective* icon on the top right side to display the list of available perspectives. Select z/OS and click the **OK** button.



Mitch Johnson (mitchj@us.ibm.com)

___4. The perspective name in the upper left hand should now be *z/OS – IBM Explorer for z/OS* (see below). To add a connection to your z/OS Connect Server select *z/OS Connect Enterprise Edition* connection in the *Host connection*s tab in the lower view and then click the **Add** button.



Tech-Tip: Eclipse based development tools like z/OS Explorer; provide a graphical interface consisting of multiple views within a single window.

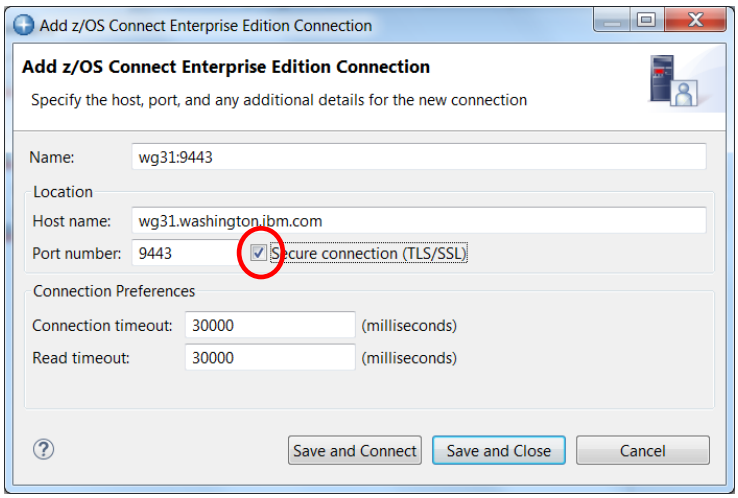
A view is an area in the window dedicated to providing a specific tool or function. For example in the window above, *Data Sets/z/OS UNIX Files* and *Jobs* are views that use different areas of the window for displaying information. At bottom on the right there is a single area for displaying the contents of four views stacked together (commonly called a *stacked views*), *z/OS Jobs, Properties, Console and Host Connections*. In a stacked view the contents of each view can be displayed by clicking on the view tab (the name of the view).

At any time a specific view can be enlarged to fill the entire window by double clicking in the view's title bar. Double clicking in the view's title bar will be restored the original arrangement. If a z/OS Explorer view is closed or otherwise disappears or if a perspective has missing views, the original arrangement can be restored by selecting Windows → Reset Perspective in the window's tool bar.

Eclipse based tools also have the ability to display multiple views based on the current role of the user. In this context a window is known as a perspective. The contents (or views) of a perspective are based on the role the user, i.e., developer or administrator.

Mitch Johnson (mitchj@us.ibm.com)

___5. In the pop-up list displayed select *z/OS Connect Enterprise Edition* and on the *Add z/OS Connect Enterprise Edition Connection* screen enter ***wg31.washington.ibm.com*** for the *Host name*, ***9443*** for the *Port Number*, check the box for *Secure connection (TLS/SSL)* and then click the **Save and Connect** button.



___6. On the *z/OS Connect Enterprise Edition – User ID* required window create new credentials for a *User ID* of ***Fred*** and a *Password or Passphrase* of ***fredpwd*** (case matters). Remember the server is configured to use basic security. Later when SAF security is enabled a valid RACF User ID and password will have to be used instead. Click **OK** to continue.

___7. Click the **Accept** button on the *Server certificate alert – Accept this certificate screen*. You may be presented with another prompt for a userid and password, enter ***Fred*** and ***fredpwd*** again.

___8. The status icon beside *wg31:9443* should now be a green circle with a lock. This shows that a secure connection has been established between the z/OS Explorer and the z/OS Connect server. A red box indicates that no connection exists.

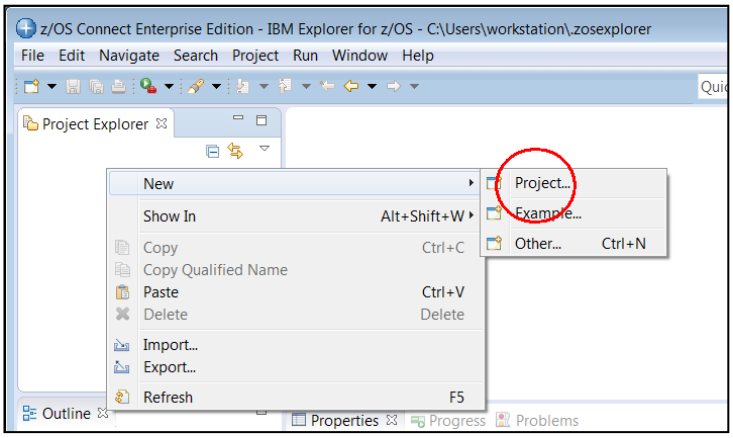
___9. Switch the perspective back to the *z/OS Connect Enterprise Edition* perspective (see Step 3 above).

Create the services

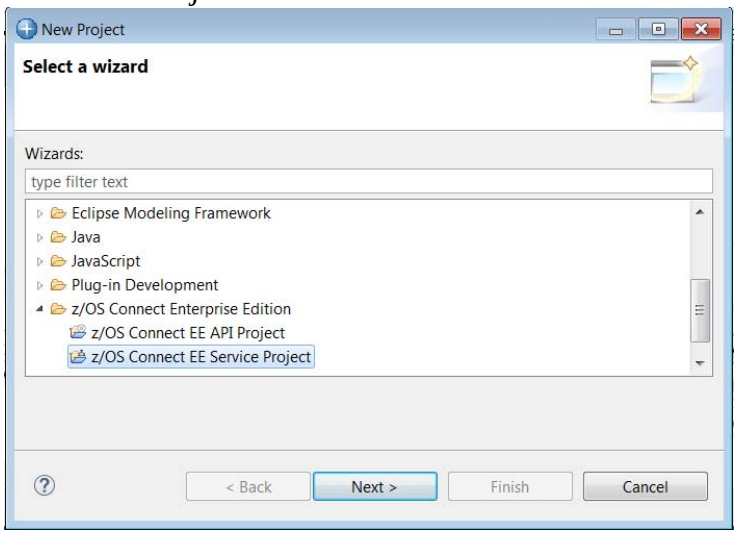
The CICS application that will be invoked by this service is CICS sample application DFH0XCMN. This program is provided with the CICS office supply sample application. DFH0XCMN is passed a COMMAREA which contains a request field. This field is used by DFH0XCMN to determine the next which of 3 functions of the sample application will be accessed. The three functions are (1) inquire on a single item in the catalog (INQS), (2) provide a list of items in the catalog starting with a specific item (INQC) or (3) place an order for an item (ORDR). The COMMAREA is based on a COPYBOOK which uses redefines the COMMAREA based on the type of request.

The next step is to create 3 services which correspond to these three functions provided by the sample program DFH0XCMN.

___1. In the upper left, position your mouse anywhere in the *Project Explorer* view and right-mouse click, then select *New →Project*:

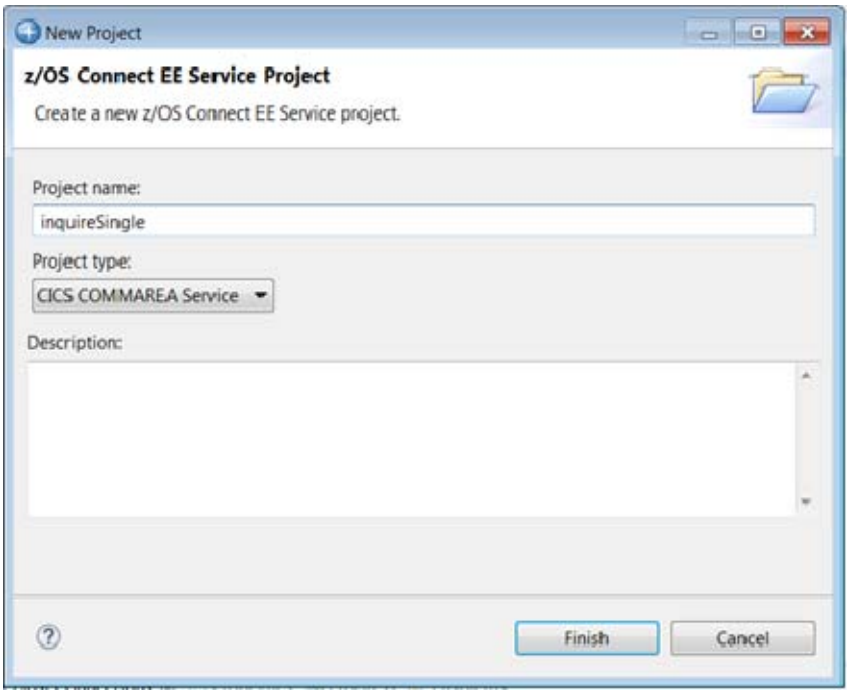


___2. In the *New Project* window, scroll down and open the *z/OS Connect Enterprise Edition* folder and select *z/OS Connect EE Service Project* and then click the **Next** button.

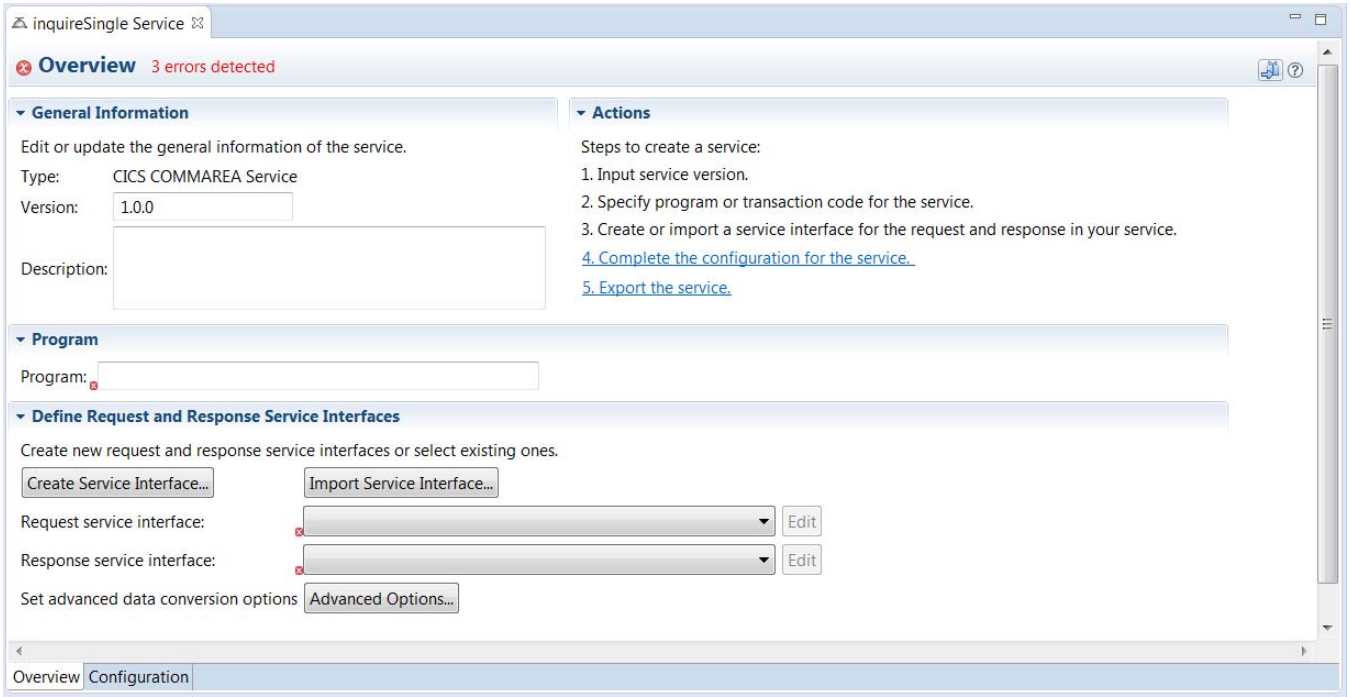


Mitch Johnson (mitchj@us.ibm.com)

___3. On the new *New Project* window enter ***inquireSingle*** as the *Project name* and use the pull down arrow to select *CICS COMMAREA Service* as the *Project type*. Click **Finish** to continue



___4. This will open the *Overview* window for the *inquireSingle Service*. For now disregard the message about the 3 errors detected, they will be addressed shortly.

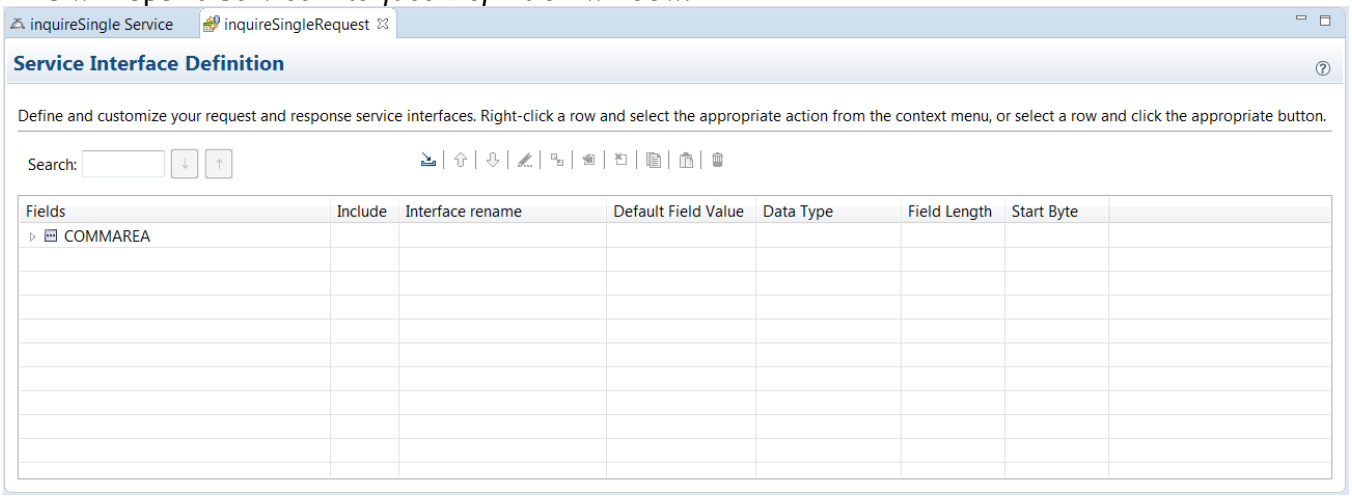


Mitch Johnson (mitchj@us.ibm.com)

___5. Next enter the target CICS program name ***DFH0XCMN*** in the area beside *Program* (case is important).

___6. Click the **Create Service Interface** button to create the first service required by this API and enter a *service interface name* of ***inquireSingleRequest***. Click **OK** to continue.

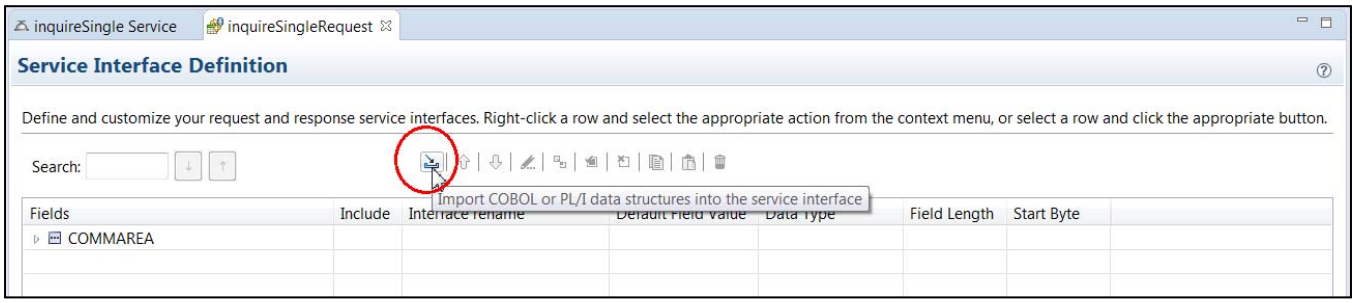
___7. This will open a *Service Interface Definition* window.



___8. The first step is to import the COBOL copy book that represents the COMMAREA when inquiring on a single item in the Catalog application. This copy book was downloaded from member DFH0XCP1 in the CICS SDFHSAMP target data set.

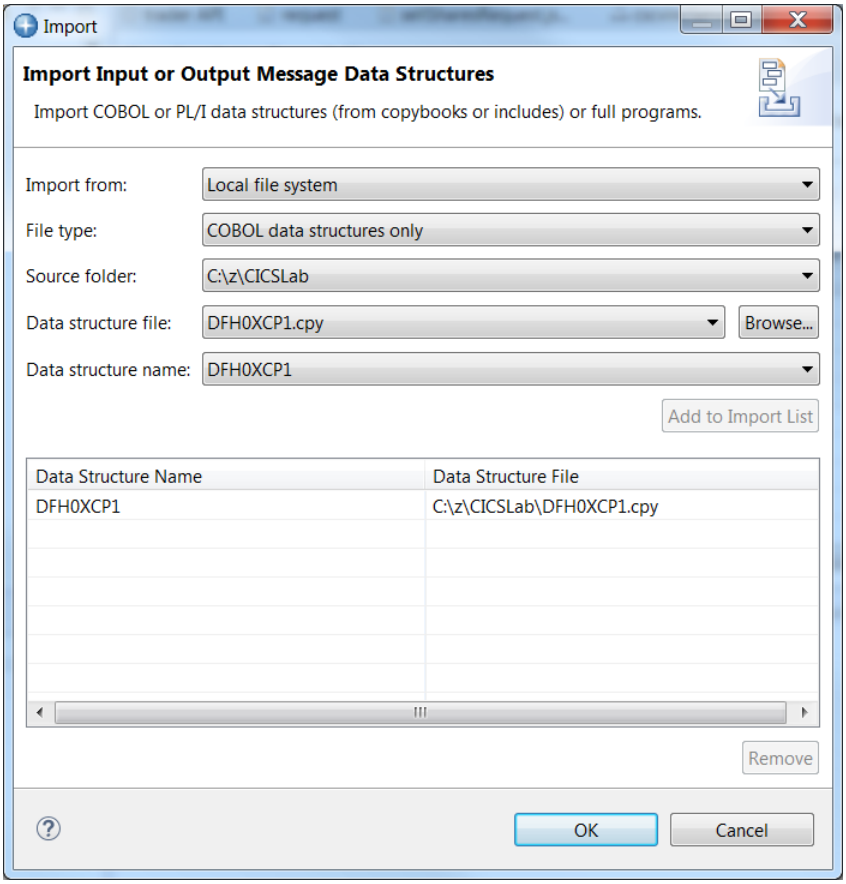
```
      *    Catalogue COMMAREA structure
           03 CA-REQUEST-ID            PIC X(6).
           03 CA-RETURN-CODE           PIC 9(2).
           03 CA-RESPONSE-MESSAGE      PIC X(79).
           03 CA-REQUEST-SPECIFIC      PIC X(911).
      *    Fields used in Inquire Catalog
           03 CA-INQUIRE-REQUEST REDEFINES CA-REQUEST-SPECIFIC.
               05 CA-LIST-START-REF        PIC 9(4).
               05 CA-LAST-ITEM-REF         PIC 9(4).
               05 CA-ITEM-COUNT            PIC 9(3).
               05 CA-INQUIRY-RESPONSE-DATA PIC X(900).
               05 CA-CAT-ITEM  REDEFINES CA-INQUIRY-RESPONSE-DATA
                              OCCURS 15 TIMES.
                   07 CA-ITEM-REF          PIC 9(4).
                   07 CA-DESCRIPTION       PIC X(40).
                   07 CA-DEPARTMENT        PIC 9(3).
                   07 CA-COST              PIC X(6).
                   07 IN-STOCK             PIC 9(4).
                   07 ON-ORDER             PIC 9(3).
      *    Fields used in Inquire Single
           03 CA-INQUIRE-SINGLE REDEFINES CA-REQUEST-SPECIFIC.
               05 CA-ITEM-REF-REQ          PIC 9(4).
               05 FILLER                   PIC 9(4).
               05 FILLER                   PIC 9(3).
               05 CA-SINGLE-ITEM.
                   07 CA-SNGL-ITEM-REF     PIC 9(4).
                   07 CA-SNGL-DESCRIPTION  PIC X(40).
                   07 CA-SNGL-DEPARTMENT   PIC 9(3).
                   07 CA-SNGL-COST         PIC X(6).
                   07 IN-SNGL-STOCK        PIC 9(4).
                   07 ON-SNGL-ORDER        PIC 9(3).
               05 FILLER                   PIC X(840).
      *    Fields used in Place Order
           03 CA-ORDER-REQUEST REDEFINES CA-REQUEST-SPECIFIC.
               05 CA-USERID                PIC X(8).
               05 CA-CHARGE-DEPT           PIC X(8).
               05 CA-ITEM-REF-NUMBER       PIC 9(4).
               05 CA-QUANTITY-REQ          PIC 9(3).
               05 FILLER                   PIC X(888).
```

___9. On the *Service Interface Definition* window there is a tool bar near the top. If you hover over an icon its function will be display as below. Click the *Import COBOL or PL/I data structures into the service interface* icon to start the import process.



Mitch Johnson (mitchj@us.ibm.com)

___10. This will open the *Import* window. On this window select *Local file system* as source of the import and *COBOL data structure only* as the *File type*. Press the **Browse** button and **Open** directory *C:\z\CICSLAB* and then select file *DFH0XCP1*.cpy and click **Open** to import this file into this project. Click the **Add to Import List** button and then click **OK** to continue.



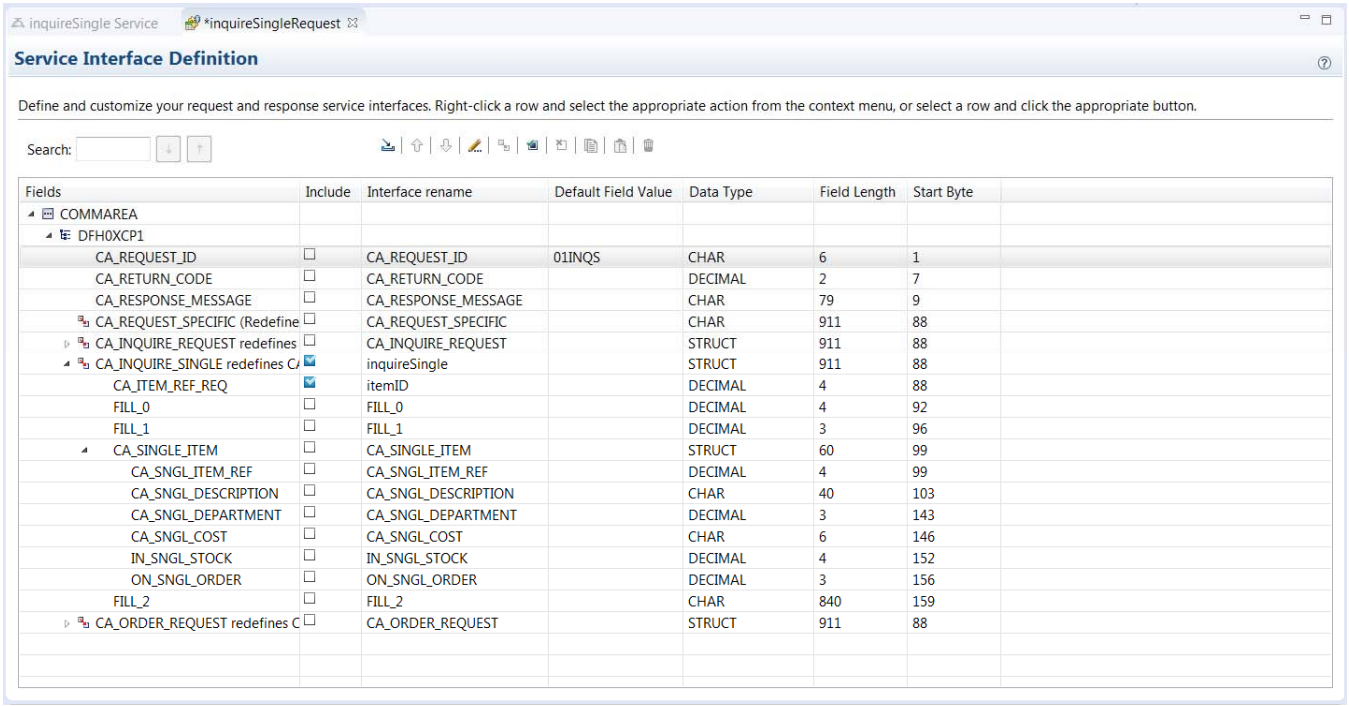
Mitch Johnson (mitchj@us.ibm.com)

___11. When you expand *COMMAREA* you will see the COBOL 'variables' that have been imported into the service project. In this service we want to return a single item and the only fields that need to be exposed in the request message is the item number.

a. First uncheck the boxes under *Include* column so that only *CA_INQUIRE_SINGLE* and *CA_ITEM_REF_REQ* are checked. These are the fields that will be exposed to the requestor of this API.

b. Double click an entry in the *Interface rename* column to open it for editing. Rename interface fields *CA_INQUIRE_SINGLE* to ***inquireSingle*** and ***CA_ITEM_REF_REQ*** to ***itemID***. Change interface field *CA_REQUEST_ID* by setting its default value to ***01INQS***.

When finished the *Service Interface Definition* should look like the below:

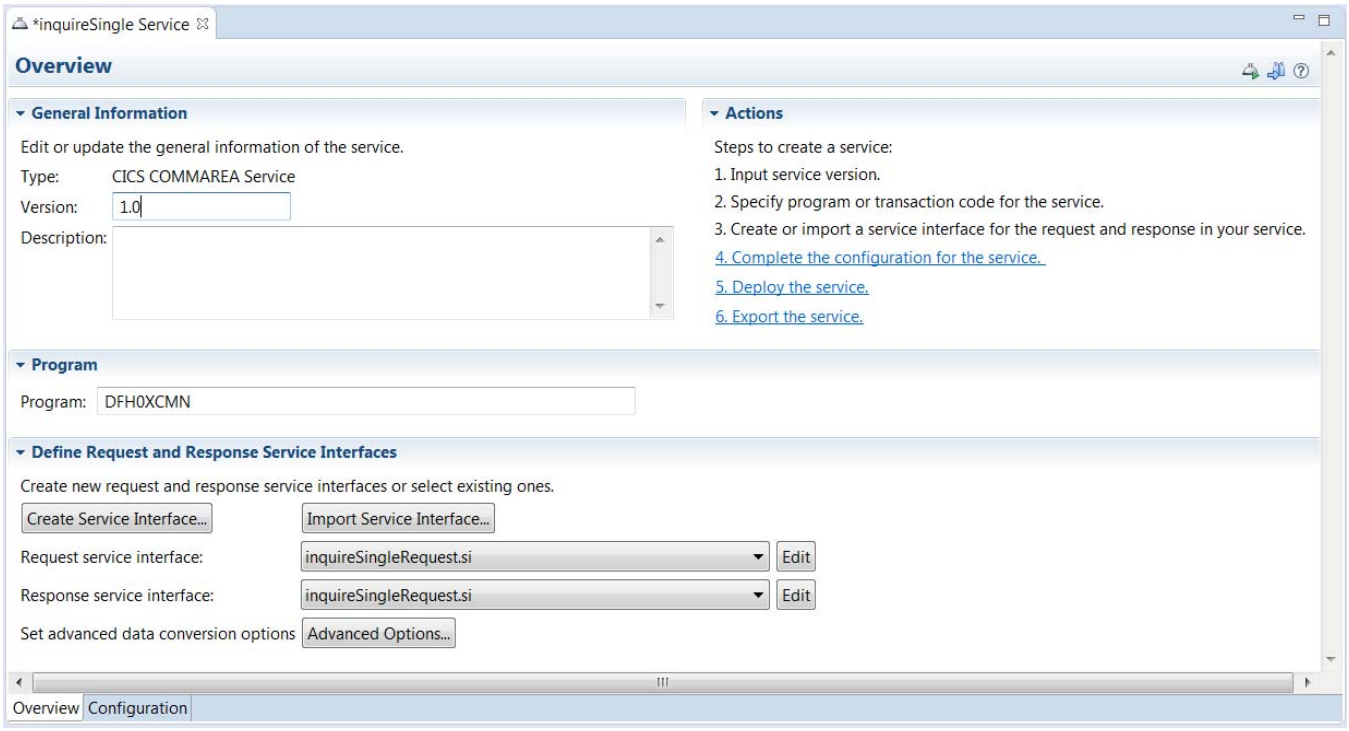


Fields	Include	Interface rename	Default Field Value	Data Type	Field Length	Start Byte
COMMAREA						
DFH0XCP1						
CA_REQUEST_ID	☐	CA_REQUEST_ID	01INQS	CHAR	6	1
CA_RETURN_CODE	☐	CA_RETURN_CODE		DECIMAL	2	7
CA_RESPONSE_MESSAGE	☐	CA_RESPONSE_MESSAGE		CHAR	79	9
CA_REQUEST_SPECIFIC (Redefine	☐	CA_REQUEST_SPECIFIC		CHAR	911	88
CA_INQUIRE_REQUEST redefines	☐	CA_INQUIRE_REQUEST		STRUCT	911	88
CA_INQUIRE_SINGLE redefines C/	☑	inquireSingle		STRUCT	911	88
CA_ITEM_REF_REQ	☑	itemID		DECIMAL	4	88
FILL_0	☐	FILL_0		DECIMAL	4	92
FILL_1	☐	FILL_1		DECIMAL	3	96
CA_SINGLE_ITEM	☐	CA_SINGLE_ITEM		STRUCT	60	99
CA_SNGL_ITEM_REF	☐	CA_SNGL_ITEM_REF		DECIMAL	4	99
CA_SNGL_DESCRIPTION	☐	CA_SNGL_DESCRIPTION		CHAR	40	103
CA_SNGL_DEPARTMENT	☐	CA_SNGL_DEPARTMENT		DECIMAL	3	143
CA_SNGL_COST	☐	CA_SNGL_COST		CHAR	6	146
IN_SNGL_STOCK	☐	IN_SNGL_STOCK		DECIMAL	4	152
ON_SNGL_ORDER	☐	ON_SNGL_ORDER		DECIMAL	3	156
FILL_2	☐	FILL_2		CHAR	840	159
CA_ORDER_REQUEST redefines C	☐	CA_ORDER_REQUEST		STRUCT	911	88

N.B. the original interface names were derived from the COBOL source shown earlier.

Mitch Johnson (mitchj@us.ibm.com)

___12. Close the *Service Interface Definition* window by clicking on the white X in the tab being sure to save the changes. Note now that the *Request service interface* and the *Response service interface* areas have now been populated with *inquireSingleRequest.si*. Also note that you can use their respective **Edit** buttons to return to the *Service Interface Definition* window for each interface.

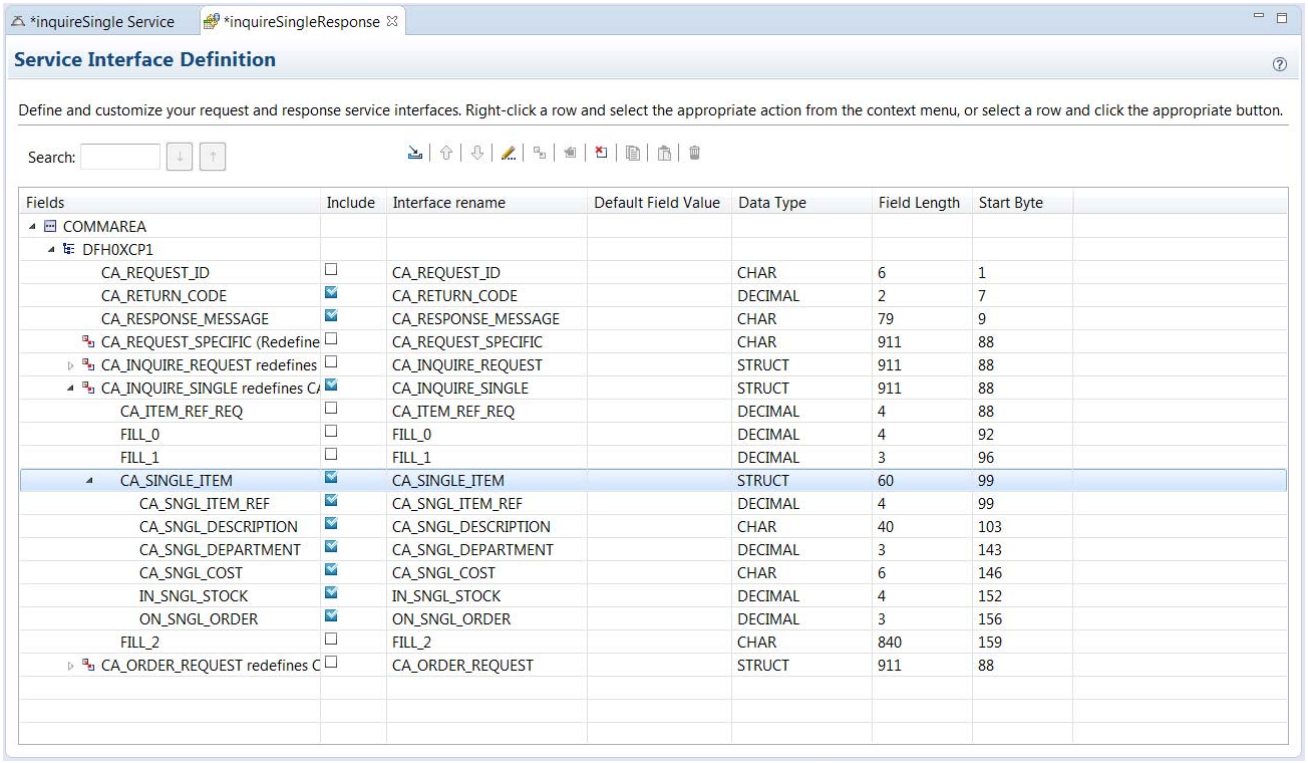


Tech-Tip: Note that even though the program uses a COMMAREA we can easily distinguish which fields are provided in a request and which fields will be in the response by providing different service interfaces. This is particularly useful when using containers.

___13. Repeat steps 6 through 10 to create a service interface for the *inquireSingle* service response message. Enter a *service interface name* of ***inquireSingleResponse*** name rather than ***inquireSingleRequest***.

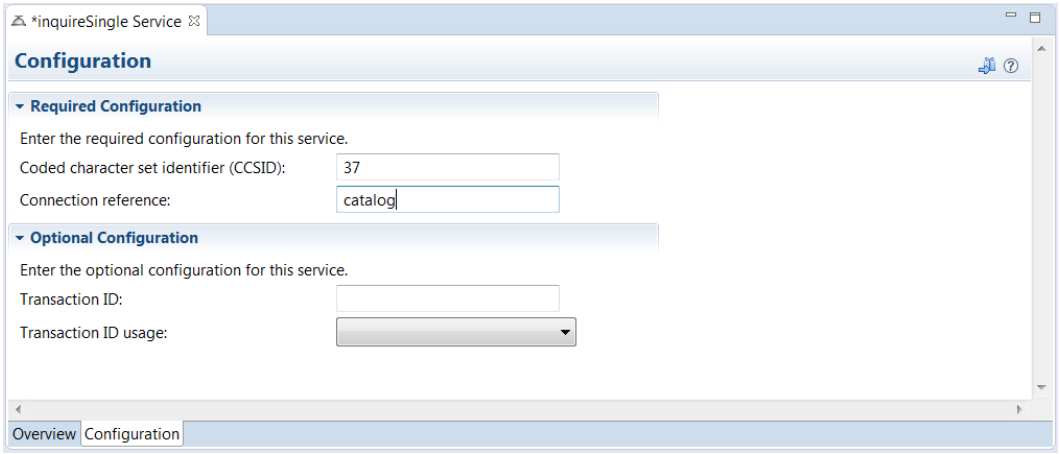
Mitch Johnson (mitchj@us.ibm.com)

___14. A different set of fields will be returned in the response Uncheck the boxes under *Include* column so that only *CA_RETURN_CODE, CA_RESPONSE_MESSAGE, CA_INQUIRE_SINGLE* and *CA_SINGLE_ITEM* checked. These are fields that will be exposed to the requestor of this API in the response. You can rename these fields if you choose. If you do some of the later screen shots may be different.



___15. Close the *Service Interface Definition* window.

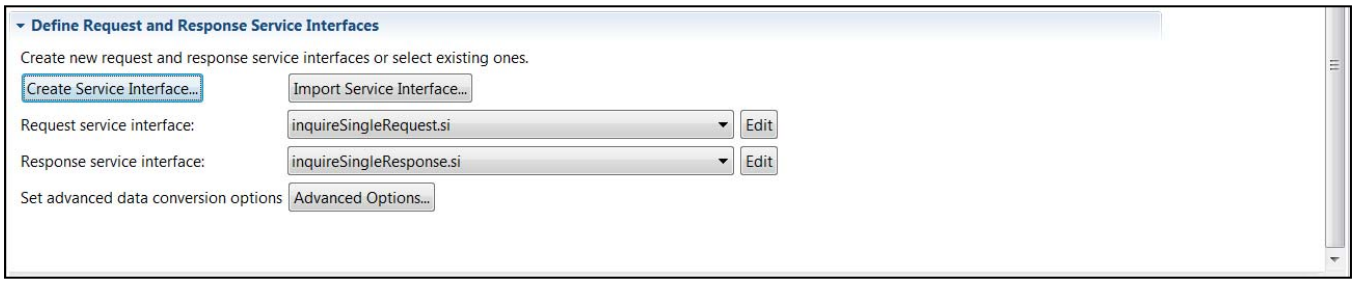
___16. Next we need to identify a connection reference for which CICS region will be used. Click on the *Configuration* tab at the bottom of the *Overview* window to display the *Configuration* window. Enter ***catalog*** in the area beside *Connection reference*.



Mitch Johnson (mitchj@us.ibm.com)

Tech-Tip: The *Connection reference* identifies the *cicsIpicConnection* element or *cicsLocalConnection* element to be used to access a CICS region in the z/OS Connect EE configuration**,** see ipic.xml on page 26

___17. Verify that the request and response service interfaces are set correctly. If not, set them as below.



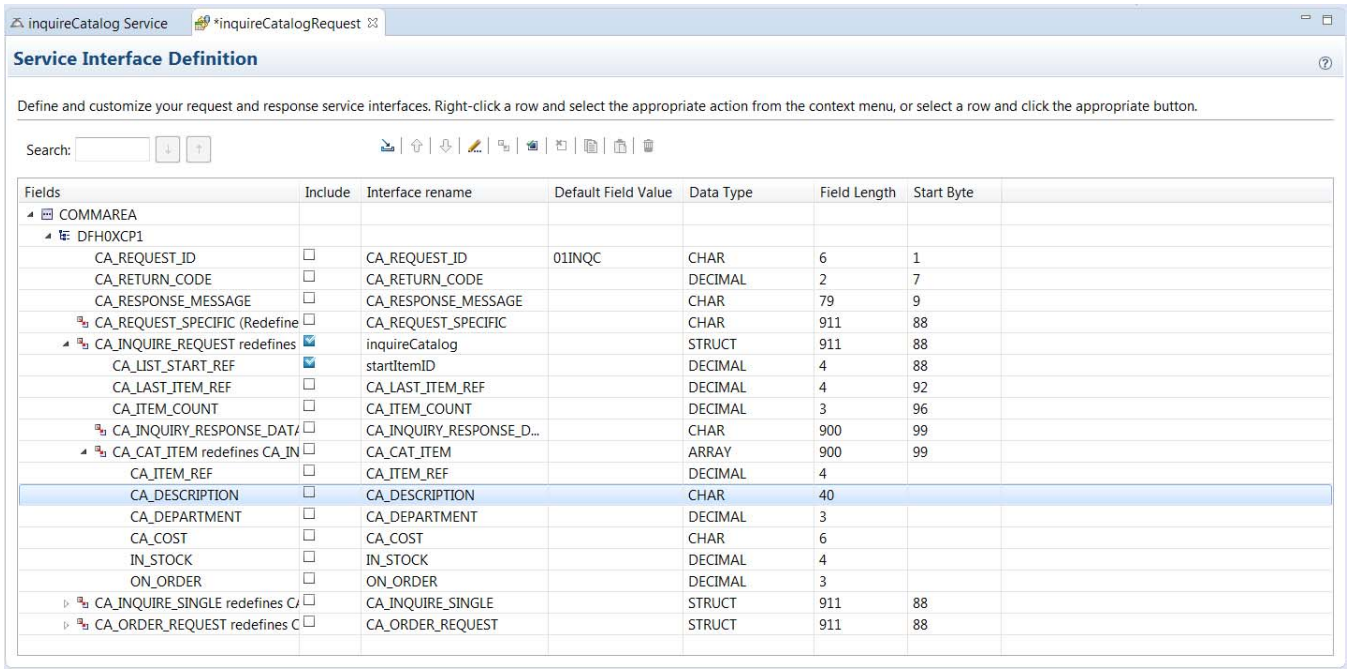
___18. Save the *inquireSingle* service by closing all open tabs.

___19. Repeat steps 1 through 5 to create the *inquireCatalog* service.

___20. Repeat steps 6 through 10 to create a service interface for *inquireCatalog* service request message. Enter a name of ***inquireCatalogRequest***.

___21. Expand the COMMAREA and perform the following steps for the *inquireCatalog* request:

a. First uncheck the boxes under *Include* column so that only *CA_INQUIRE_REQUEST* and *CA_LIST_START_REF* are checked. These are the fields that will be exposed to the requestor of this API.



Fields	Include	Interface rename	Default Field Value	Data Type	Field Length	Start Byte
COMMAREA						
DFH0XCP1						
CA_REQUEST_ID	☐	CA_REQUEST_ID	01INQC	CHAR	6	1
CA_RETURN_CODE	☐	CA_RETURN_CODE		DECIMAL	2	7
CA_RESPONSE_MESSAGE	☐	CA_RESPONSE_MESSAGE		CHAR	79	9
CA_REQUEST_SPECIFIC (Redefine	☐	CA_REQUEST_SPECIFIC		CHAR	911	88
CA_INQUIRE_REQUEST redefines	☑	inquireCatalog		STRUCT	911	88
CA_LIST_START_REF	☑	startItemID		DECIMAL	4	88
CA_LAST_ITEM_REF	☐	CA_LAST_ITEM_REF		DECIMAL	4	92
CA_ITEM_COUNT	☐	CA_ITEM_COUNT		DECIMAL	3	96
CA_INQUIRY_RESPONSE_DAT/	☐	CA_INQUIRY_RESPONSE_D...		CHAR	900	99
CA_CAT_ITEM redefines CA_IN	☐	CA_CAT_ITEM		ARRAY	900	99
CA_ITEM_REF	☐	CA_ITEM_REF		DECIMAL	4	
CA_DESCRIPTION	☐	CA_DESCRIPTION		CHAR	40	
CA_DEPARTMENT	☐	CA_DEPARTMENT		DECIMAL	3	
CA_COST	☐	CA_COST		CHAR	6	
IN_STOCK	☐	IN_STOCK		DECIMAL	4	
ON_ORDER	☐	ON_ORDER		DECIMAL	3	
CA_INQUIRE_SINGLE redefines C/	☐	CA_INQUIRE_SINGLE		STRUCT	911	88
CA_ORDER_REQUEST redefines C	☐	CA_ORDER_REQUEST		STRUCT	911	88

b. Double click an entry in the *Interface rename* column to open it for editing. Rename interface fields *CA_INQUIRE_REQUEST* to ***inquireCatalog*** and ***CA_LIST_START_REF*** to ***startItemID***.

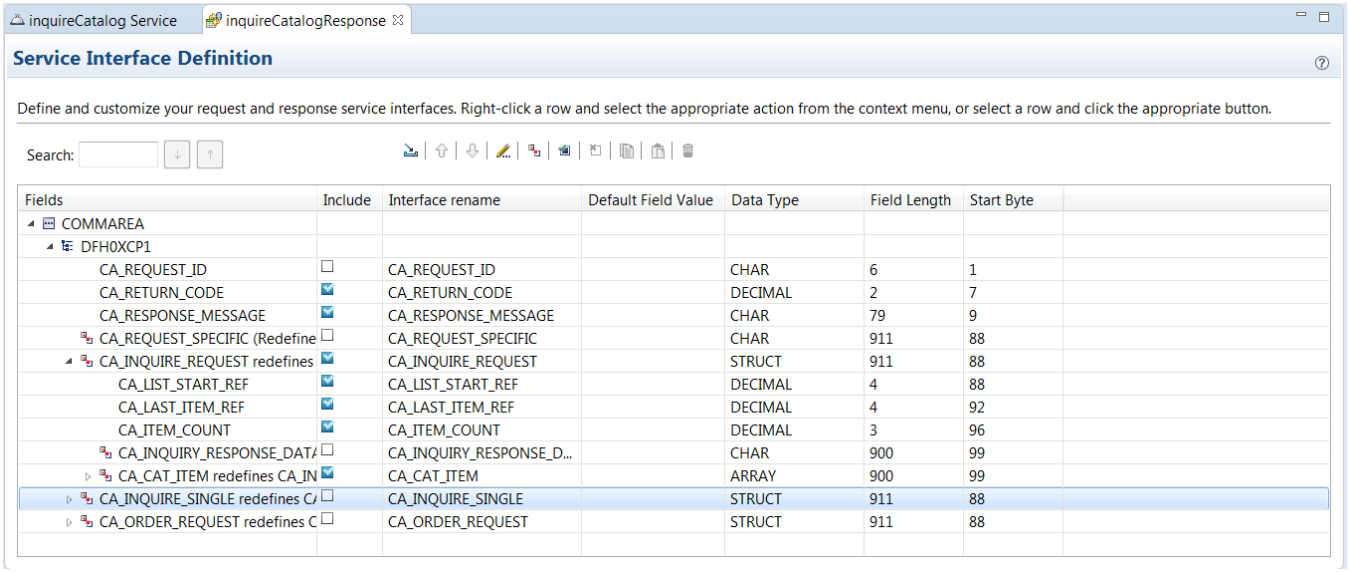
c. Change interface field *CA_REQUEST_ID* by setting its default value to ***01INQC***.

Mitch Johnson (mitchj@us.ibm.com)

___22. Close the *Service Interface Definition* window.

___23. Repeat steps 6 through 10 to create a service interface for the *inquireCatalog* response message with the *service interface name* of ***inquireCatalogResponse***.

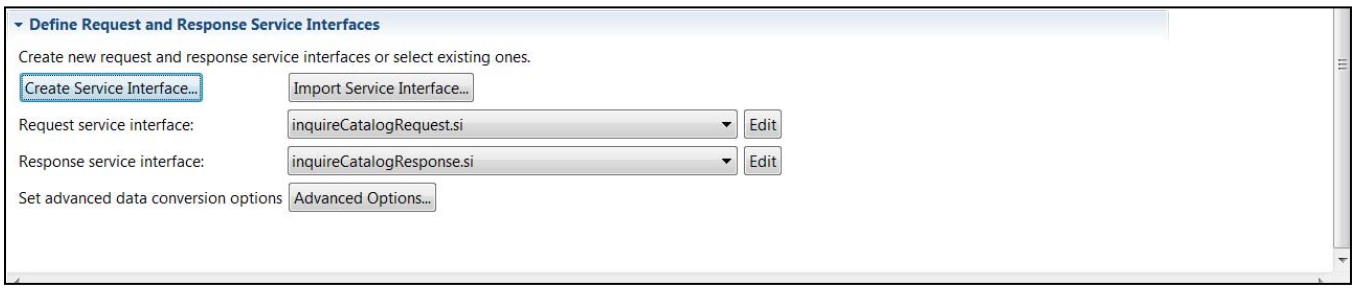
___24. A different set of fields will be returned in the response. Uncheck the boxes under *Include* column so that only *CA_RETURN_CODE, CA_RESPONSE_MESSAGE, CA_INQUIRE_REQUEST, CA_LIST_START_REF, CA_LAST_ITEM_REF, CA_ITEM_COUNT* and *CA_CAT_ITEM* are checked. These are fields that will be exposed to the requestor of this API in the response. Rename these fields if you choose.



___25. Close the *Service Interface Definition* window.

___26. Identify the connection reference for which CICS region will be used for this service. Click on the Configuration tab at the bottom of the *Overview* window to display the *Configuration* window. Enter ***catalog*** in the area beside *Connection reference*.

___27. Verify that the request and response service interfaces are set correctly, if not set them as below.



Mitch Johnson (mitchj@us.ibm.com)

___28. Save the *inquireCatalog* service by closing all open tabs.

___29. Create a *placeOrder* service by repeating steps 1 through 5.

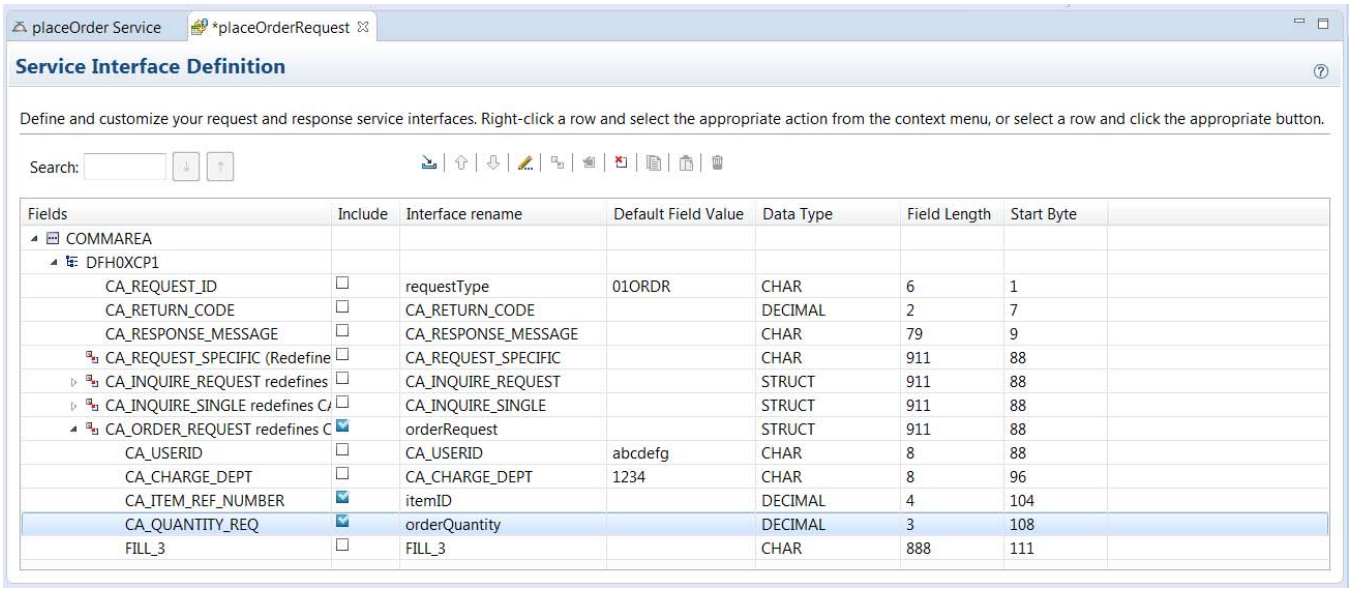
___30. Repeat steps 6 through 10 to create *service interface name* ***placeOrderRequest***

___31. Expand the COMMAREA and perform the following steps for the *placeOrderRequest*:

a. First uncheck the boxes under *Include* column so that only *CA_ORDER_REQUEST, CA_ITEM_REF_NUMBER and CA_QUANTITY_REQ* are checked. These are the fields that will be exposed to the requestor of this API.

b. Double click an entry in the *Interface rename* column to open it for editing. Rename interface fields *CA_ORDER_REQUEST* to ***orderRequest***, ***CA_ITEM_REF_NUMBER*** to ***itemID*** and CQ_QUANTITY_REQ to ***orderQuantity***. Rename interface field *CA_REQUEST_ID* to ***requestType*** and set its default value to ***01ORDR***.

c. Set the default values for *CA_USERID* to ***abcdefgh*** and *CA_CHARGE_DEPT* to ***1234***.



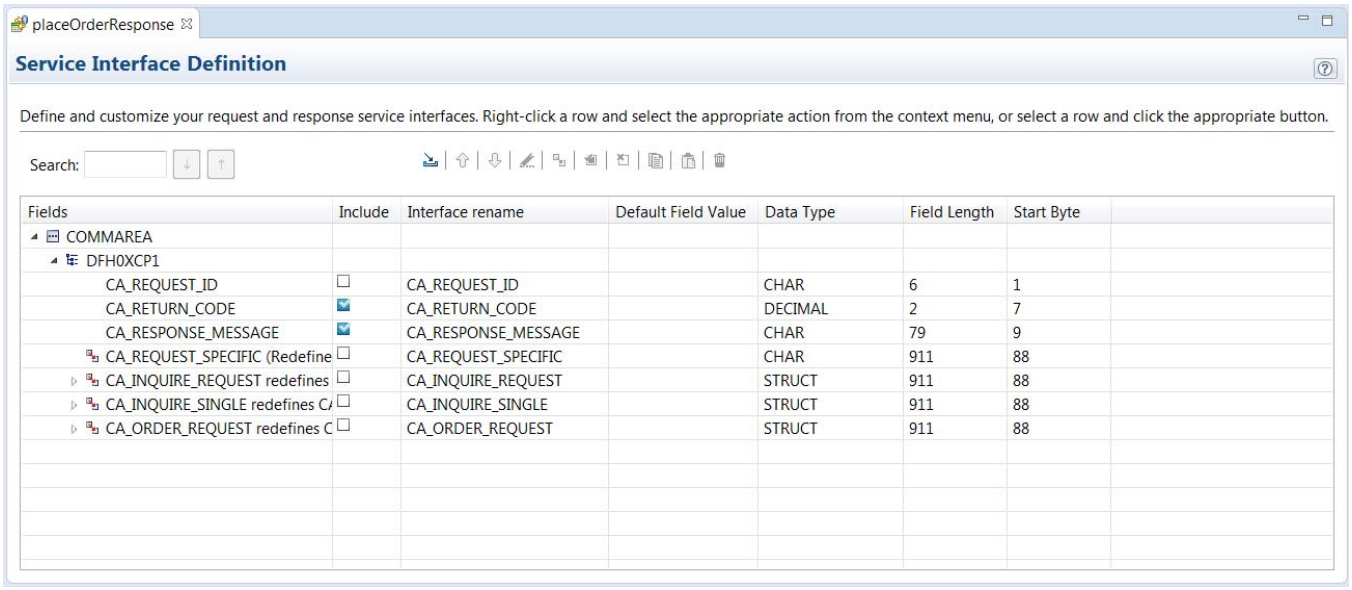
Fields	Include	Interface rename	Default Field Value	Data Type	Field Length	Start Byte
COMMAREA						
DFH0XCP1						
CA_REQUEST_ID	☐	requestType	01ORDR	CHAR	6	1
CA_RETURN_CODE	☐	CA_RETURN_CODE		DECIMAL	2	7
CA_RESPONSE_MESSAGE	☐	CA_RESPONSE_MESSAGE		CHAR	79	9
CA_REQUEST_SPECIFIC (Redefine	☐	CA_REQUEST_SPECIFIC		CHAR	911	88
CA_INQUIRE_REQUEST redefines	☐	CA_INQUIRE_REQUEST		STRUCT	911	88
CA_INQUIRE_SINGLE redefines CA	☐	CA_INQUIRE_SINGLE		STRUCT	911	88
CA_ORDER_REQUEST redefines C	☑	orderRequest		STRUCT	911	88
CA_USERID	☐	CA_USERID	abcdefg	CHAR	8	88
CA_CHARGE_DEPT	☐	CA_CHARGE_DEPT	1234	CHAR	8	96
CA_ITEM_REF_NUMBER	☑	itemID		DECIMAL	4	104
CA_QUANTITY_REQ	☑	orderQuantity		DECIMAL	3	108
FILL_3	☐	FILL_3		CHAR	888	111

___32. Close the *Service Interface Definition* window.

___33. Repeat steps 6 through 10 to create a service interface ***placeOrderResponse*** response message for the *placeOrder* service.

___34. A different set of fields will be returned in the response. Uncheck the boxes under *Include* column so that only *CA_RETURN_CODE* and *CA_RESPONSE_MESSAGE*, are checked. These are fields that will be exposed to the requestor of this API in the response. Rename these fields if you choose.

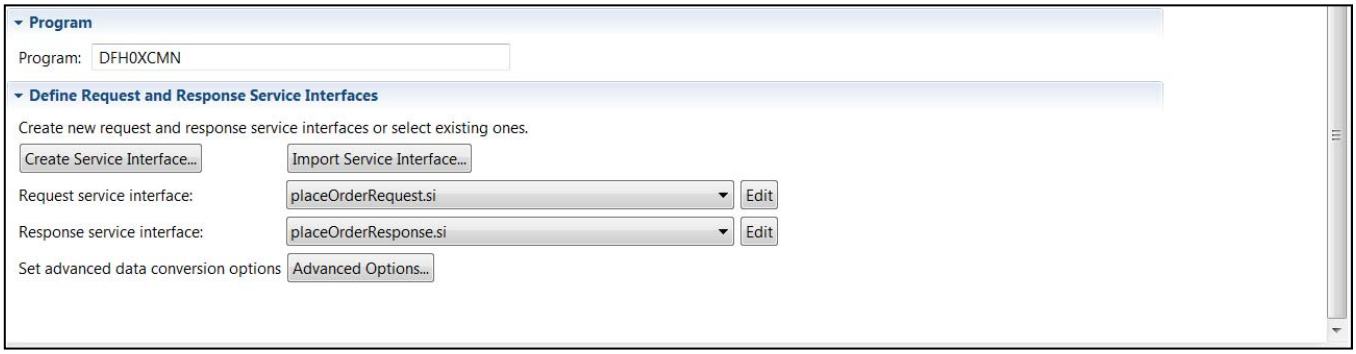
Mitch Johnson (mitchj@us.ibm.com)



___35. Close the *Service Interface Definition* window.

___36. Identify the connection reference for which CICS region will be used for this service. Click on the Configuration tab at the bottom of the *Overview* window to display the *Configuration* window. Enter ***catalog*** in the area beside *Connection reference*.

___37. Verify the request and response services interfaces are set correctly, if not set them as below.



___38. Save the *placeOrder* service either by closing all open tabs.

These three services now need to be make available for developing the API and for deployment to the z/OS Connect EE server.

Mitch Johnson (mitchj@us.ibm.com)

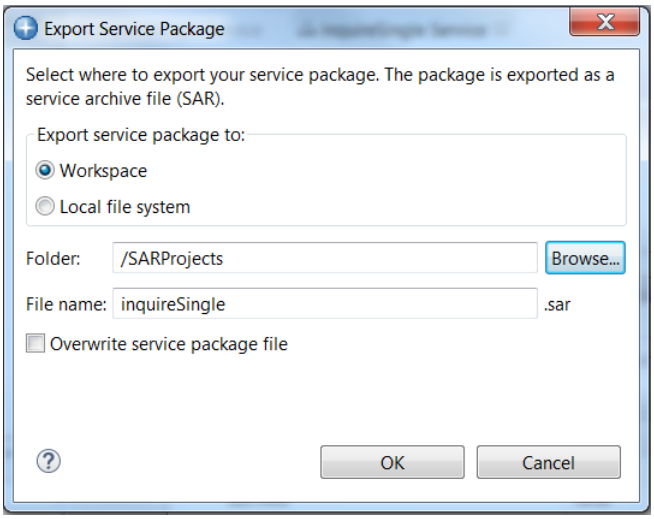
Export and deploy the Service Archive files

Before a Service Interface can be used it must be exported to create Service Archive (SAR) file). There are two uses for a SAR file. The first is for use in developing an API in the z/OS Connect EE Toolkit and the second is for deploying to a z/OS Connect EE server. This section describes the process for creating and exporting SAR files.

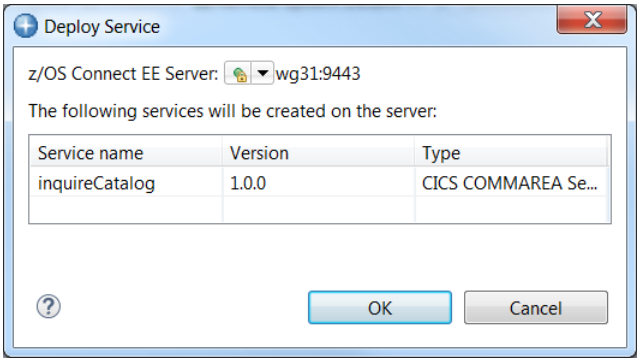
___1. First 'export' them into another project in the z/OS Connect EE Toolkit. Select ***File*** on the tool bar and then on the pop up select **New → Project**. Expand the *General* folder and select *Project* to create a target project for exporting the Service Archive (SAR) files. Click **Next** to continue.

___2. On the *New Project* window enter ***SARProjects*** as the *Project name*. Click **Finish** to continue. This action will add a new project in the *Project Explorer* named *SARProjects*.

___3. Select the *inquireCatalog* service project and right mouse button click. On the pop-up selection select **z/OS Connect EE → Export z/OS Connect EE Service Archive**. On the *Export Services Package* window select the radio button beside *Workspace* and use the **Browse** button to select the *SARProjects* folder. Click **OK** to continue.



___4. While the *inquireCatalog* project is still selected, right mouse button click again and on the pop-up selection select **z/OS Connect EE → Deploy Service to z/OS Connect EE Server**. On the *Deploy Service* window select the target server (*wg31:9443*) and click **OK** twice to continue.



Mitch Johnson (mitchj@us.ibm.com)

___5. Repeat these two steps to export the *inquireSingle* and *placeOrder* service project to both the *SARProjects* folder and target z/OS Connect EE server.

___6. When finished the *SARProjects* folder should contain 3 SAR files.

Mitch Johnson (mitchj@us.ibm.com)

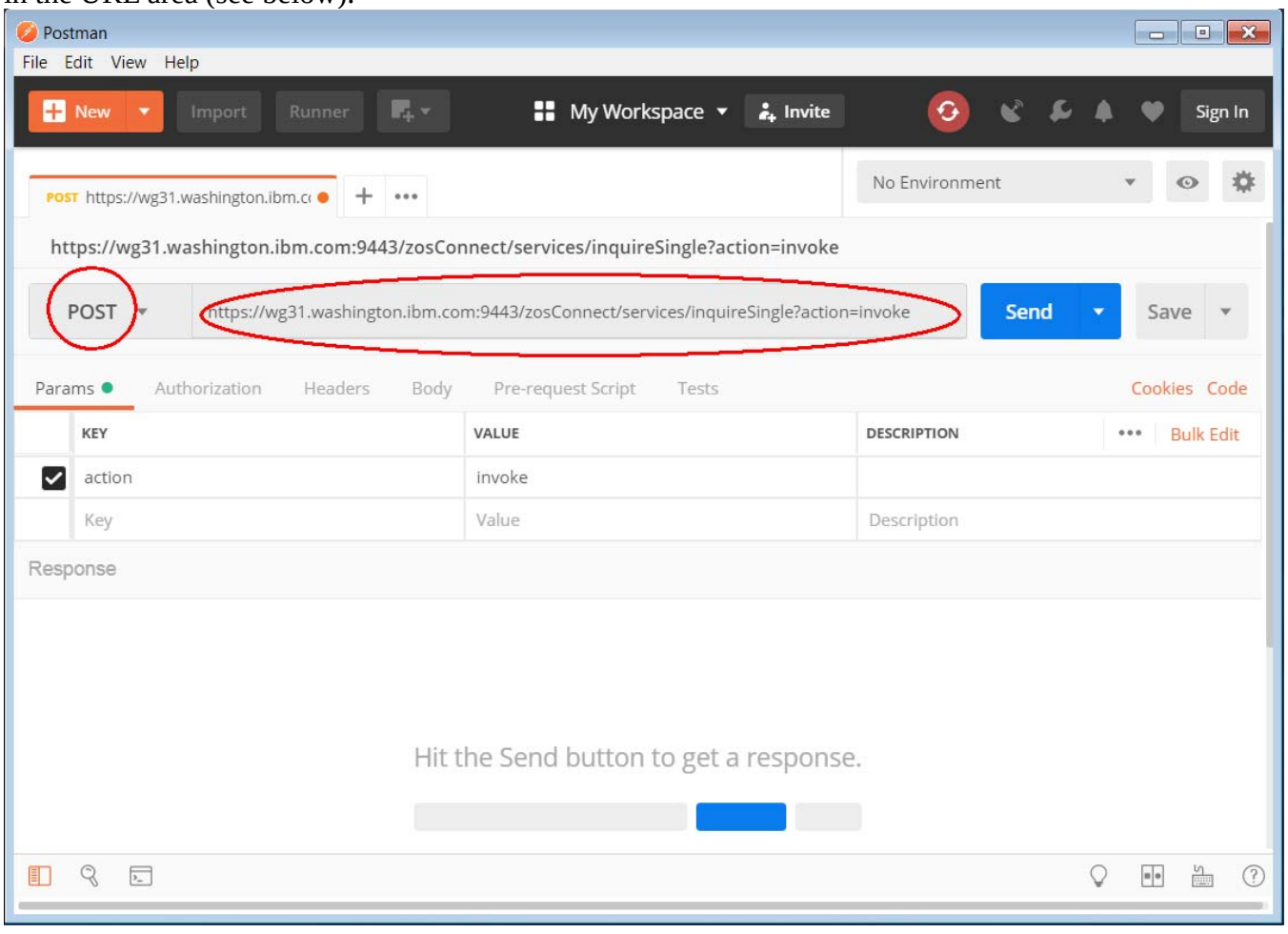
Test the Services

The services should be tested before developing an API to ensure the infrastructure and the request and response messages are as expected.

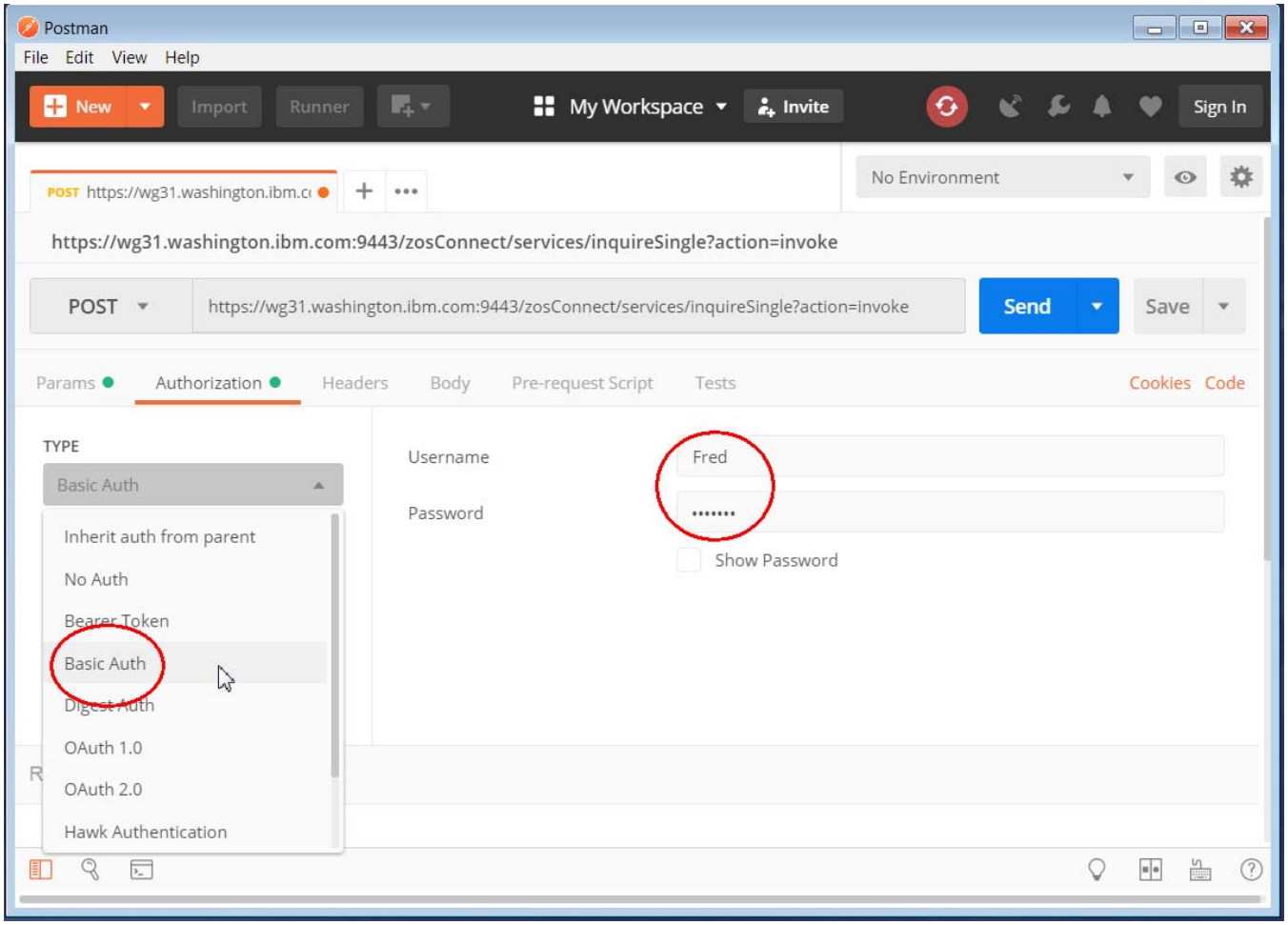
Two products which seem to be most popular tools for testing RESTful APIs used to test the services. The two products are *Postman* which is available for downloading from https://www.getpostman.com/apps and *cURL (client URL)* which is available for downloading from https://curl.haxx.se/download.html. The use of both will be shown in this section of the exercise.

Using Postman

___1. Open the *Postman* tool icon on the desktop and if necessary reply to any prompts and close any welcome messages, use the down arrow to select **POST** and enter ***https://wg31.washington.ibm.com:9443/zosConnect/services/inquireSingle?action=invoke*** in the URL area (see below).

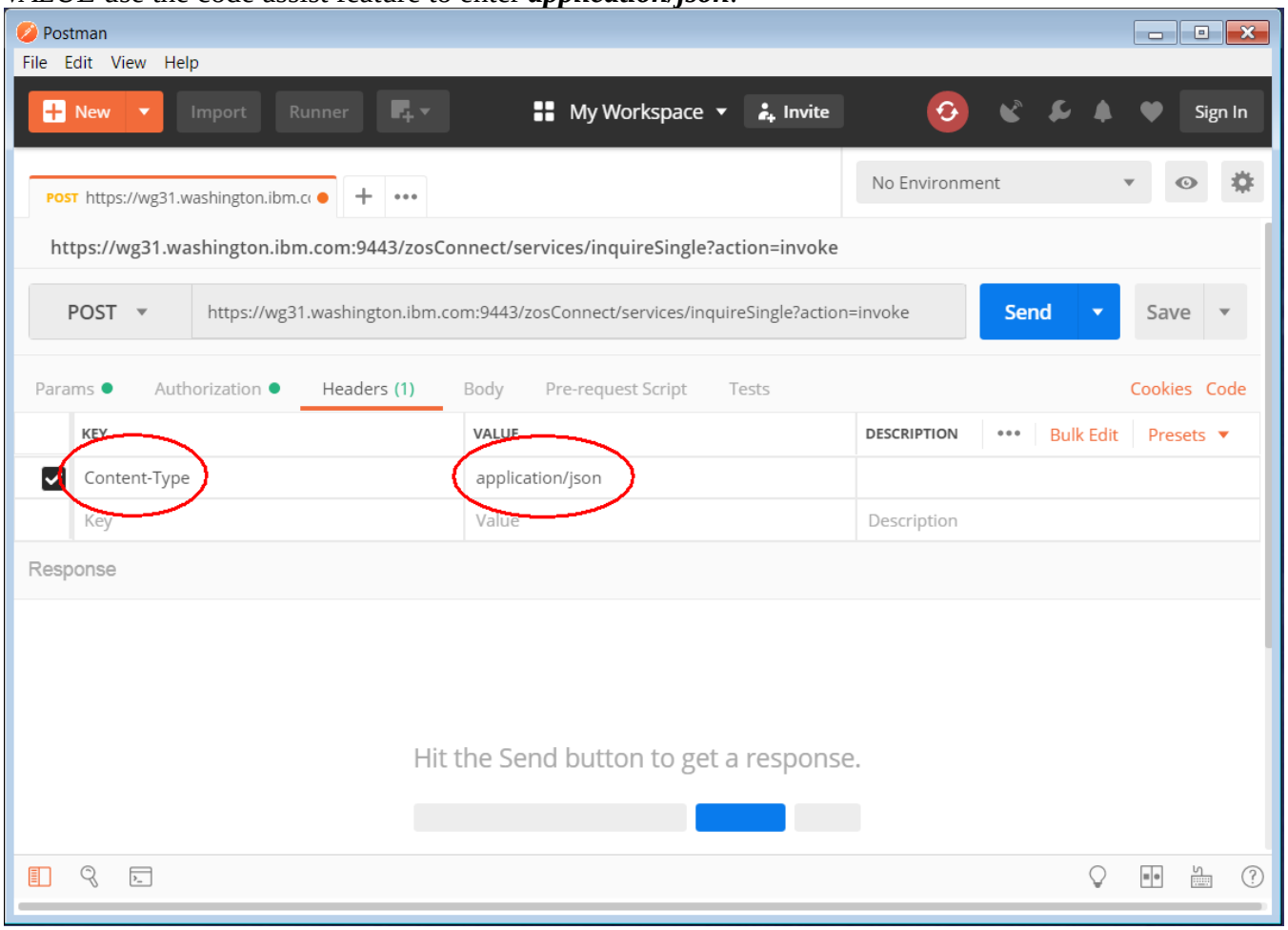


___2. No *query* or *path* parameters are required so next select the *Authorization* tab to enter an authorization identity and password. Use the pull down arrow to select *Basic Auth* and enter ***Fred*** as the username and ***fredpwd*** as the Password.



Mitch Johnson (mitchj@us.ibm.com)

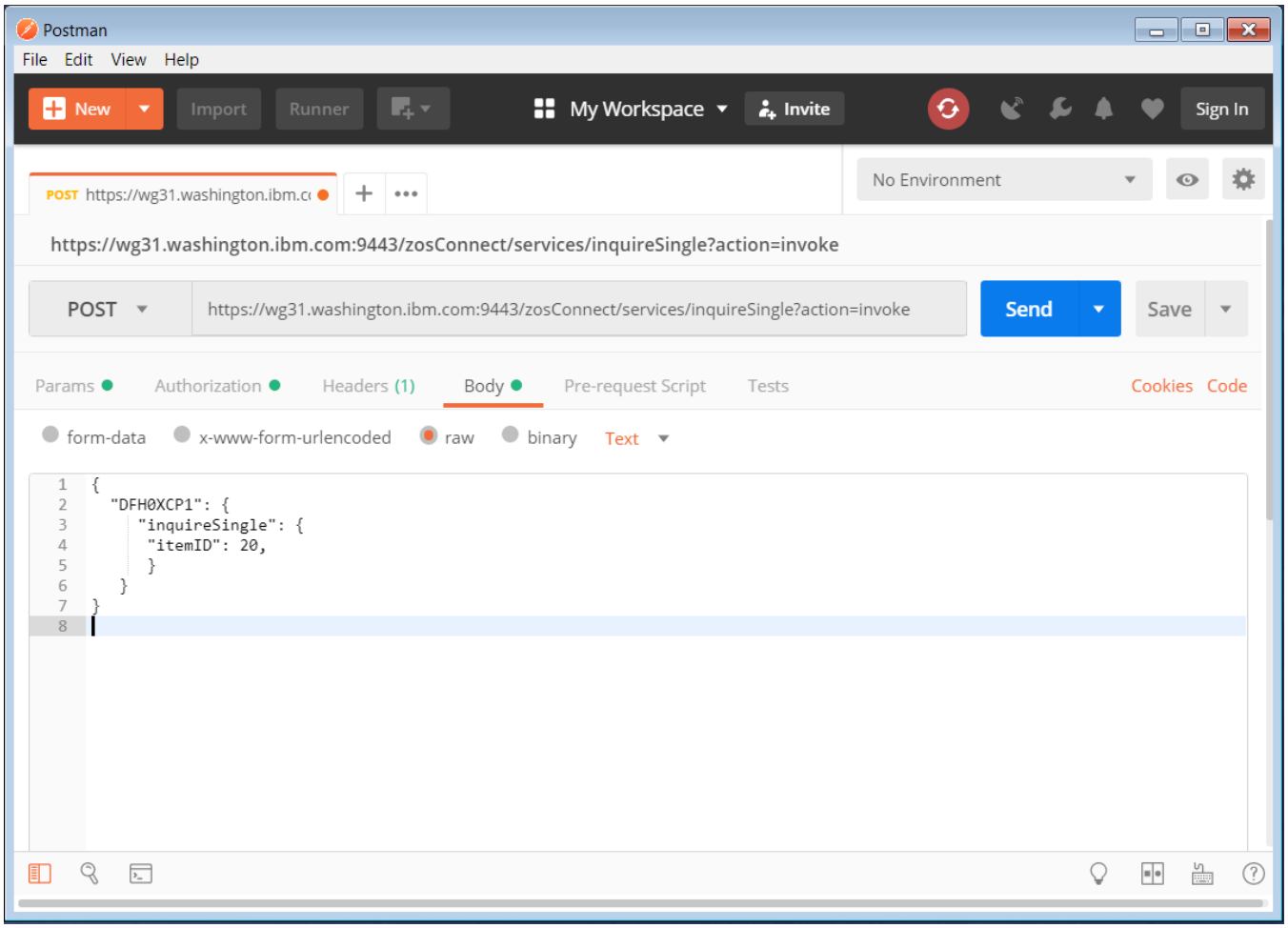
___3. Next select the *Headers* tab and under *KEY* use the code assist feature to enter ***Content-Type*** and under *VALUE* use the code assist feature to enter ***application/json***.



Tech-Tip: Code assist simply means that when text is entered in field, all the valid values for that field that match the typed text will be displayed. You can select the desired value for the field from the list displayed and that value will populate that field.

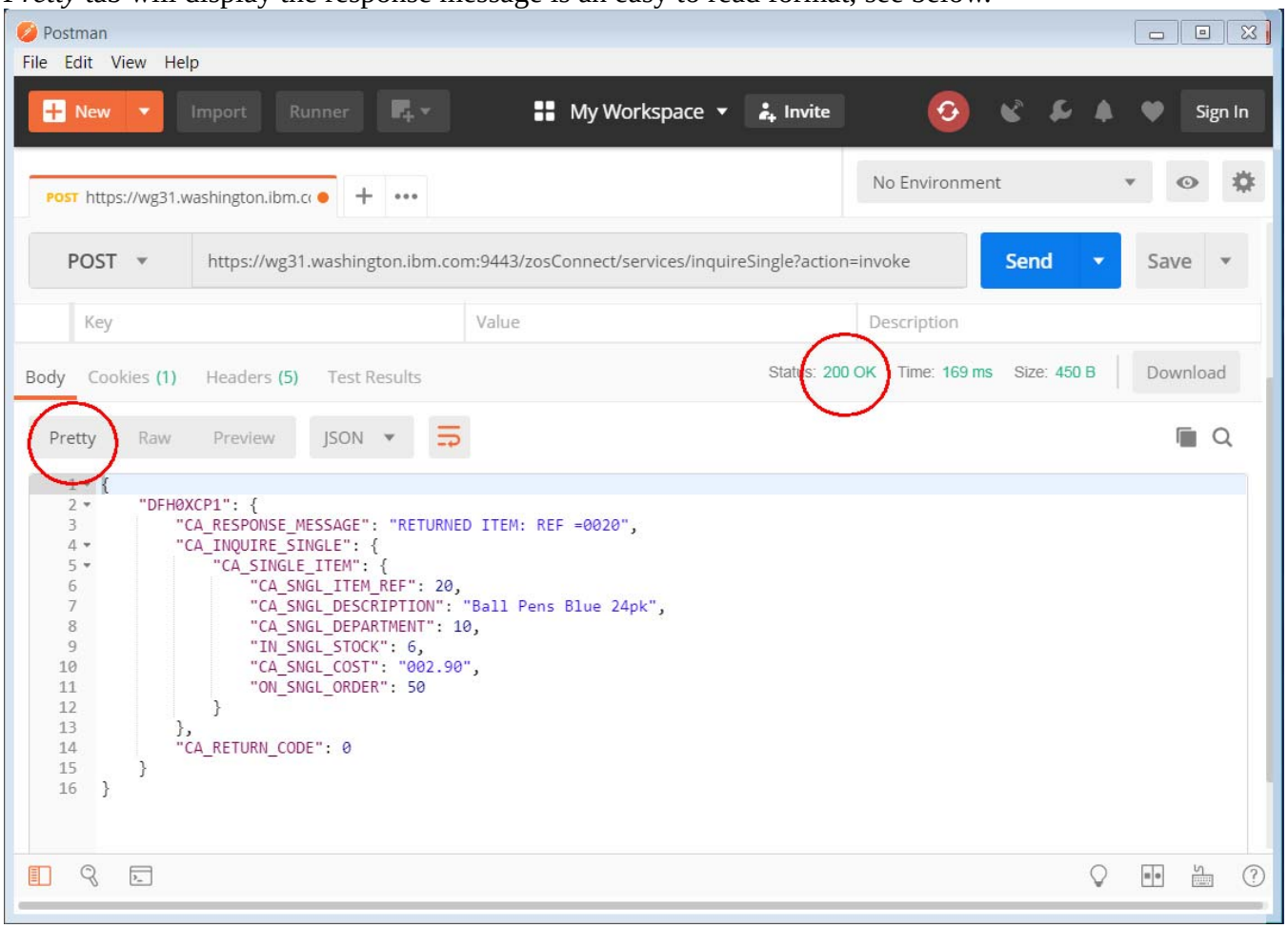
___4. Next select the *Body* tab and select the *raw* radio button and enter the JSON message below in the *Body* area and press the S**end** button.

```
{
  "DFH0XCP1": {
     "inquireSingle": {
      "itemID": 20,
      }
   }
}
```



Mitch Johnson (mitchj@us.ibm.com)

___5. Pressing the **Send** button invokes the API. The Status of request should be 200 OK and pressing the *Pretty* tab will display the response message is an easy to read format, see below.



Using cURL

The *cURL* tool provides a command line interface to REST APIs. The same service just tested with *Postman* can be tested with *cURL* as shown here.

___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\CICSLab, e.g.
cd c:\z\CICSLAB

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

curl -X POST --user Fred:fredpwd --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)

Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Users\workstation>***cd c:\z\cicslab***

c:\z\CICSLAB>***curl -X POST --user Fred:fredpwd --header "Content-Type: applicatio /json" -d @inquireSingle.json --insecure https://wg31.washington.ibm.com:9443/zosConnect/services/inquireSingle?action=invoke***
{"DFH0XCP1":{"CA_RESPONSE_MESSAGE":"RETURNED ITEM: REF =0020","CA_INQUIRE_SINGLE":{"CA_SINGLE_ITEM":{"CA_SNGL_ITEM_REF":20,"CA_SNG L_DESCRIPTION":"Ball Pens Blue 24pk", "CA_SNGL_DEPARTMENT":10, "IN_SNGL_STOCK":6, "CA_SNGL_COST":"002.90", "ON_SNGL_ORDER":50}}, "CA_RETURN_CODE":0}}

Tech-Tip: In the above example:

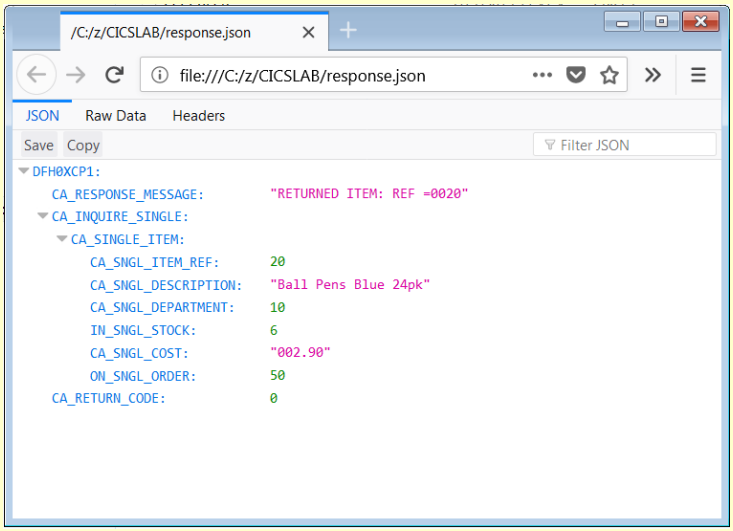
--user Fred:fredpwd could have been specified as ***--header "Authorization: Basic RnJlZDpmcmVkcHdk"***

inquireSingle.json is a file in the same directory that contains the JSON request message

--insecure is a *cURL* directive that tells *cURL* to ignore the self-signed certificate sent by the z/OS Connect EE server

The text in green is the JSON response message.

Tech-Tip: Another useful cURL directive is *-o response.json*
When this directive is used the JSON response message is written to a file named *response.json* which then can be opened with Firefox and viewed in a more readable format, e.g. command *firefox file:///c:/z/cicslab/response.json*



___4. The *inquireCatalog* service can be tested with *Postman* or *cURL* with URL
https://wg31.washington.ibm.com:9443/zosConnect/services/inquireCatalog?action=invoke
and JSON request message.

```
{
  "DFH0XCP1": {
     "inquireCatalog": {
      "startItemID": 20
      }
   }
}
```

The corresponding cURL command and results are shown below:

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

Mitch Johnson (mitchj@us.ibm.com)

```
c:\z\CICSLAB>curl -X POST --user Fred:fredpwd --header "Content-Type: applicatio
n/json" -d @inquireCatalog.json --insecure https://wg31.washington.ibm.com:9443/
zosConnect/services/inquireCatalog?action=invoke
{"DFH0XCP1":{"CA_RESPONSE_MESSAGE":"+15 ITEMS RETURNED","CA_INQUIRE_REQUEST":{"C
A_LAST_ITEM_REF":150,"CA_CAT_ITEM":[{"ON_ORDER":0,"CA_ITEM_REF":10,"CA_COST":"00
2.90","IN_STOCK":135,"CA_DESCRIPTION":"Ball Pens Black 24pk","CA_DEPARTMENT":10}
,{"ON_ORDER":50,"CA_ITEM_REF":20,"CA_COST":"002.90","IN_STOCK":6,"CA_DESCRIPTION
":"Ball Pens Blue 24pk","CA_DEPARTMENT":10},{"ON_ORDER":0,"CA_ITEM_REF":30,"CA_C
OST":"002.90","IN_STOCK":106,"CA_DESCRIPTION":"Ball Pens Red 24pk","CA_DEPARTMEN
T":10},{"ON_ORDER":0,"CA_ITEM_REF":40,"CA_COST":"002.90","IN_STOCK":80,"CA_DESCR
IPTION":"Ball Pens Green 24pk","CA_DEPARTMENT":10},{"ON_ORDER":0,"CA_ITEM_REF":5
0,"CA_COST":"001.78","IN_STOCK":83,"CA_DESCRIPTION":"Pencil with eraser 12pk","C
A_DEPARTMENT":10},{"ON_ORDER":40,"CA_ITEM_REF":60,"CA_COST":"003.89","IN_STOCK":
13,"CA_DESCRIPTION":"Highlighters Assorted 5pk","CA_DEPARTMENT":10},{"ON_ORDER":
20,"CA_ITEM_REF":70,"CA_COST":"007.44","IN_STOCK":101,"CA_DESCRIPTION":"Laser Pa
per 28-lb 108 Bright 500\/ream","CA_DEPARTMENT":10},{"ON_ORDER":0,"CA_ITEM_REF":
80,"CA_COST":"033.54","IN_STOCK":25,"CA_DESCRIPTION":"Laser Paper 28-lb 108 Brig
```

The *placeOrder* service can be tested using *Postman* or *cURL* with URL:
https://wg31.washington.ibm.com:9443/zosConnect/services/placeOrder?action=invoke

and JSON request message.

```
{
  "DFH0XCP1": {
     "orderRequest": {
      "itemID": 70,
      "orderQuantity": 1
      }
   }
}
```

The corresponding cURL command and results are shown below:

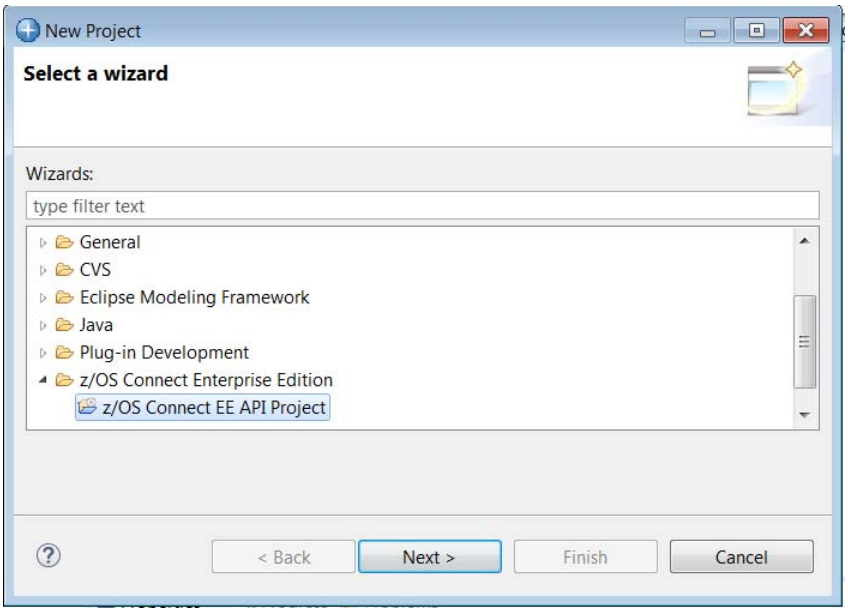
curl -X POST --user Fred:fredpwd --header "Content-Type: application/json" -d @placeOrder.json --insecure https://wg31.washington.ibm.com:9443/zosConnect/services/placeOrder?action=invoke

```
c:\z\CICSLAB>curl -X POST --user Fred:fredpwd --header "Content-Type: applicatio
n/json" -d @placeOrder.json --insecure https://wg31.washington.ibm.com:9443/zosC
onnect/services/placeOrder?action=invoke
{"DFH0XCP1":{"CA_RESPONSE_MESSAGE":"ORDER SUCCESSFULLY PLACED","CA_RETURN_CODE":
0}}
```


Mitch Johnson (mitchj@us.ibm.com)

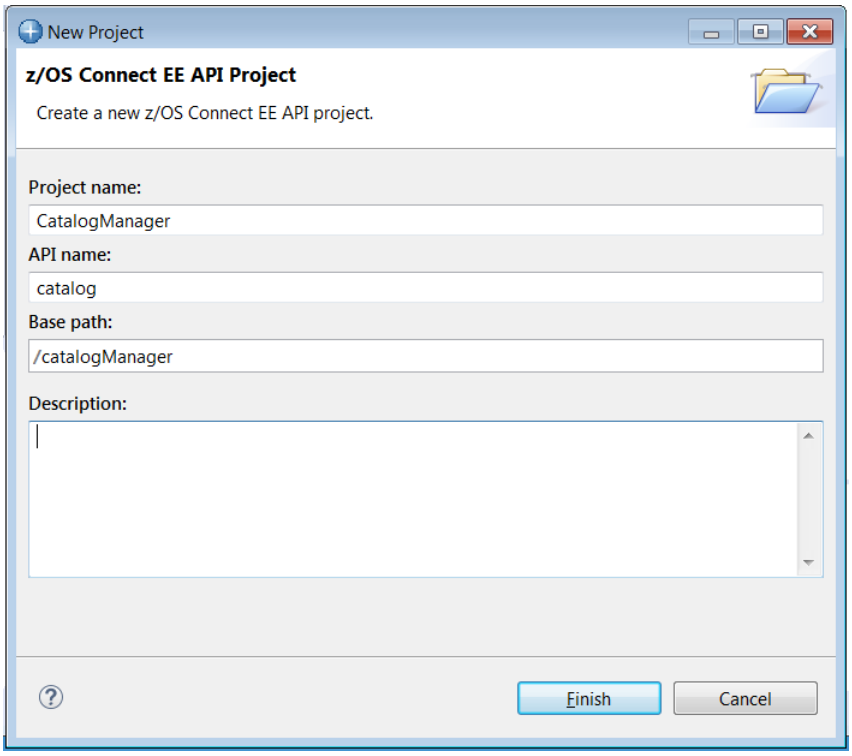
Create the API Project

___1. Back in the z/OS Connect EE tool kit create a new project. In the *New Project* window, scroll down and open the *z/OS Connect Enterprise Edition* folder and select *z/OS Connect EE API Project* and then click the **Next** button.



Mitch Johnson (mitchj@us.ibm.com)

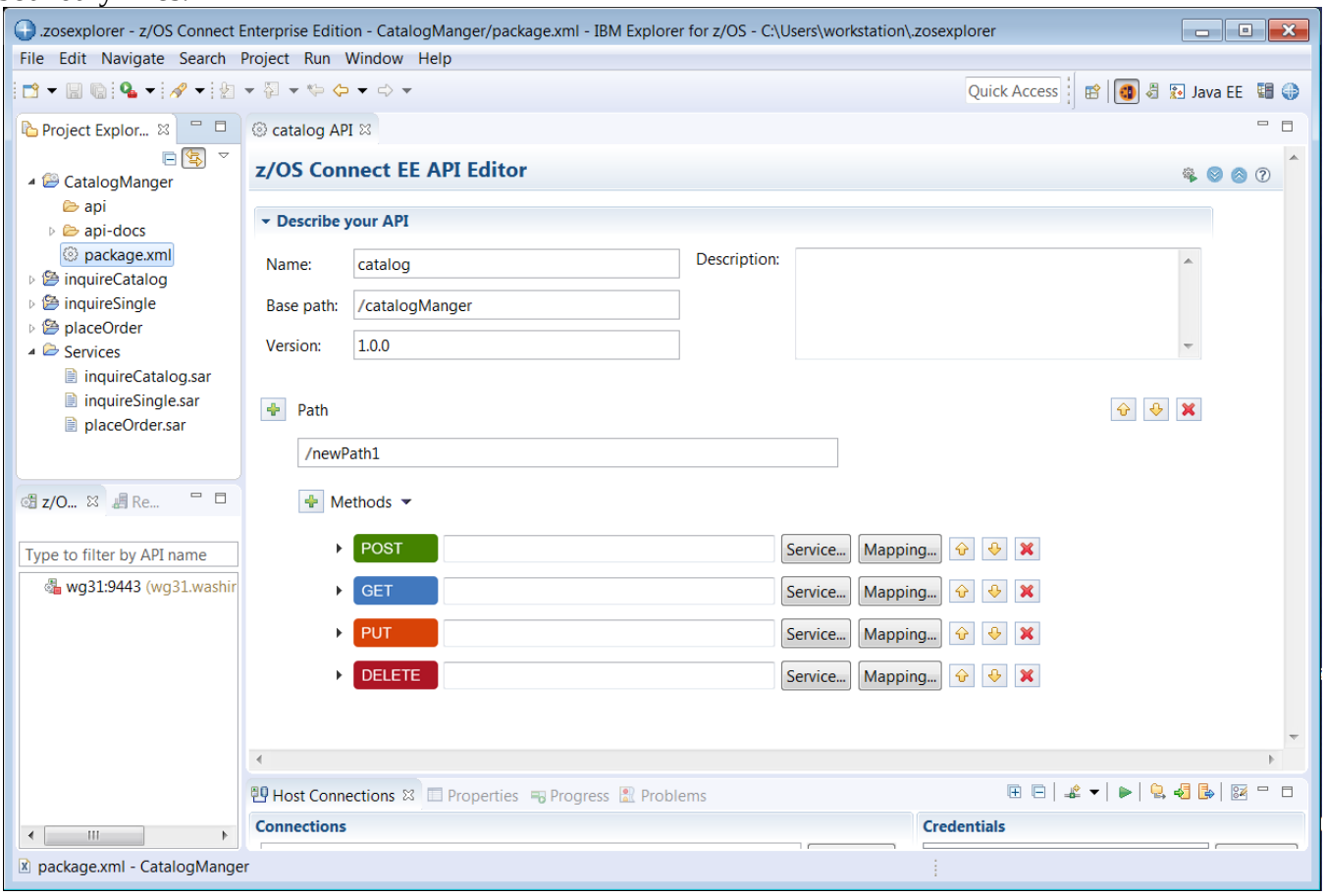
___2. Enter ***CatalogManager*** for the *Project name*, ***catalog*** for the *API name* and ***/catalogManager*** for the *Base path* value (case is important). Click **Finish** to continue.



. **Important:** The values are somewhat arbitrary, but they do relate to later tasks. For example, the API name, e.g. catalog must match the value of the *zosconnectAPI* name specified in the *catalog.xml* file (see Figure 1 on page 26) included earlier in *server.xml*. If you use the values and cases as supplied, then the use of subsequent URLs will work seamlessly.

Mitch Johnson (mitchj@us.ibm.com)

___13. You should now see something like below. The views may need to be adjusted by dragging the view boundary lines.



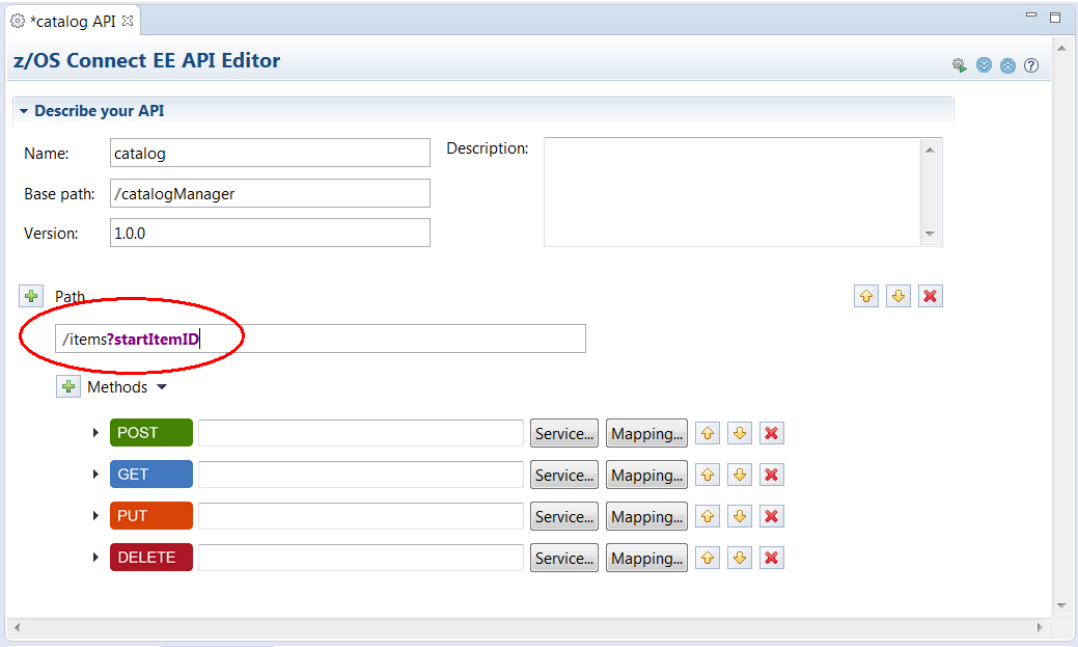
Summary

You created the basic framework for the API project in the API editor.

Mitch Johnson (mitchj@us.ibm.com)

Compose the API

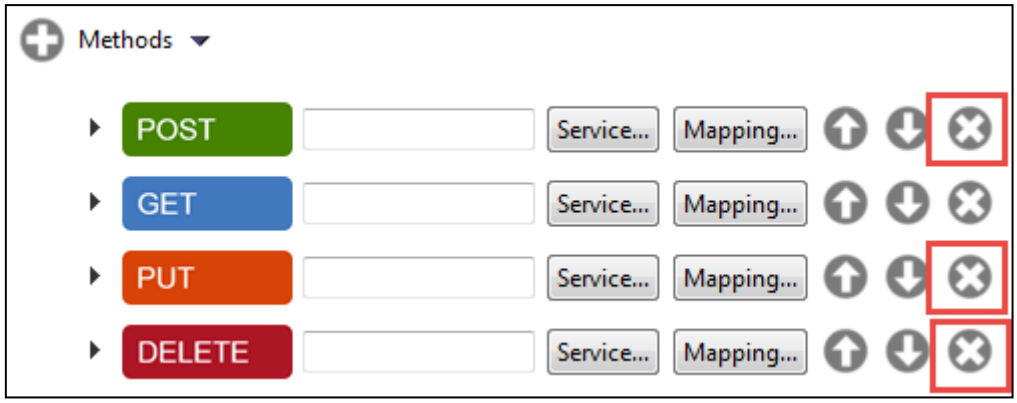
___1. To compose the *inquireCatalog* service start by entering a *Path* of ***/items?startItemID***.



Note: The */items* path element is somewhat arbitrary, but use this value so tasks later in the lab will work unchanged.

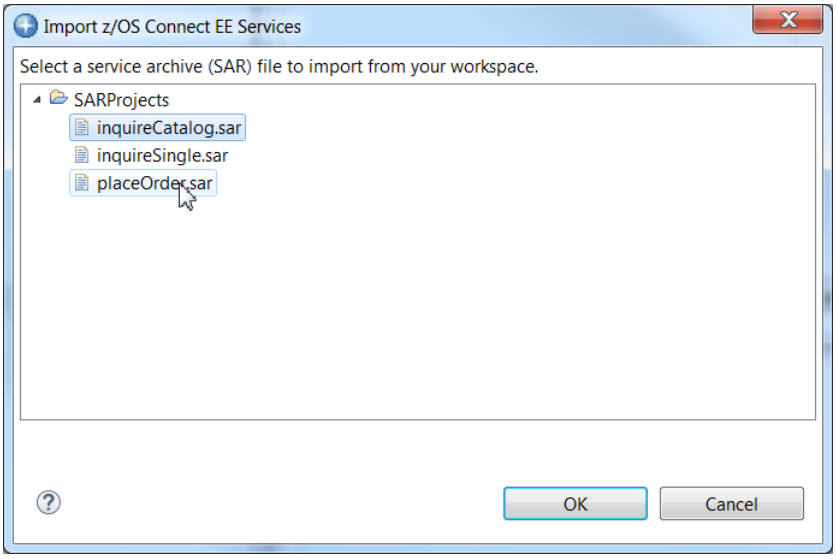
The *?startItemID* element tells the editor that value will be supplied as a path parameter in the URL.

___2. For the *inquireCatalog* service the HTTP method will be **GET**, so we do not need the **POST**, **PUT** and **DELETE** methods. Remove them by clicking the *X* icon to the right of each:

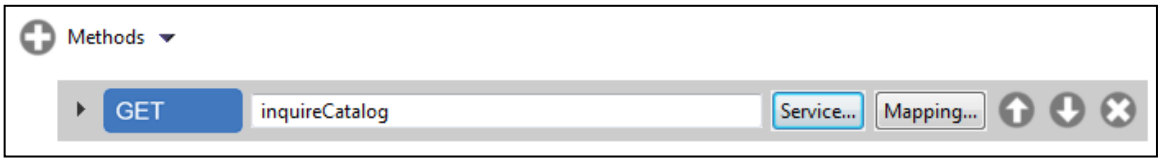


That should leave you with only the **GET** method.

___3. Next import the service required for this method of this *Path*. Click on the **Service** button to the right of the **GET** me*thod. On the Select a z/OS Connect EE Service window click the* **Workspace** button and expand the *SARProjects* folder on the next window. Select the *inquireCatalog* service from the list of service archive files and click **OK** three times



This will populate the field to the right of the method:

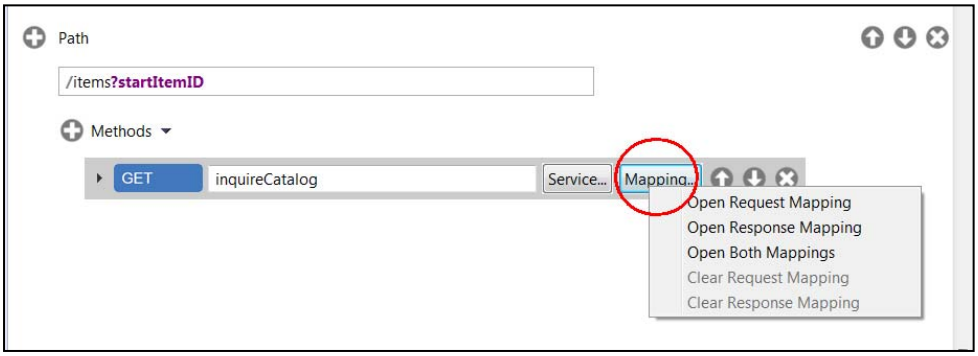


___4. Save the changes so far by using the key sequence **Ctrl-S.**

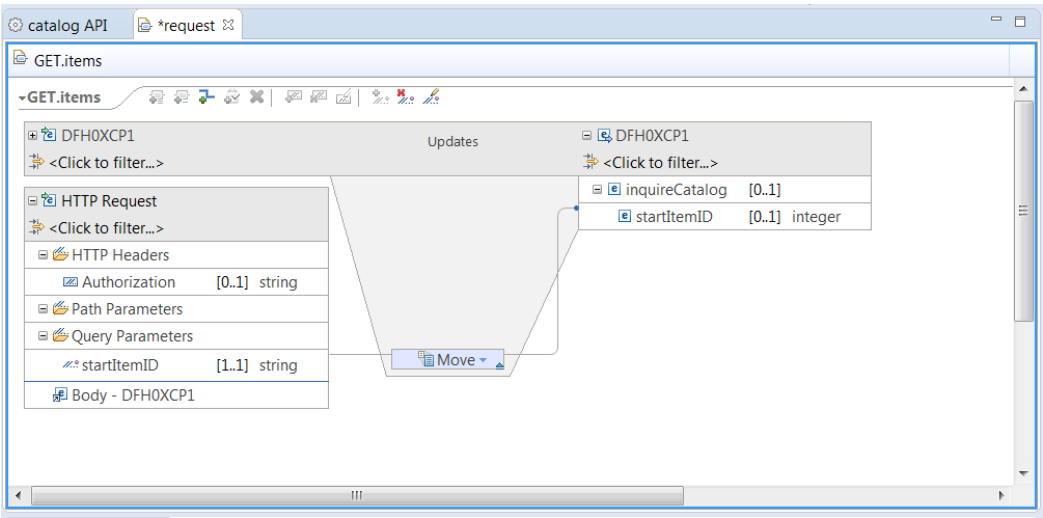
> **Tech-Tip: I**f any change is made in any edit view an asterisk (*) will appear before the name of the artifact in the view tab, e.g. **package.xml*. Changes can be saved at any time by using the **Ctrl-S** key sequence.

Mitch Johnson (mitchj@us.ibm.com)

___5. Next, click on the **Mapping** button, then select *Open Request Mapping*:



___6. Expand *inquireCatalog* on the right and next, on the left side hover select *startItemID* and without releasing create a connection between *startItemID* and field *startItemID* on the right hand side. The result is a line that maps the value of *startItemID* from the query parameter to the field *startItemID*. This means the value or contents of *startIemID* query parameter specified in a URL will be moved to the *startItemID* field.



What you have done is reduce the request to a simple GET URI with a single query parameter:

http://<*host*>:<*port*>/catalogManager/items?startItemID=10

The query parameter provides the program the *starting* item number for the listing of items.
Those fields not needed on a request are hidden from the REST client's view. Those fields that have the same static value every time (*ca_request_id*) are assigned a value of 01INQC.

___7. Save the changes using key sequence **Ctrl-S**.

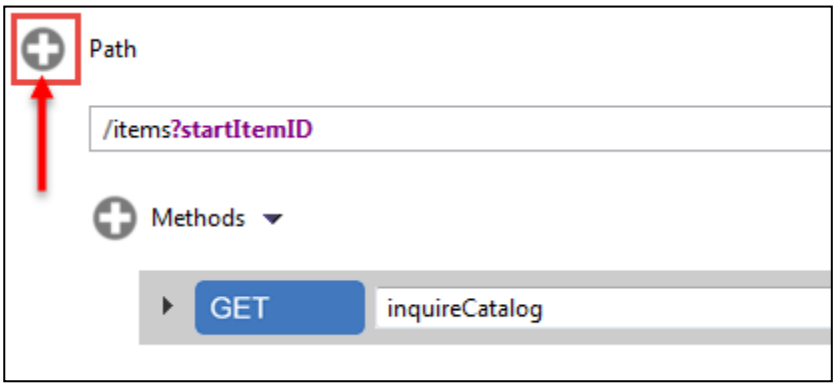
___8. Close the *request* mapping tab by clicking on the *white X* in the tab window.

___9. No changes are required for the response mapping.

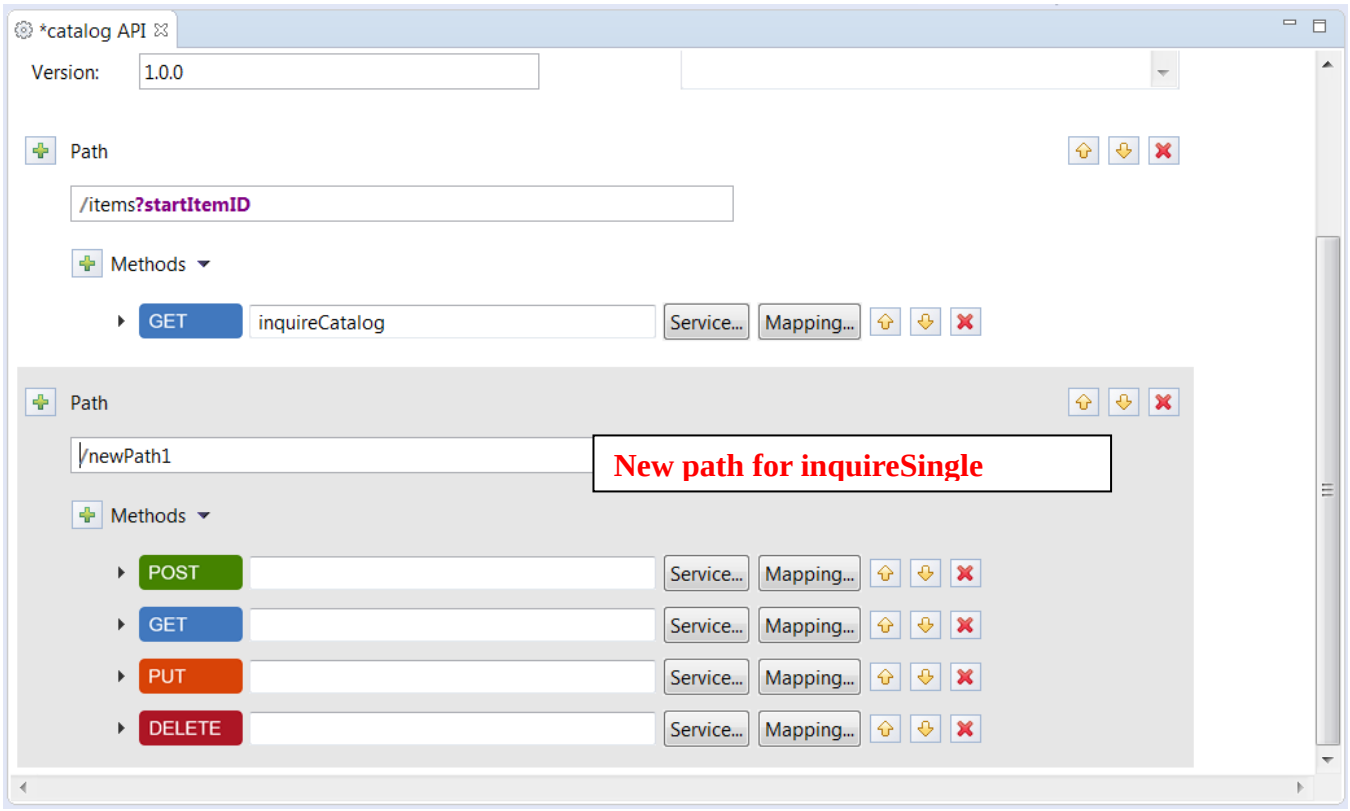
Mitch Johnson (mitchj@us.ibm.com)

You're done with the *inquireCatalog* portion of the API. Next up is *inquireSingle*. The steps are very similar. You will find you get better at this the more you do it.

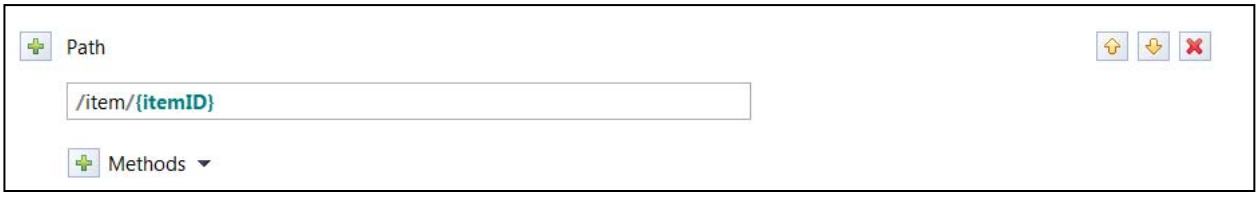
___10. Click the plus icon to add another path to the API.



The result is:



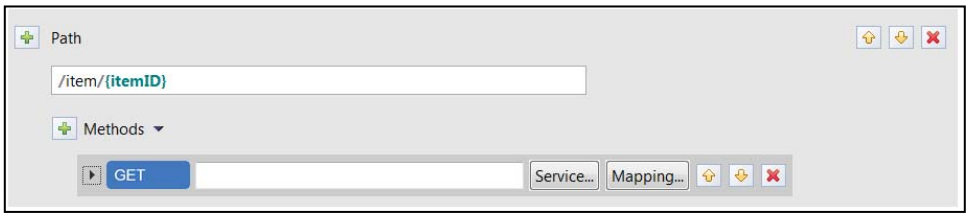
___11. Enter a path value of ***/item/{itemID}***:



The string *{itemID}* is recognized by the editor as a path parameter.

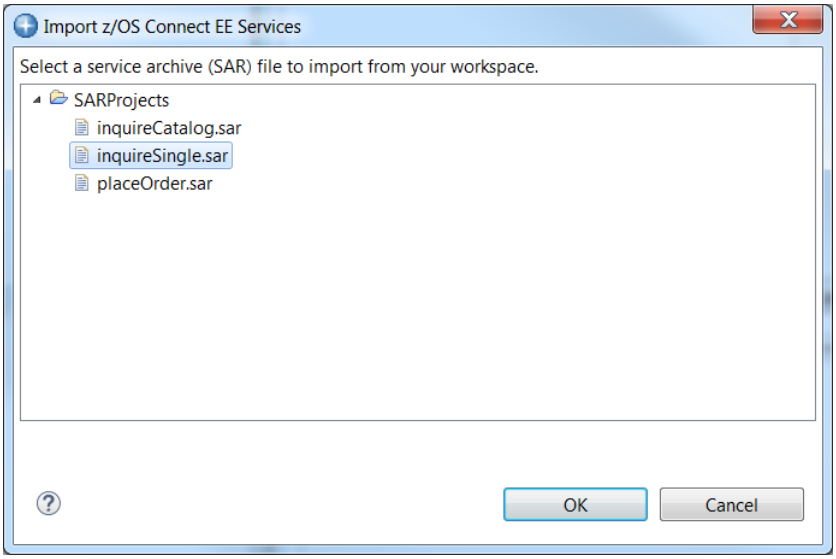
Tech-Tip: The {} characters in the above examples are braces {} not parenthesis ().

___12. Once again, the method we will use for this is **GET**, so we do not need **POST**, **PUT** or **DELETE.** Remove those by clicking the *X* to the right of each. The result should be:



Tech-Tip: Additional *Paths* can be added by clicking the + icon beside *Path and a*dditional *Methods* can be added by clicking the + icon beside *Methods*.

___14. Next import the service required for this method of this *Path*. Click on the **Service** button to the right of the **GET** m*ethod. On the Select a z/OS Connect EE Service window click the* **Workspace** *button and expand the SARProjects* folder on the next window. Select the *inquireSingle* service from the list of service archive files and click **OK** three times.



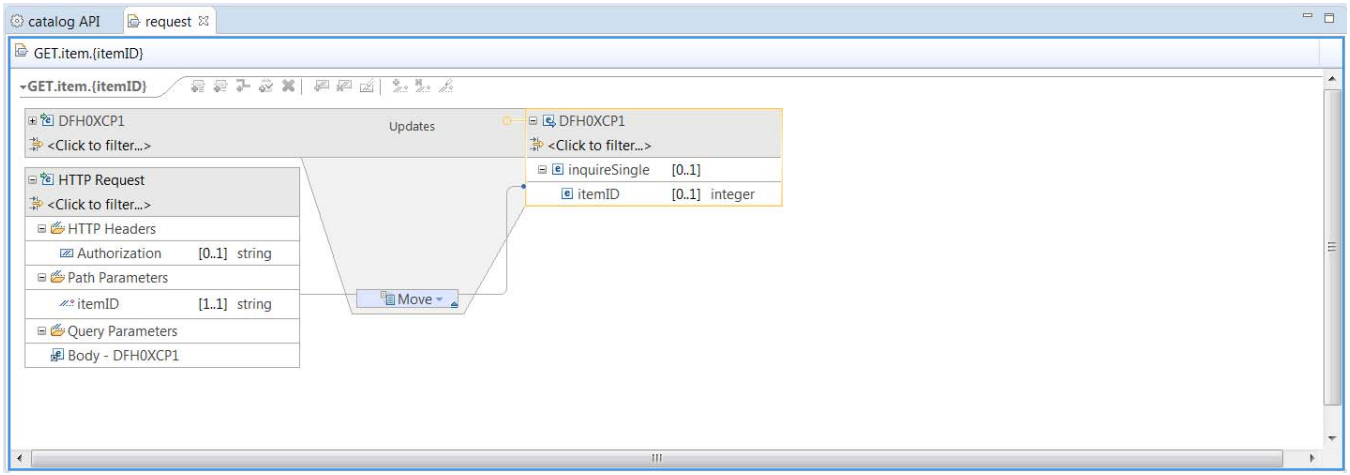
Mitch Johnson (mitchj@us.ibm.com)

___15. Save the changes by using the key sequence **Ctrl-S**.

___16. Click on *Mapping* → *Open request mapping.*

___17. Expand *inquireSingle* by clicking on the little + sign to the left of the field. Select *itemID* on the left hand side and drag it over to *itemID* on the right hand side to make a move connection so the value or contents of *itemID* are moved into *itemID* field.

___18. The picture below is the final result. Inspect your mapping and make sure:



What you have done is reduce the request to a simple GET URI with a single path parameter:

http://<host>:<port>/catalogManager/item/10

The path parameter provides the program the item number for the listing of that item..

___19. Save the the changes using the key sequence **Ctrl-S**.

___20. Close the *request* mapping tab.

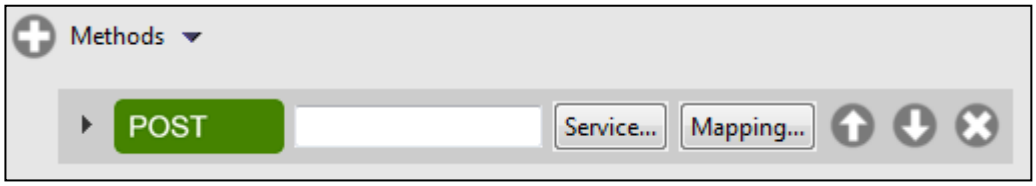
___21. No changes are required to the *response* mapping.

One more service to add – **orderItem**. The process is similar. The steps should be getting familiar to you by this time.

___22. Add another path by clicking on the + circle:

___23. In your new path field, provide a path value of ***/orders***:

___24. The HTTP method for this will be **POST**, so we can get ride of **GET**, **PUT** and **DELETE.** Remove those methods by clicking the **X**. You should then see:



___25. Import the service required for this method of this *Path*. Click on the **Service** button to the right of the **GET** m*ethod. On the Select a z/OS Connect EE Service window click the* **Workspace** button and expand the *SARProjects* folder on the next window. Select the *placeOrder* service from the list of service archive files and click **OK** three times.

___26. Save the changes by using the key sequence **Ctrl-S**.

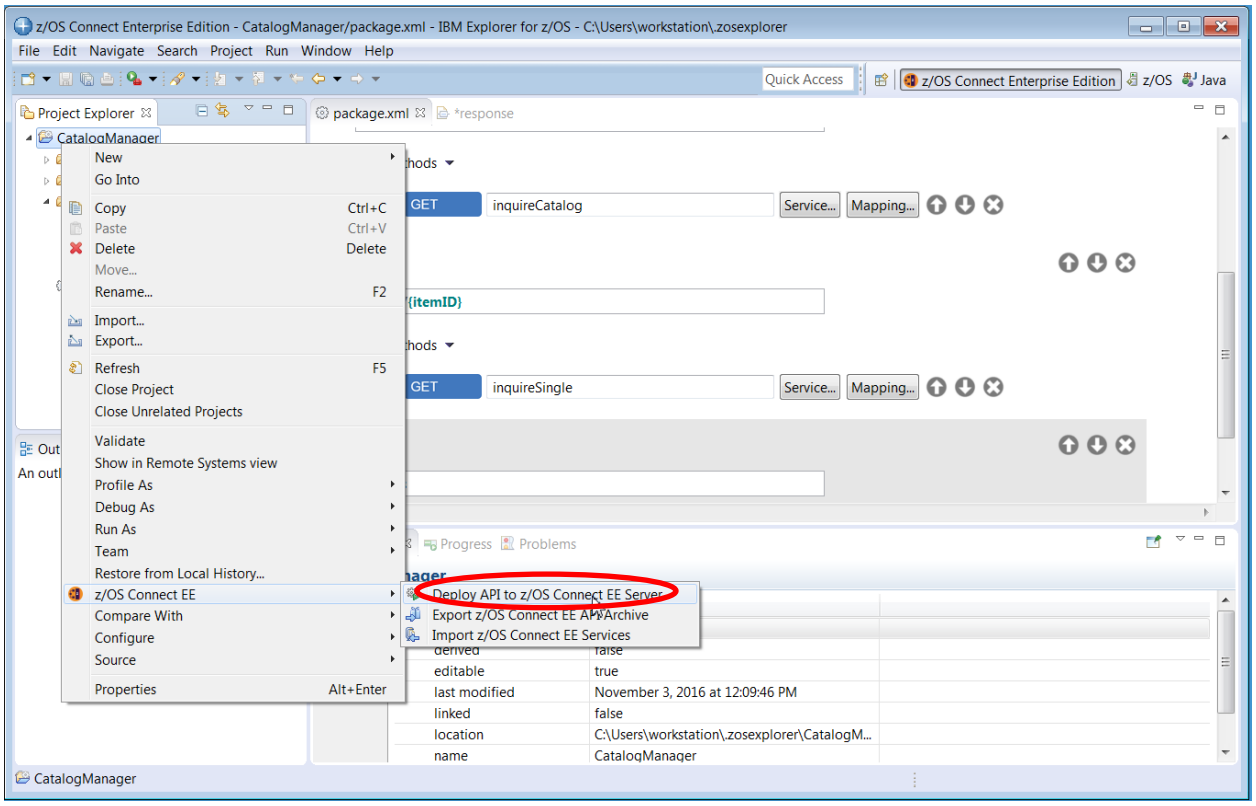
Summary

You created the API, which consists of three paths and the request and response mapping associated with each. That API will now be deployed into z/OS Connect EE V3.0. The REST interfaces are now simpler than the earlier coarse-grained services. The services are still used, but by z/OS Connect EE V3.0 as it processes the higher-level API requests.

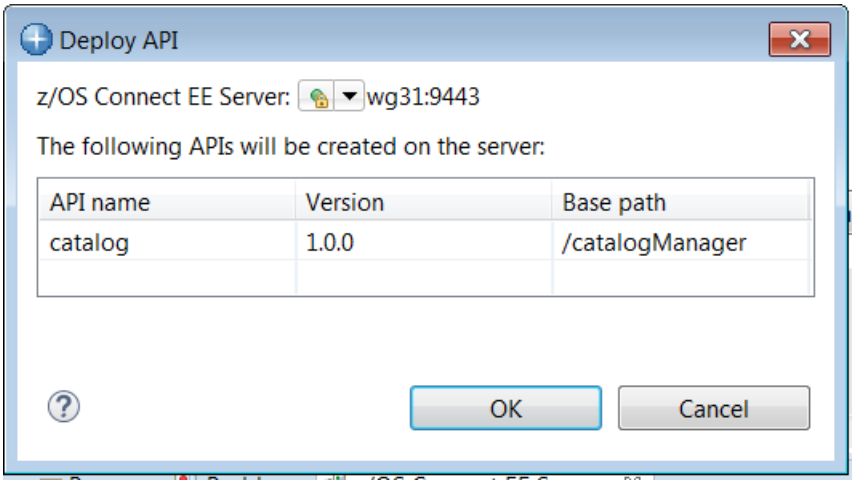
Mitch Johnson (mitchj@us.ibm.com)

Deploy the API to a z/OS Connect EE Server

___1. In the *Project Explorer* view (upper left), right-mouse click on the *CatalogManager* folder, then select *z/OS Connect EE→ Deploy API to z/OS Connect EE Server*.

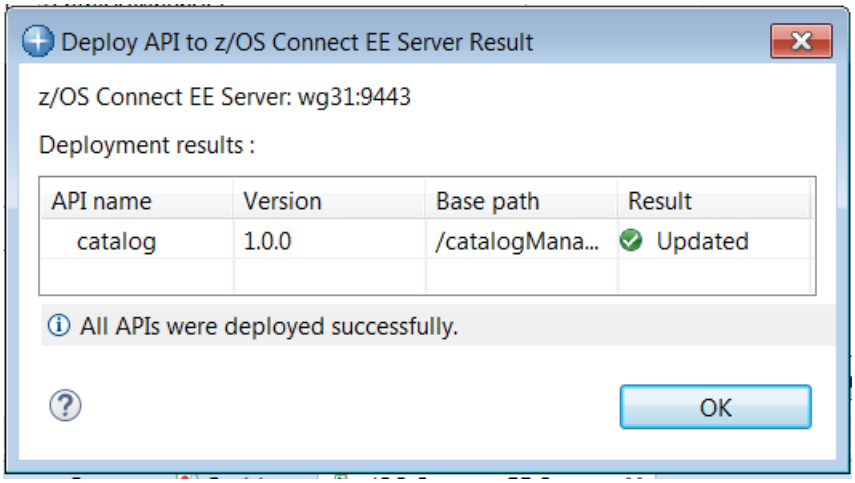


___2. Since z/OS Explorer is connected to only one z/OS Connect server there is only one choice (*wg31:9443*). If z/OS Explorer had multiple host connections to z/OS Connect servers then the pull down arrow would allow a selection to which server to deploy. Click **OK** on this screen to continue.



Mitch Johnson (mitchj@us.ibm.com)

___3. The API artifacts will be transferred to z/OS and copied into the */var/zosconnect/servers/myServer/resources/zosconnect/apis* directory.



Summary

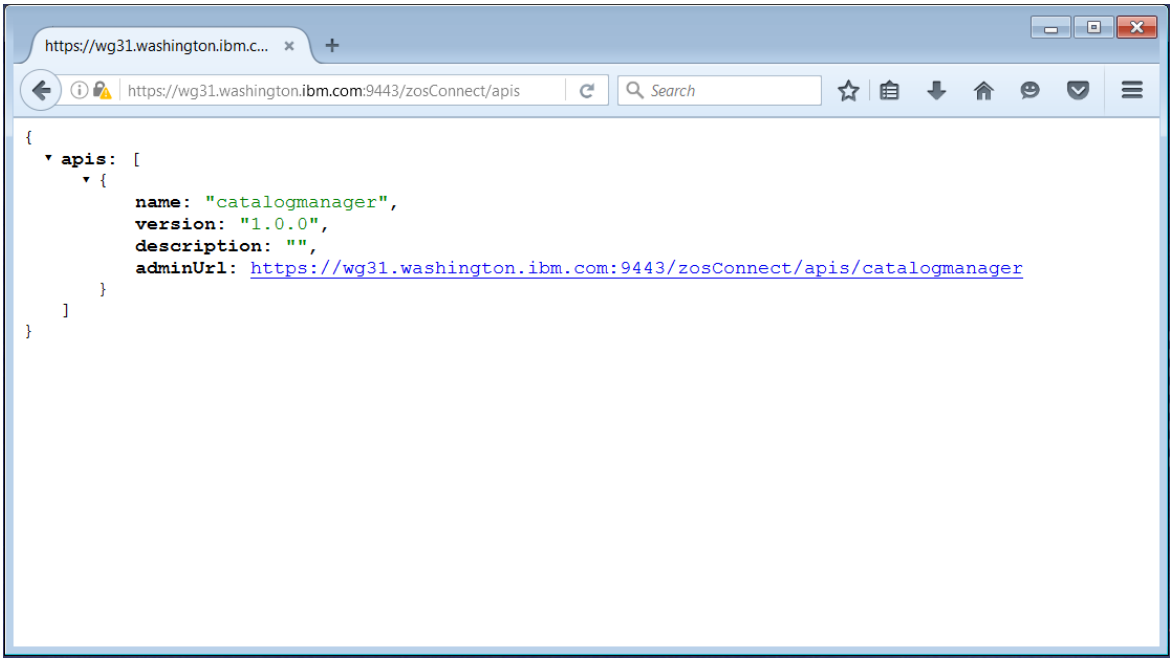
The API was contained in the AAR file, which you uploaded to z/OS and deployed. The update of the server.xml told z/OS Connect EE V3.0 about the presence of the API.

Mitch Johnson (mitchj@us.ibm.com)

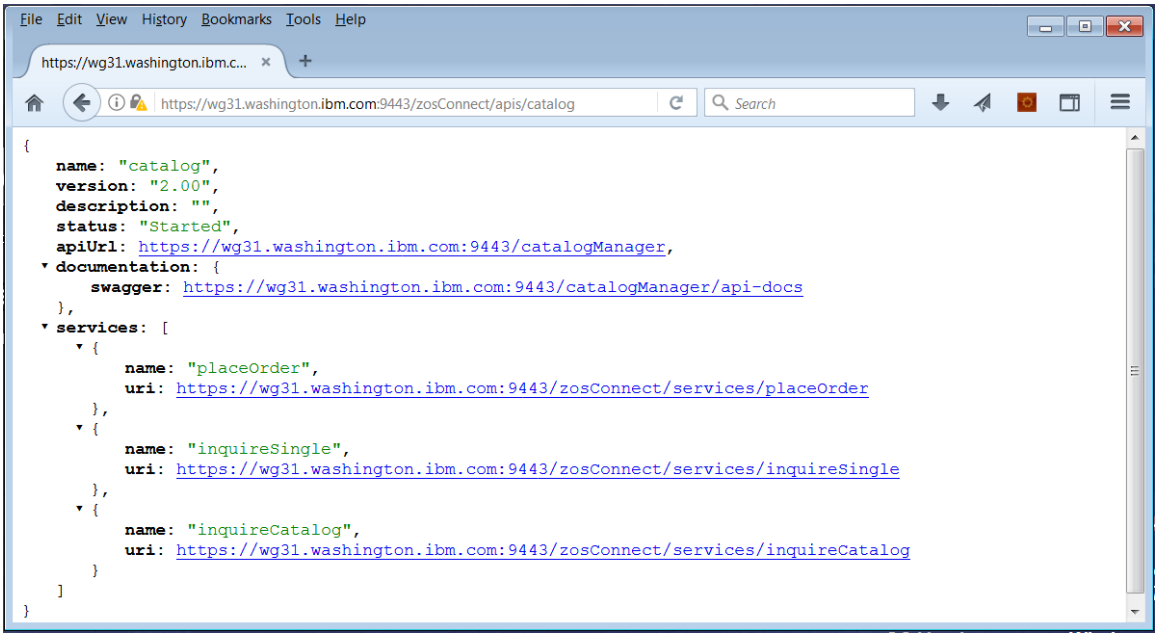
Test the API

Important: The Swagger UI in the API Toolkit has been configured to use an external browser. This simplifies the management of digital certificates.

___1. Next enter URL ***https://wg31.washington.ibm.com:9443/zosConnect/apis*** in the Firefox browser and you should see the window below. The *catalog* API now shows as being available.

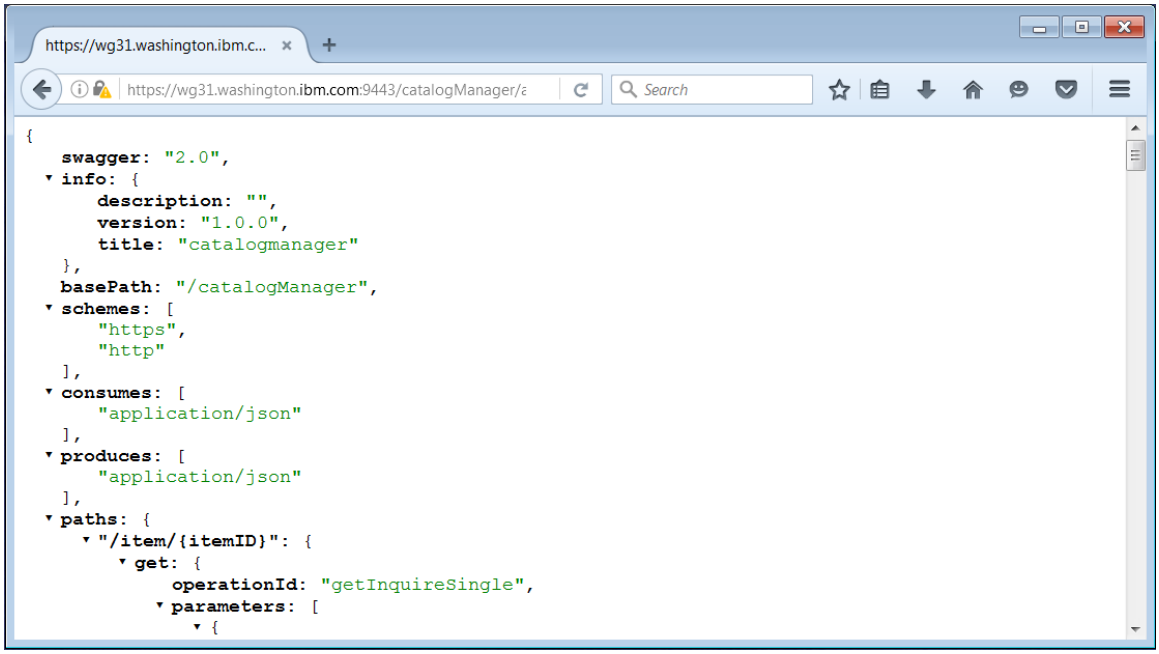


___2. If you click on *adminUrl* URL the window below should be displayed:



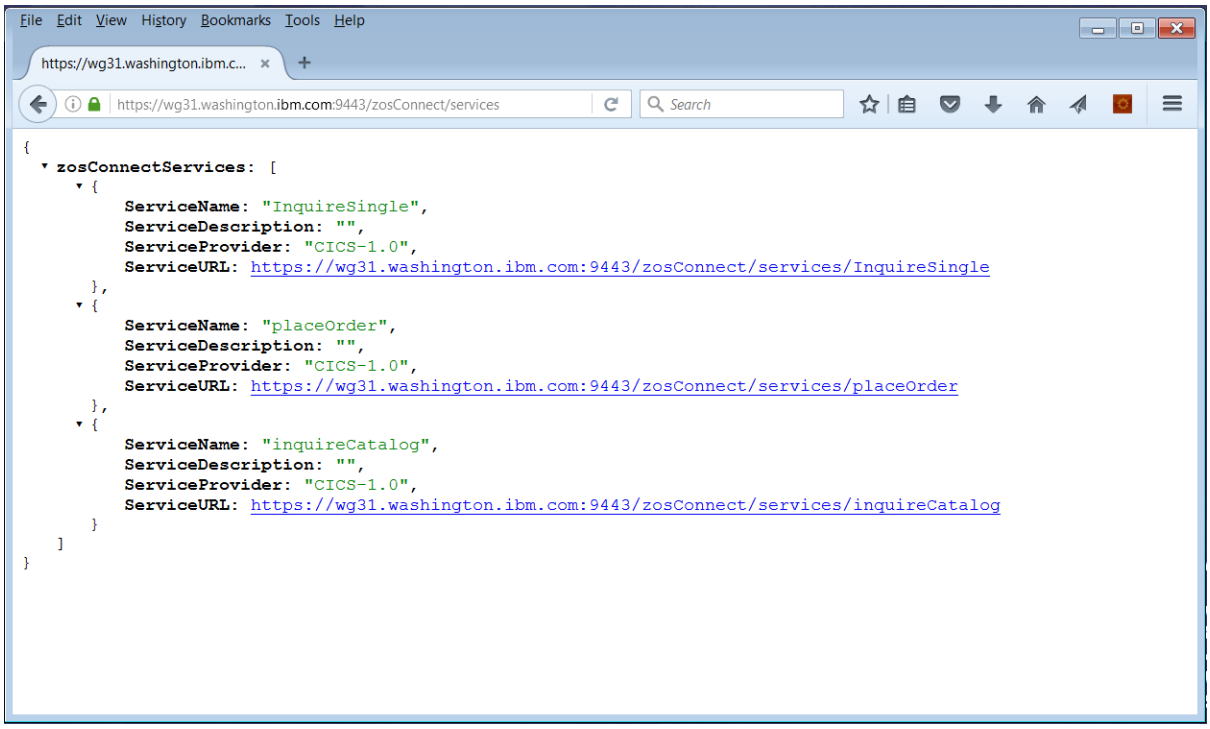
Mitch Johnson (mitchj@us.ibm.com)

___3. Next click on the *swagger* URL and you should see the Swagger document associated with this API.



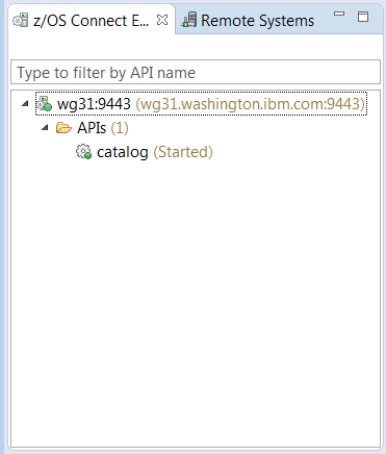
Explore this Swagger document and you will see the results of the request and response mapping performed earlier. This Swagger document can be used by a developer or other tooling to develop REST clients for this specific API.

___4. Next enter URL ***https://wg31.washington.ibm.com:9443/zosConnect/services*** in the Firefox browser and you should see the window below. The three services are now available.



Mitch Johnson (mitchj@us.ibm.com)

___5. In the lower left hand side of the *z/OS Connect Explorer* perspective there is view entitled *z/OS Connect EE Servers*. Expand *wg31:9453* and the expand the *APIs* folder. You should see a list of the APIs installed in the server.

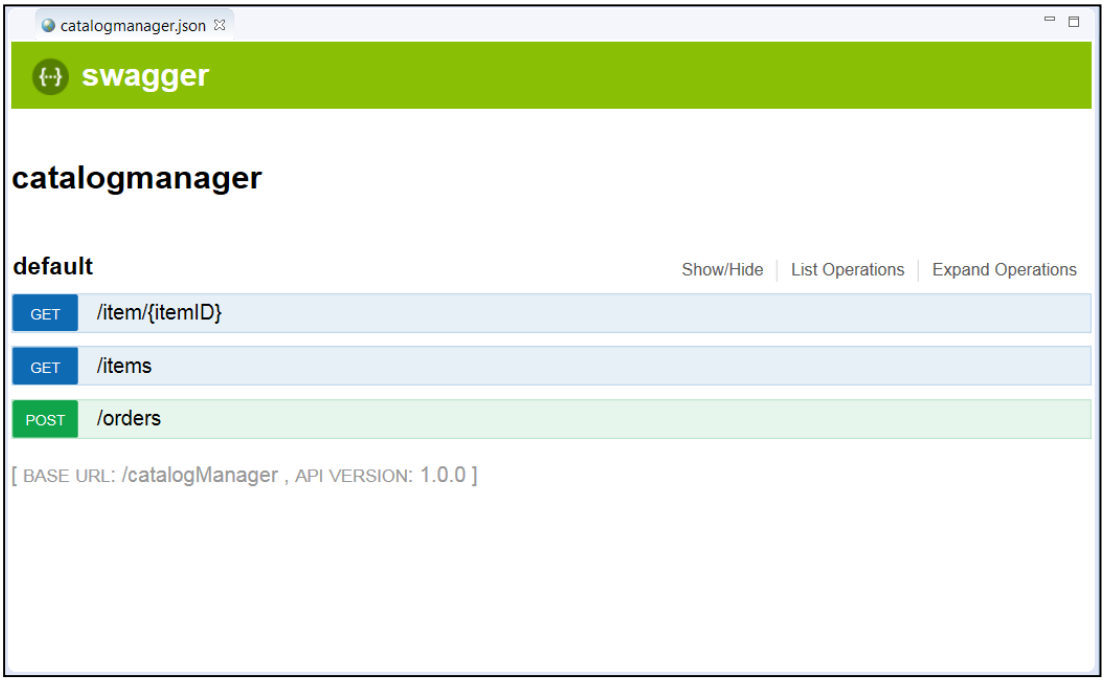


___6. Right mouse button click on *catalog* and select *Open in Swagger UI*. Click **OK** if an informational prompt about certificates appears (see Tech Tip for explanation). This will open a new view showing a *Swagger* test client (see below).

Tech Tip: Swagger UI will not try to automatically download the self-signed certificate from the z/OS Connect EE server. The server's self-signed certificate must be install using other means. In this exercise the self-signed certificate was installed earlier when displaying the APIs or services installed in the z/OS Connect EE server. Note that this must be done for every server that will be accessed using the Swagger UI.

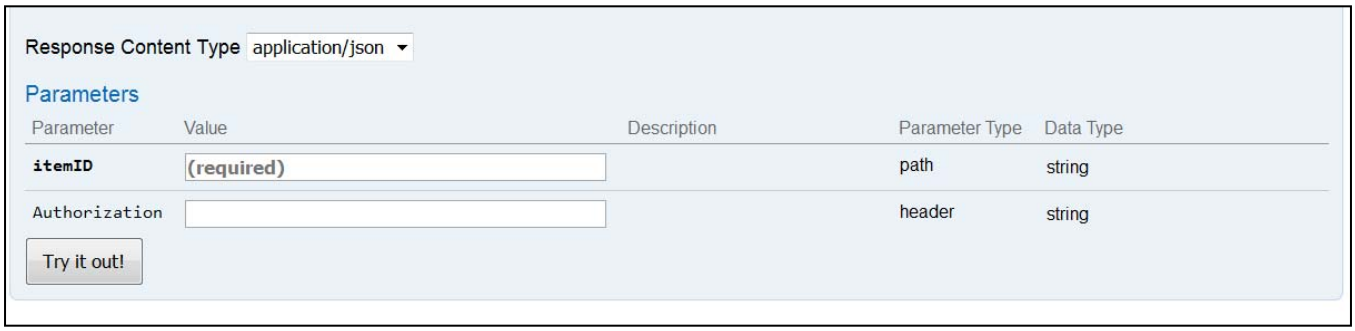
Mitch Johnson (mitchj@us.ibm.com)

___7. Click on *List Operations* option in this view and this will display a list of available HTTP methods in this API.



Mitch Johnson (mitchj@us.ibm.com)

___8. Select the method for selecting a single item from the catalog by item number table by clicking on the *GET* box. This action will expand this method in this view and provides a Swagger UI test client (you may have to use the slider bar and adjust the perspective to see the entire client).

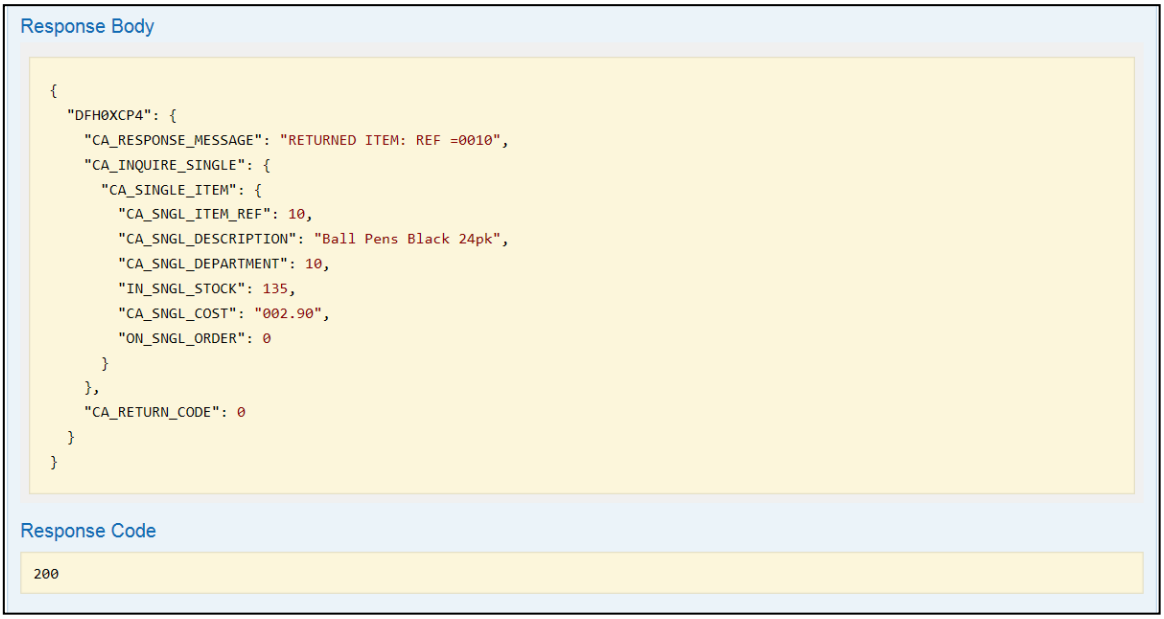


___9. Enter ***10*** in the box beside *itemID* and ***Basic RnJlZDpmcmVkcHdk*** in the area beside *Authorization* and press the **Try it out!** button. You may see a Security Alert pop-up warning about the self-signed certificate being used by the z/OS Connect EE server. Click **Yes** on this pop-up.

> **Tech Tip:** Since the z/OS Connect EE server require security the Swagger UI must include the authorization string in the Request Header. The format of this authorization string is the type of authorization (e.g. *Basic*) and the base 64 encoded string for *Fred:fredpwd ,e.g RnJlZDpmcmVkcHdk*, see URL https://www.base64encode.org. Be sure there are no extra spaces at the end of the string.

Mitch Johnson (mitchj@us.ibm.com)

___10. Scroll down the view and you should see the *Response Body* which contains the results of the GET method (see below). Note that the request field removed from the interface in an earlier steps is not present.



Mitch Johnson (mitchj@us.ibm.com)

___11. Repeat this process with other items and observe the results. For example using a value of ***30*** would return.

Response Body

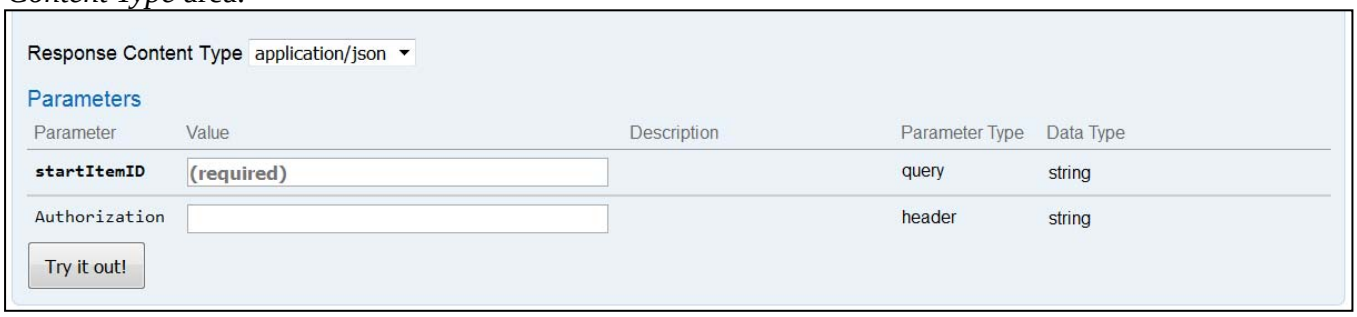

```
{
  "DFH0XCP4": {
    "CA_RESPONSE_MESSAGE": "RETURNED ITEM: REF =0030",
    "CA_INQUIRE_SINGLE": {
      "CA_SINGLE_ITEM": {
        "CA_SNGL_ITEM_REF": 30,
        "CA_SNGL_DESCRIPTION": "Ball Pens Red 24pk",
        "CA_SNGL_DEPARTMENT": 10,
        "IN_SNGL_STOCK": 105,
        "CA_SNGL_COST": "002.90",
        "ON_SNGL_ORDER": 0
      }
    },
    "CA_RETURN_CODE": 0
  }
}
```


The available catalog items are listed below.

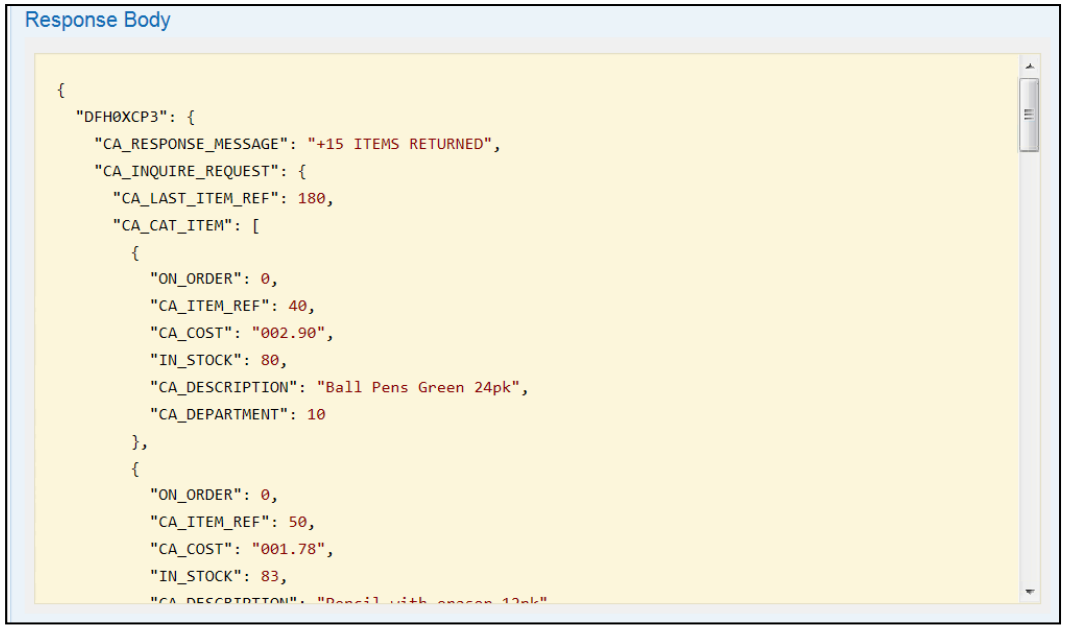
Item#	Description	Dept	Cost	In Stock	On Order
0010	Ball Pens Black 24pk	010	002.90	0135	000
0020	Ball Pens Blue 24pk	010	002.90	0006	050
0030	Ball Pens Red 24pk	010	002.90	0106	000
0040	Ball Pens Green 24pk	010	002.90	0080	000
0050	Pencil with eraser 12pk	010	001.78	0083	000
0060	Highlighters Assorted 5pk	010	003.89	0013	040
0070	Laser Paper 28-lb 108 Bright 500/ream	010	007.44	0102	020
0080	Laser Paper 28-lb 108 Bright 2500/case	010	033.54	0025	000
0090	Blue Laser Paper 20lb 500/ream	010	005.35	0022	000
0100	Green Laser Paper 20lb 500/ream	010	005.35	0003	020
0110	IBM Network Printer 24 - Toner cart	010	169.56	0012	000
0120	Standard Diary: Week to view 8 1/4x5 3/4	010	025.99	0007	000
0130	Wall Planner: Eraseable 36x24	010	018.85	0003	000
0140	70 Sheet Hard Back wire bound notepad	010	005.89	0084	000
0150	Sticky Notes 3x3 Assorted Colors 5pk	010	005.35	0036	045
0160	Sticky Notes 3x3 Assorted Colors 10pk	010	009.75	0067	030
0170	Sticky Notes 3x6 Assorted Colors 5pk	010	007.55	0064	030
0180	Highlighters Yellow 5pk	010	003.49	0088	010
0190	Highlighters Blue 5pk	010	003.49	0076	020
0200	12 inch clear rule 5pk	010	002.12	0014	010
0210	Clear sticky tape 5pk	010	004.27	0073	000

___12. Collapse the Swagger UI test area for the **GET** single item method for the clicking on the blue *GET* box beside */item/{itemID}*.

___13. Click on the blue *GET* box beside */items* to access the **GET** method to open its Swagger Test user interface for the inquire catalog service (e.g. *inquireCatalog*) and scroll down to the *Response Content Type* area.

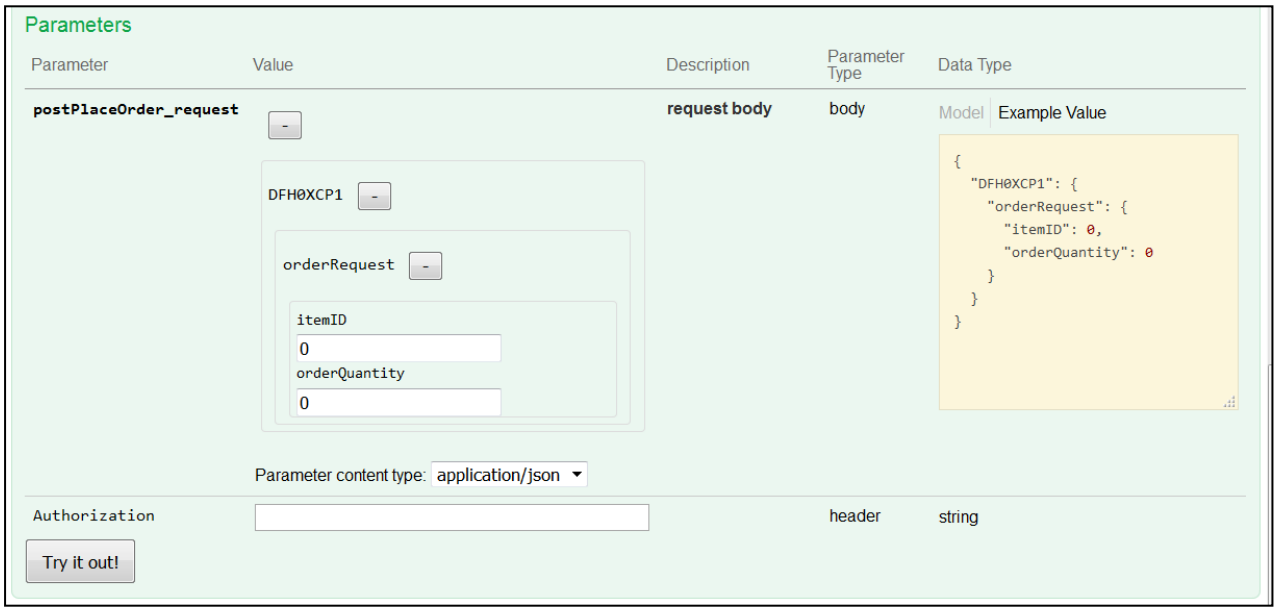


___14. Enter ***40*** in the box beside *startItemID* and ***Basic RnJlZDpmcmVkcHdk*** in the area beside *Authorization* and press the **Try it out!** button. Scroll down and you should see the following information in the *Response Body*.



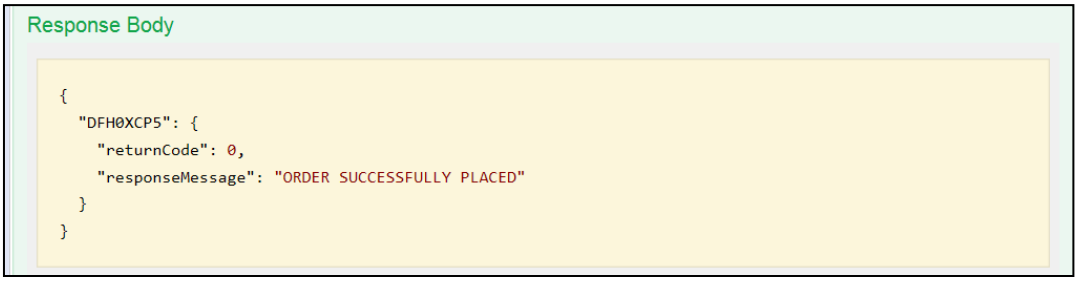
Try a few other values for *startItemID* and compare the resultse.

___15. Click on the green POST box to access the **POST** method to open its Swagger Test user interface for the place order service (e.g. *placeOrder*) and scroll down to the *Response Content Type* area.



___16. Enter ***40*** in the area below *itemID* and ***1*** in the area under *orderQuantity* and ***Basic RnJlZDpmcmVkcHdk*** in the area beside *Authorization* and press the **Try it out!** button

Scroll down and you should see the following information in the *Response Body*.



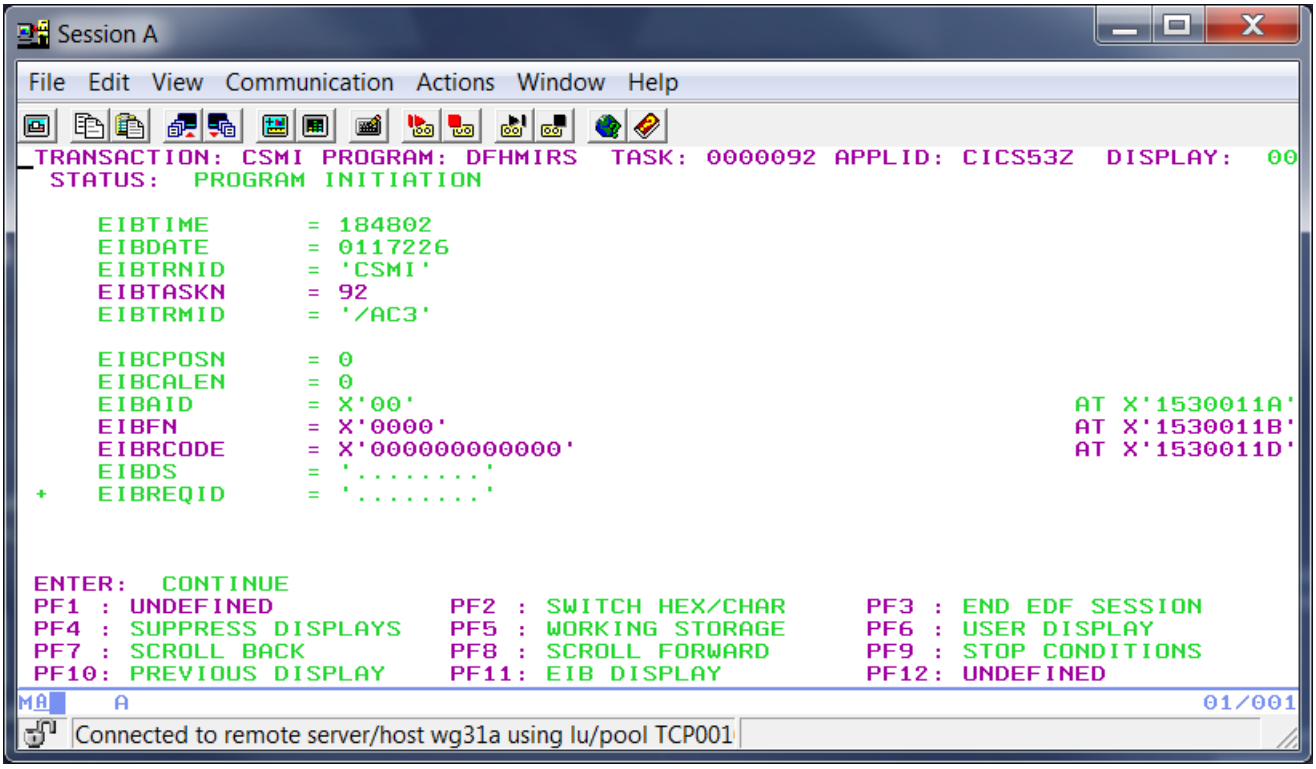
Summary

You have verified the API. The API layer operates above the service layer you defined and tested earlier. The API layer provides a further level of abstraction and allows a more flexible use of HTTP verbs, and better mapping of data via the API editor function.

Mitch Johnson (mitchj@us.ibm.com)

Optional

If you are familiar with CICS Execution Diagnostic Facility (EDF) start a 3270-terminal session with CICS, clear the screen and enter CICS transaction ***CEDX CSMI***. When you repeat of any of the above test you should be able to trace the flow of the request through CICS.



If you were running in a CICS embedded z/OS Connect EE server, you would trace transaction CJSA.

Summary

You have verified the API. The API layer operates above the service layer you defined. The API layer provides a further level of abstraction and allows a more flexible use of HTTP verbs, and better mapping of data via the API editor function.

Mitch Johnson (mitchj@us.ibm.com)

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 lab we'll push some things down to SAF (RACF in the case of this workshop).

Using SAF for registry and access role checking

___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. Browse data set *USER1.ZCEE30.CNTL*. You should see the members in that data set.

___3. Next browse member **ZC3RACF3**, you should see the RACF commands below.

```
//ZC2RACF3 JOB (0),'LIBERTY RACF',CLASS=A,REGION=0M,
//             MSGCLASS=H,NOTIFY=&SYSUID
//*-------------------------------------------------------------*/
//*   DEFINE USER PROFILES FOR LIBERTY SERVERS                  */
//*-------------------------------------------------------------*/
//RACF     EXEC PGM=IKJEFT01,REGION=0M
//SYSTSPRT DD SYSOUT=*
//SYSTSIN  DD *
ADDGROUP WSGUESTG OMVS(AUTOGID) OWNER(SYS1)
ADDUSER WSGUEST RESTRICTED DFLTGRP(WSGUESTG) OMVS(AUTOUID -
  HOME(/u/wsguest) PROGRAM(/bin/sh)) NAME('UNAUTHENTICATED USER') -
  NOPASSWORD NOOIDCARD
ADDUSER FRED DFLTGRP(LIBGRP) OMVS(AUTOUID HOME(/u/fred/) -
  PROGRAM(/bin/sh)) NAME('USER FRED')
ALTUSER FRED PASSWORD(FRED) NOEXPIRED
RDEFINE APPL BBGZDFLT UACC(NONE) OWNER(SYS1)
PERMIT BBGZDFLT CLASS(APPL) RESET
PERMIT BBGZDFLT CLASS(APPL) ACCESS(READ) ID(WSGUEST)
RALT APPL BBGZDFLT UACC(READ)
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(FRED,USER1) ACCESS(READ)
SETR RACLIST(EJBROLE) REFRESH
/*
```

What? In summary, that's creating an ID to be used when someone is unauthenticated, then Fred's ID and password, then it creates the EJROLE and Fred and USER1 are given access. This will take the place of the equivalent security definitions in the server.xml file.

Tech Tip: The value *BBGZDFLT* in the above commands matches the value of attribute *profileprefix* in the *saf.xml* file described on the next page.

___4. Submit the job for executions by entering ISPF command ***SUBMIT*** on the command line. The job should complete with a return code of zero.

Mitch Johnson (mitchj@us.ibm.com)

___8. Go to the ISPF Edit Entry Panel (option *2)* by entering ISPF command ***=2*** on the command line and pressing **Enter**.

___9. Enter ***/wasetc/zc3lab*** into the area beside *Name* under *Other Partitioned, Sequential or VSAM Data Set, or z/OS UNIX file:* and press **Enter**.

___10. Use the ***EA*** (Edit ASCII) line command to open the *saf.xml* file in EBCDIC 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

Notes

- The *zosSecurity-1.0* feature adds the z/OS security feature
- The *safRegistry, safAuthorization* and *safCredentials* elements enable authentication and authorization using RACF.

___11. Repeat these steps to access ***/var/zosconnect/servers/myServer*** and edit *server.xml*.

___12. Change the *include* statement for *basic.xml* to an *include* for *saf.xml*.

<include location="/wasetc/zc3lab/saf.xml" optional="true"/>

___13. Add a new include statement for *keystore.xml*.

<include location="/wasetc/zc3lab/keystore.xml" optional="true"/>

```
<?xml version="1.0" encoding="UTF-8"?>
<server description="new server">
<include location="/wasetc/zc3lab/ipic.xml" optional="true"/>
<include location="/wasetc/zc3lab/saf.xml" optional="true"/>
<include location="/wasetc/zc3lab/keystore.xml" optional="true"/>
```

___14. Enter the following MVS command to refresh the z/OS Connect EE V3.0 server configuration
F BAQSTRT,ZCON,REFRESH

___15. Close all instances of the Firefox browser (we want to force another prompt for ID, and closing the browser clears the cookies from before).

___16. Start Firefox and enter the following URL:

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

Mitch Johnson (mitchj@us.ibm.com)

___17. Next you will be prompted for a userid and password. Enter Enter **Fred** and **fredpwd** as you have before.

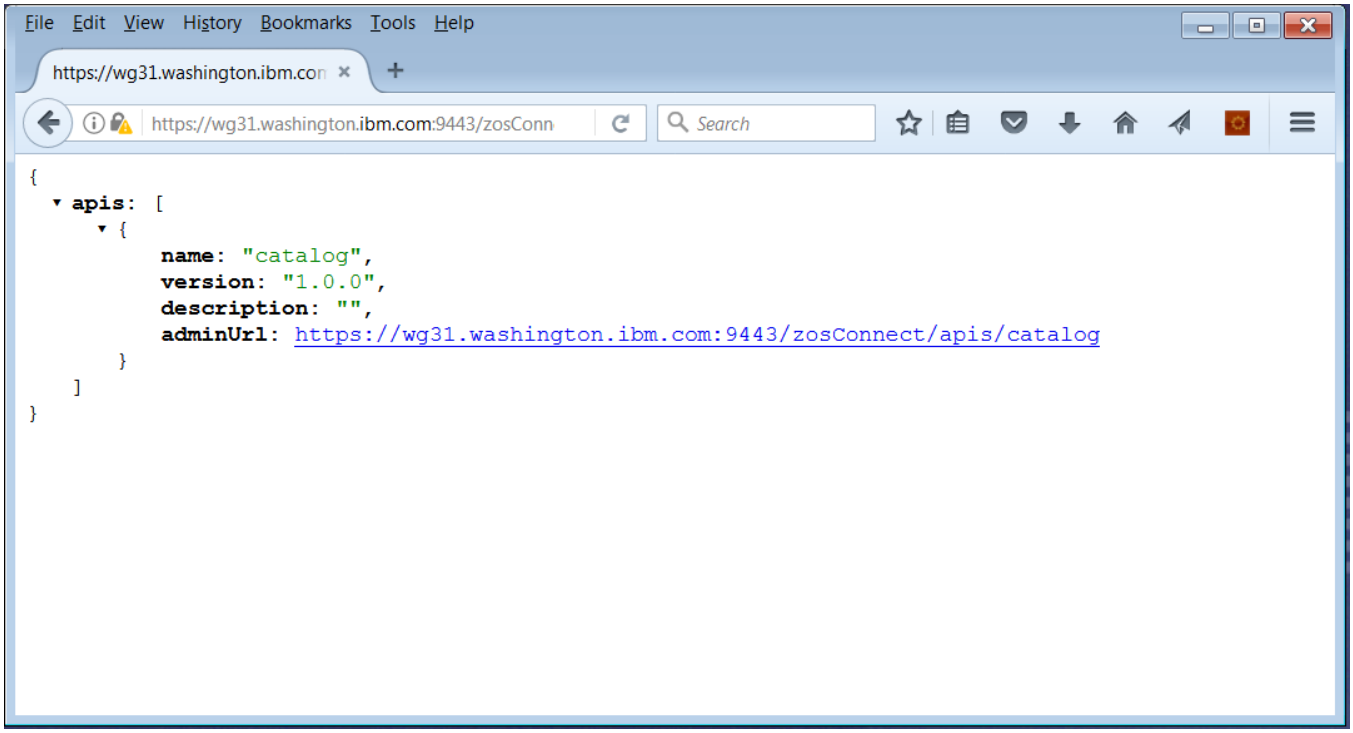
It should fail !

___17. In the userid/password prompt, enter ***Fred*** and ***FRED***.

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, and there Fred's password is FRED.

___18. 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.

___19. You should see a list of the APIs:



___20. 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.

___21. The request should fail with message *Error 403: AuthorizationFailed*. 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 EE 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   )
```


Mitch Johnson (mitchj@us.ibm.com)

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.

Using SAF for controlling access

___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)***

___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/group.xml** and you should see the XML elements 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 *group.xml* to the list of include files in the *server.xml*.

<include location="/wasetc/zc3lab/group.xml" optional="true"/>

```
<?xml version="1.0" encoding="UTF-8"?>
<server description="new server">
<include location="/wasetc/zc3lab/ipic.xml" optional="true"/>
<include location="/wasetc/zc3lab/saf.xml" optional="true"/>
<include location="/wasetc/zc3lab/keystore.xml" optional="true"/>
<include location="/wasetc/zc3lab/group.xml" optional="true"/>
```

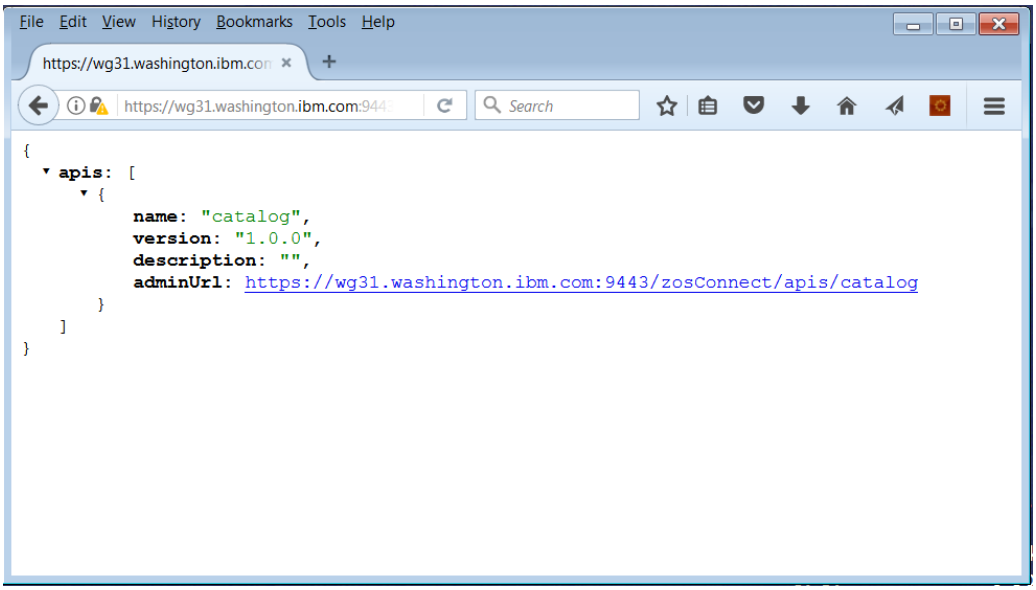
___7. Use SDSF to start your server with MVS command ***/S BAQSTRT***. You should see your server start:

___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

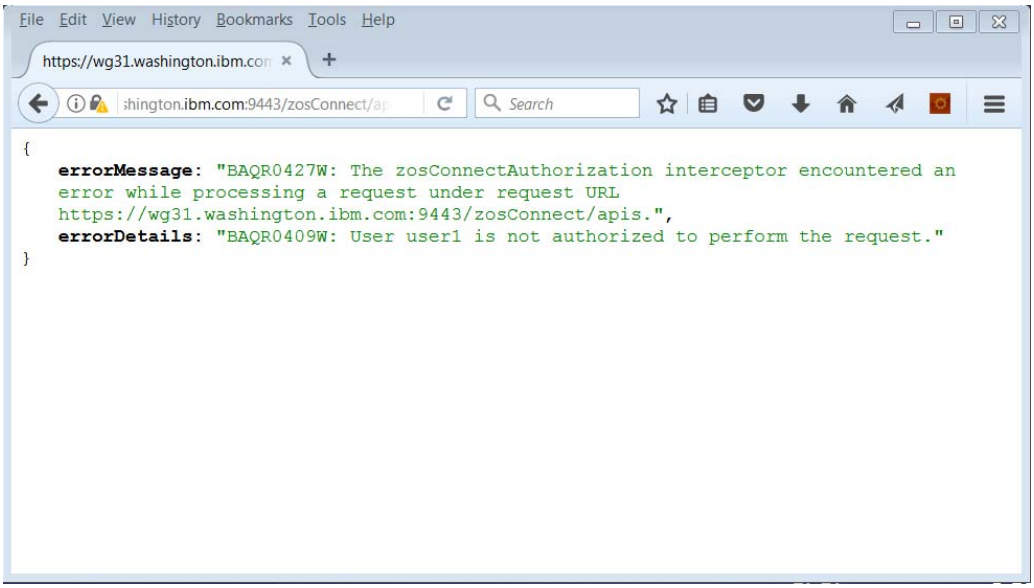
___10. On the *Authentication Required* popup window enter, enter ***Fred*** and ***FRED***. You should see:



FRED in in the administrators group and has the authority perform this function.

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

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



Next try to invoke an API.

___20. Use the *Command Prompt* icon on the desktop to open a DOS command prompt session.

___21. In the session use the change directory (cd) command to go to directory c:\z\CICSLab, e.g.
cd c:\z\CICSLAB

___22. Paste the command below at the command prompt and press **Enter**.

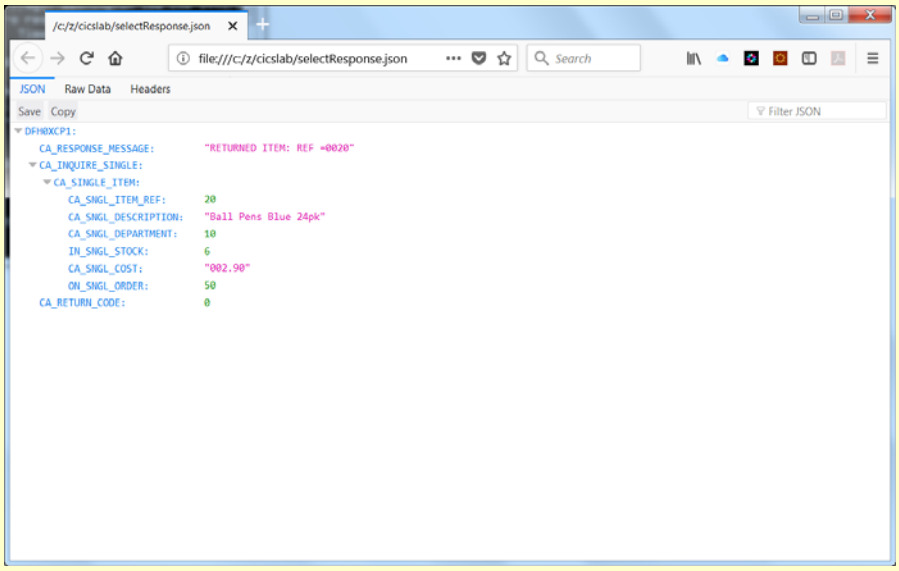
curl -X POST --user USER1:USER1 --header "Content-Type: application/json" -d @inquireSingle.json --insecure https://wg31.washington.ibm.com:9443/zosConnect/services/inquireSingle?action=invoke

___23. 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\cicslab\selectResponse.json*** you will see a browser session open with the JSON response formatted as below:



___24. 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

___25. 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.

Using SAF for SSL and key store management

Now let's go through the steps to use RACF (SAF) for SSL management.

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

___2. Go to data set *USER1.ZCEE30.CNTL* data set and browse the **ZC3RACF4** member. You will see something like this:

```
RACDCERT CERTAUTH GENCERT SUBJECTSDN(CN('CA for Liberty') -
  OU('LIBERTY')) WITHLABEL('LibertyCA.LIBERTY') TRUST -
  SIZE(2048) NOTAFTER(DATE(2018/12/31))
RACDCERT CERTAUTH EXPORT(LABEL('LibertyCA.LIBERTY')) -
  DSN('USER1.CERTAUTH.CRT') FORMAT(CERTDER)
RACDCERT ID(LIBSERV) GENCERT SUBJECTSDN(CN('wg31.washington.ibm.com') -
  O('IBM') OU('LIBERTY')) WITHLABEL('DefaultCert.LIBERTY') -
  SIGNWITH(CERTAUTH LABEL('LibertyCA.LIBERTY')) SIZE(2048) -
  NOTAFTER(DATE(2018/12/30))
RACDCERT ID(LIBSERV) ADDRING(Keyring.LIBERTY)
RACDCERT CONNECT(ID(LIBSERV) -
  LABEL('DefaultCert.LIBERTY') RING(Keyring.LIBERTY)) -
  ID(LIBSERV)
RACDCERT CONNECT(CERTAUTH LABEL('LibertyCA.LIBERTY') -
  RING(Keyring.LIBERTY)) ID(LIBSERV)
RACDCERT ID(FRED) GENCERT SUBJECTSDN(CN('Fred D. Client') -
  O('IBM') OU('LIBERTY')) WITHLABEL('FRED') -
  SIGNWITH(CERTAUTH LABEL('LibertyCA.LIBERTY')) SIZE(2048) -
  NOTAFTER(DATE(2018/12/30))
RACDCERT ID(FRED) EXPORT(LABEL('FRED')) -
  DSN('USER1.FRED.P12') FORMAT(PKCS12DER) -
  PASSWORD('secret')
RACDCERT ID(FRED) EXPORT(LABEL('FRED')) -
  DSN('USER1.FRED.PEM') -
  PASSWORD('secret')
RACDCERT ID(USER1) GENCERT SUBJECTSDN(CN('USER1 D. Client') -
  O('IBM') OU('LIBERTY')) WITHLABEL('USER1') -
  SIGNWITH(CERTAUTH LABEL('LibertyCA.LIBERTY')) SIZE(2048) -
  NOTAFTER(DATE(2018/12/30))
RACDCERT ID(USER1) EXPORT(LABEL('USER1')) -
  DSN('USER1.USER1.P12') FORMAT(PKCS12DER) -
  PASSWORD('secret')
RACDCERT ID(USER1) EXPORT(LABEL('USER1')) -
  DSN('USER1.USER1.PEM') -
  PASSWORD('secret')
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
```

What? In summary, that's creating a server certificate and self-signing that certificate, then adding the certificates to a "keyring," which is the RACF mechanism for holding certificates. It is also exporting a client certificate for Fred to be used in the next (optional) lab section.

Mitch Johnson (mitchj@us.ibm.com)

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

___8. Go to the ISPF Edit Entry Panel (option *2)* by entering ISPF command ***=2*** on the command line and pressing **Enter**.

___9. Enter ***/wasetc/zc3lab*** into the area beside *Name* under *Other Partitioned, Sequential or VSAM Data Set, or z/OS UNIX file:* and press **Enter**.

___10. Use the ***EA*** (Edit ASCII) line command to open the *keyring.xml* file in EBCDIC mode. You should see:

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

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

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

1

2

Notes

- The *transportSecurity-1.0* feature has been added
- The basic *keystore* element specifying a truststore was replace keystore elements specify RACF keyrings.

___4. Change the *sever.xml* to include for *keyring.xml*

<include location="/wasetc/zc3lab/keyring.xml" optional="true"/>

```
<?xml version="1.0" encoding="UTF-8"?>
<server description="new server">
<include location="/wasetc/zc3lab/ipic.xml" optional="true"/>
<include location="/wasetc/zc3lab/saf.xml" optional="true"/>
<include location="/wasetc/zc3lab/keyring.xml" optional="true"/>
<include location="/wasetc/zc3lab/group.xml" optional="true"/>
```

___5. Start your server with MVS command ***/S BAQSTRT***.

___6. Close all instances of your Firefox browser[2].

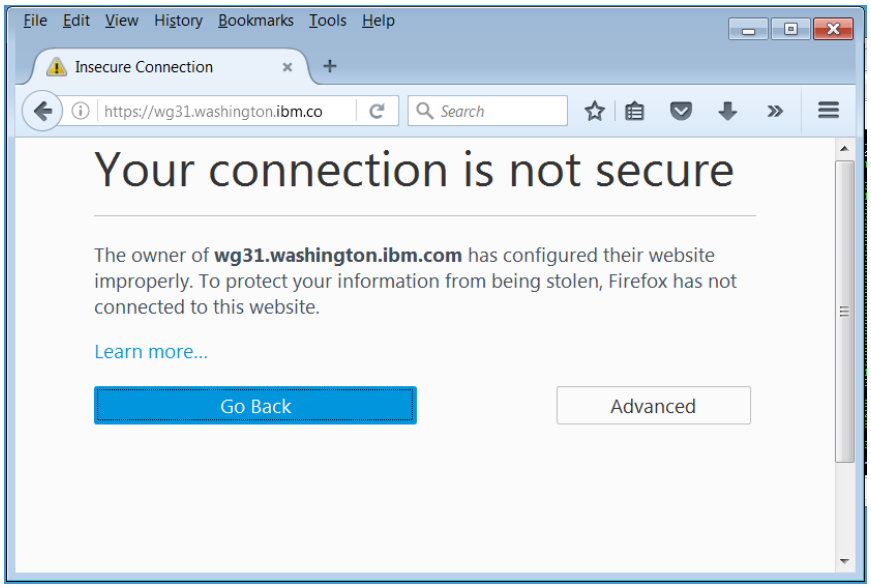
___7. Start Firefox and issue the following URL:

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

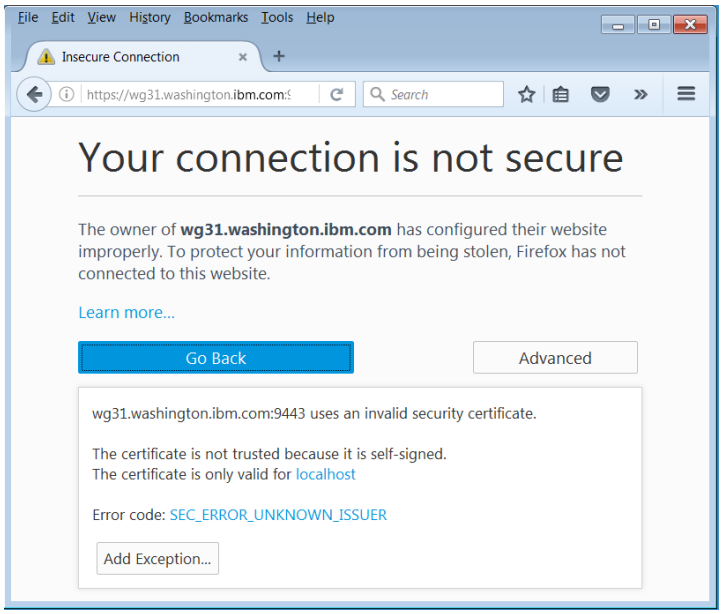
2 So the certificate accepted earlier is cleared and you're forced to see the new SAF-created certificate.

Mitch Johnson (mitchj@us.ibm.com)

___8. 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.

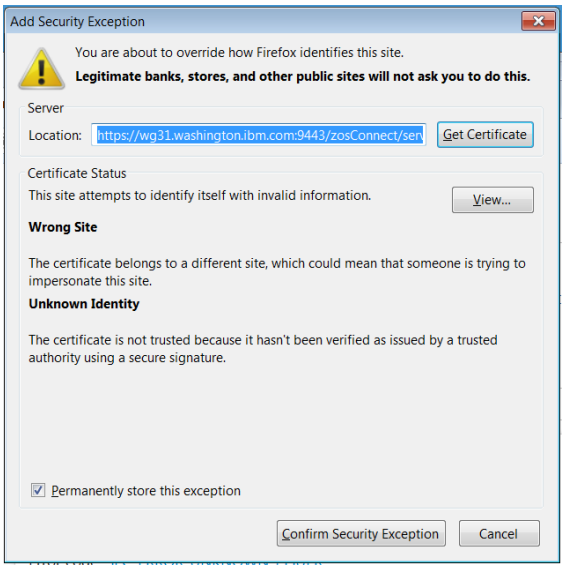


___8. Click the **Add Exception** button to continue.

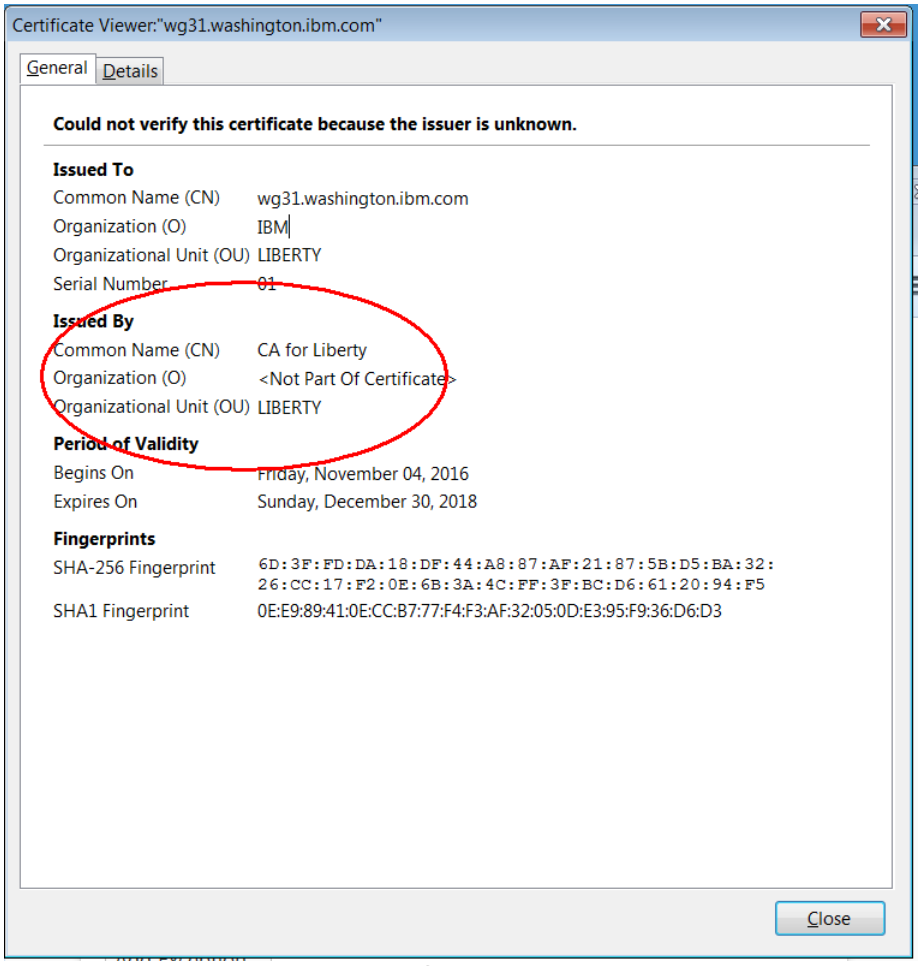


Mitch Johnson (mitchj@us.ibm.com)

___9. Click on the **View** button to display details about the certificate.



___10. This Certificate Authority (CA) that issued this certificate does not exist in the trust store used by Firefox. Click the **Close** button to continue.



Mitch Johnson (mitchj@us.ibm.com)

That is something different from what you saw earlier!

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('LibertyCA.LIBERTY') TRUST -
  SIZE(2048) NOTAFTER(DATE(2018/12/31))
```

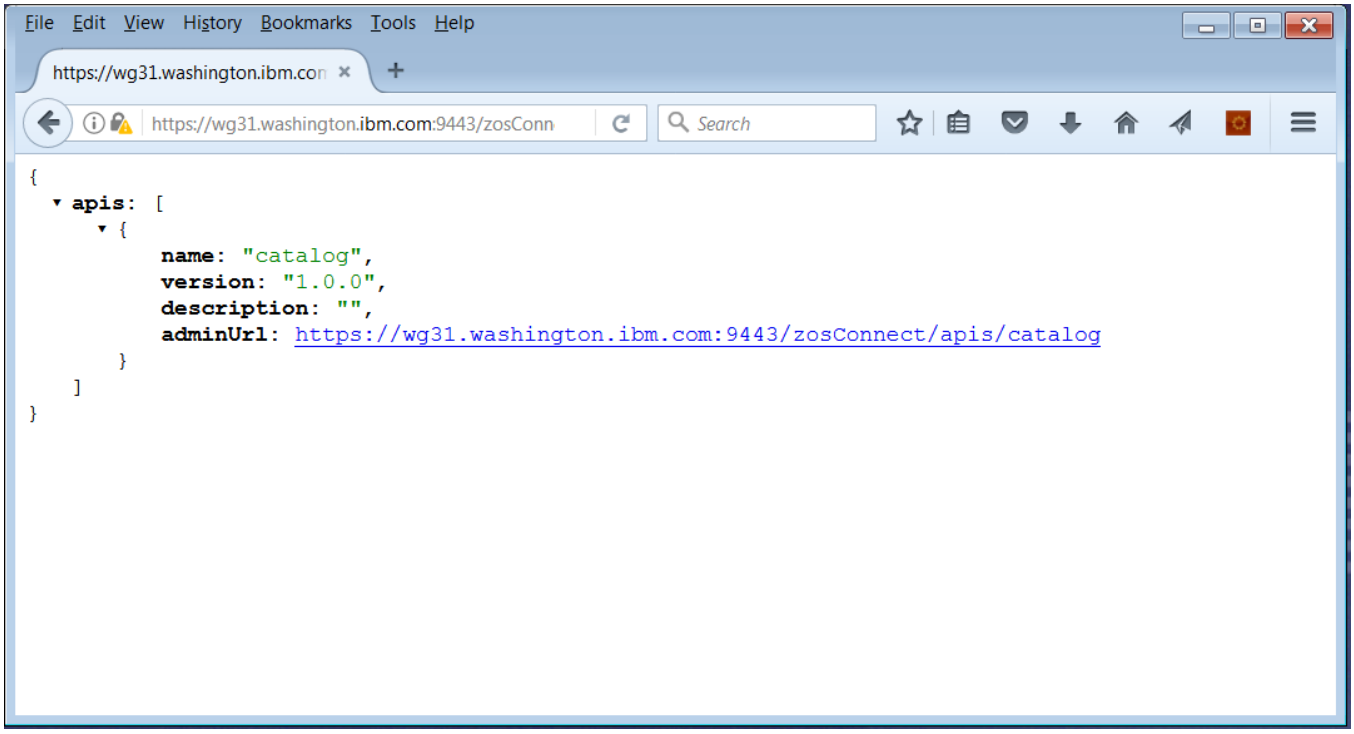
. **Note:** The certificate is still unverified, but it is different from before. That means it's no longer using the Liberty-generated certificate, but rather it is using the RACF-generated certificate in SAF.

___12. Click on the **Confirm Security Exception** button.

___13. 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.

___14. You should see a familiar list of APIs:



Mitch Johnson (mitchj@us.ibm.com)

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.

(Optional) Using client certificates

If time permits and you're interested in this topic, then proceed.

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

___2. Use the ***EA*** (Edit ASCII) line command to open the *mutual.xml* file in EBCDIC 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="CellDefaultTrustStore"
        clientAuthenticationSupport="true"
        clientAuthentication="true" />
    <keyStore id="CellDefaultKeyStore"
        location="safkeyring:///Keyring.LIBERTY"
        password="password" type="JCERACFKS"
        fileBased="false" readOnly="true" />
    <keyStore id="CellDefaultTrustStore"
        location="safkeyring:///Keyring.LIBERTY"
        password="password" type="JCERACFKS"
        fileBased="false" readOnly="true" />
</server>
```

1

Notes

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

Mitch Johnson (mitchj@us.ibm.com)

___5. Change the *sever.xml* so instead of an include for the file *keyring.xml* the file *mutual.xml* is included

<include location="/wasetc/zc3lab/mutual.xml" optional="true"/>

```
<?xml version="1.0" encoding="UTF-8"?>
<server description="new server">
<include location="/wasetc/zc3lab/ipic.xml" optional="true"/>
<include location="/wasetc/zc3lab/saf.xml" optional="true"/>
<include location="/wasetc/zc3lab/mutual.xml" optional="true"/>
<include location="/wasetc/zc3lab/group.xml" optional="true"/>
```

When you ran the **ZC2RACF4** job earlier, one of the things it did was generate and export a client certificate for Fred.

___6. Restart the server with MVS command ***/S BAQSTRT.***

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

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

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

___9. Enter the command: ***ftp wg31.washington.ibm.com***

___10. Logon with *USER1* and the password for that ID

___11. Enter the command: ***prompt off***

___12. Enter the command: ***mget *.pem***

___13. Enter the command: ***bin*** (this will set FTP mode to binary, or 'image')

___14. Enter the command: ***mget *.crt***

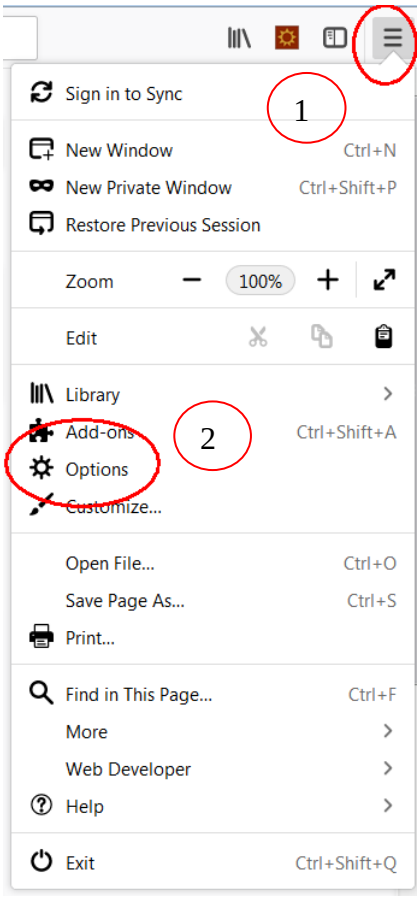
___15. Enter the command: ***mget *.p12***

___16. Enter the command***: quit***

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

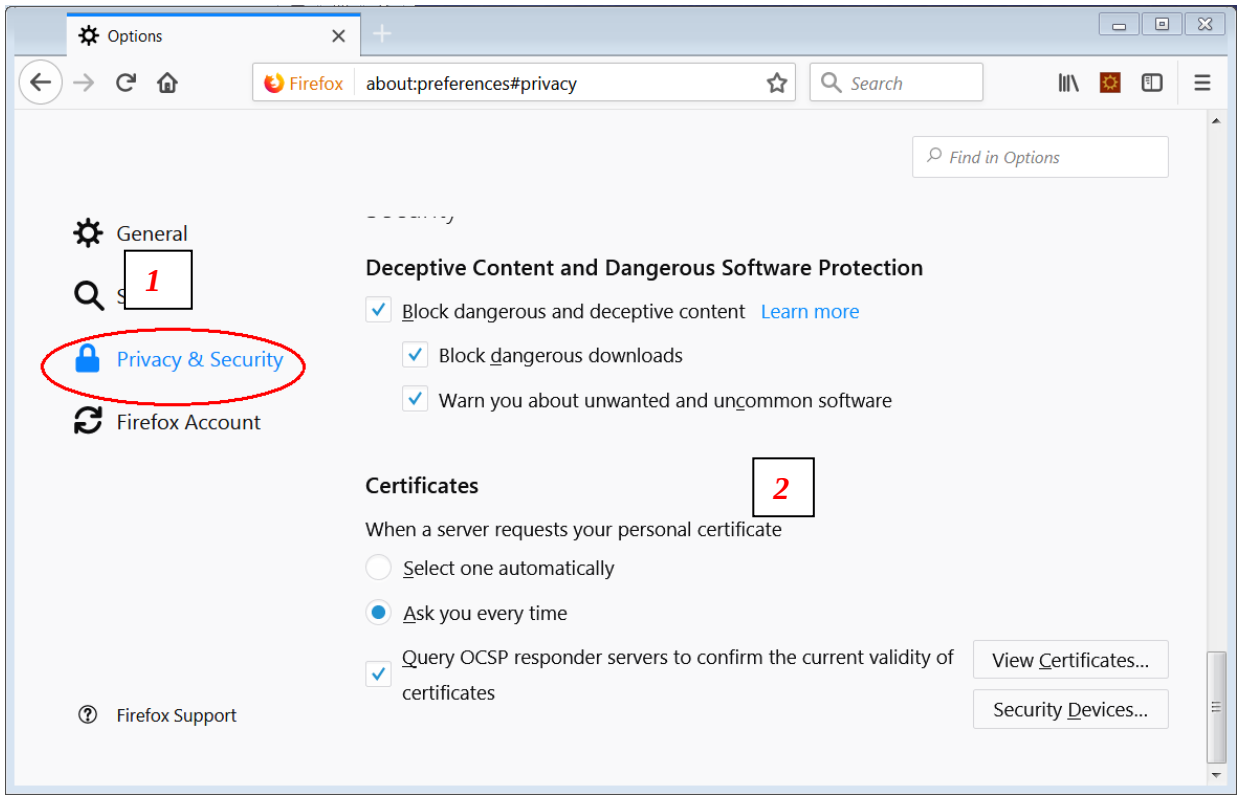
Mitch Johnson (mitchj@us.ibm.com)

___17. In Firefox, click on the to the *Open Menu* (1) icon and select the *Options* (2) tool.



Mitch Johnson (mitchj@us.ibm.com)

___18. Click on *Privacy & Security* (1) then scroll down to the *Certificates* (2) tab:



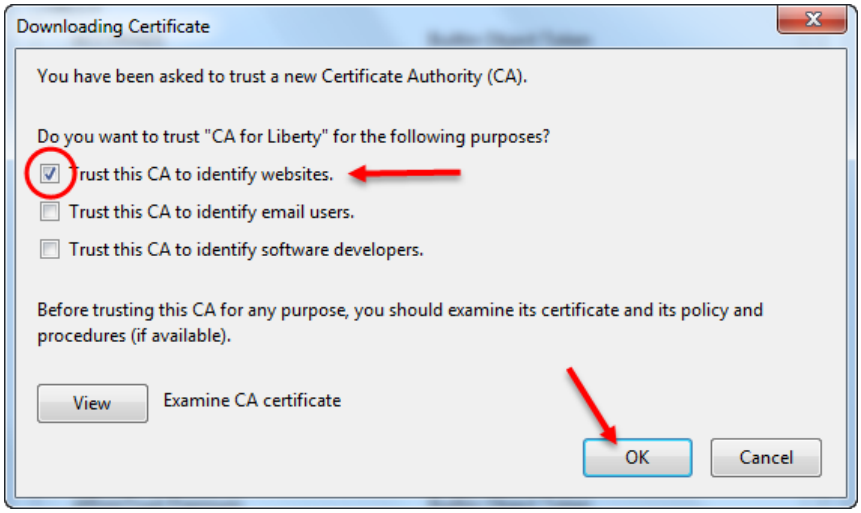
___19. Then click the **View Certificates** button.

___20. Then click on the *Authorities* tab, and the **Import** button.

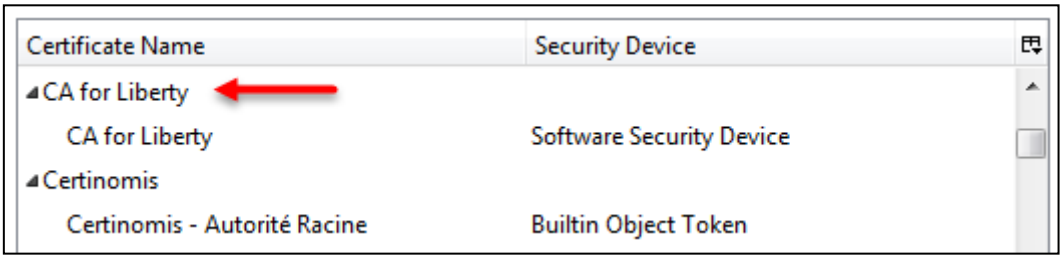
Mitch Johnson (mitchj@us.ibm.com)

___19. Navigate to the *c:\z directory* and double-click on the **certauth.crt** file.

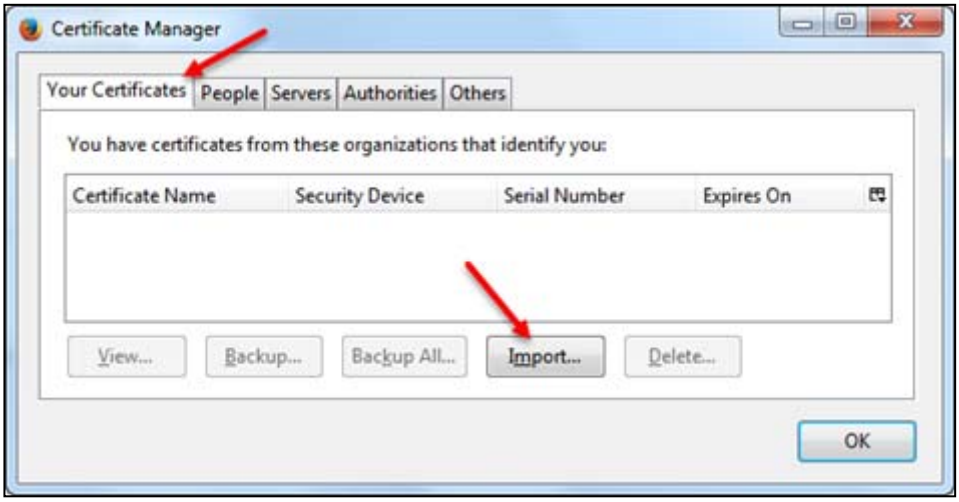
___20. Then check the *Trust this CA to identify websites* box and click **OK:**



___21. Verify the certificate has been imported by scrolling down and looking for the "CA for Liberty" certificate in the list:



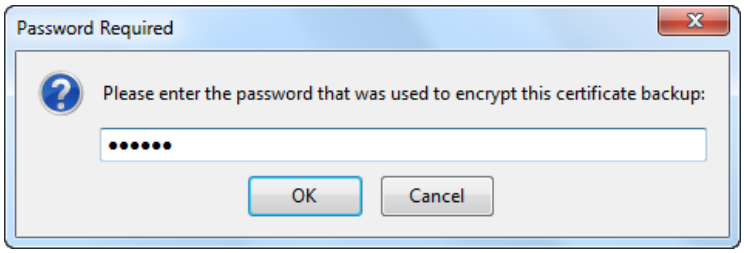
___22. Next, click the *Your certificates tab* and then the **Import** button:



Mitch Johnson (mitchj@us.ibm.com)

___23. It should open up at the *c:*\z directory from before, but if not then navigate to that location. Locate the **fred.p12** certificate and double-click on it.

___24. A window will appear asking you to enter the password for the certificate:



Enter the value[3] ***secret*** and click **OK**. You should see confirmation:

___25. Click **OK** to clear the confirmation, then

___26. **OK** to close the certificate manager panel, **OK** to close the options panel, and then close *all instances* of your Firefox browser.

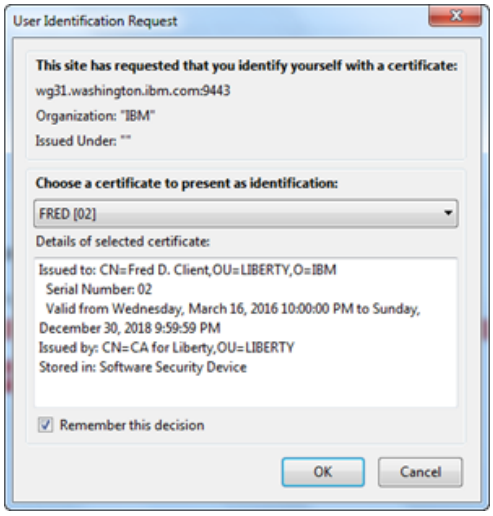
___27. Restart your server with ***/S BAQSTRT***.

___28. Start Firefox and go to URL ***https://wg31.washington.ibm.com:9443/zosConnect/services***
Click the **Send** button.

3 If interested, look in the ZC2RACF4 job to see where this value is specified on the certificate export.

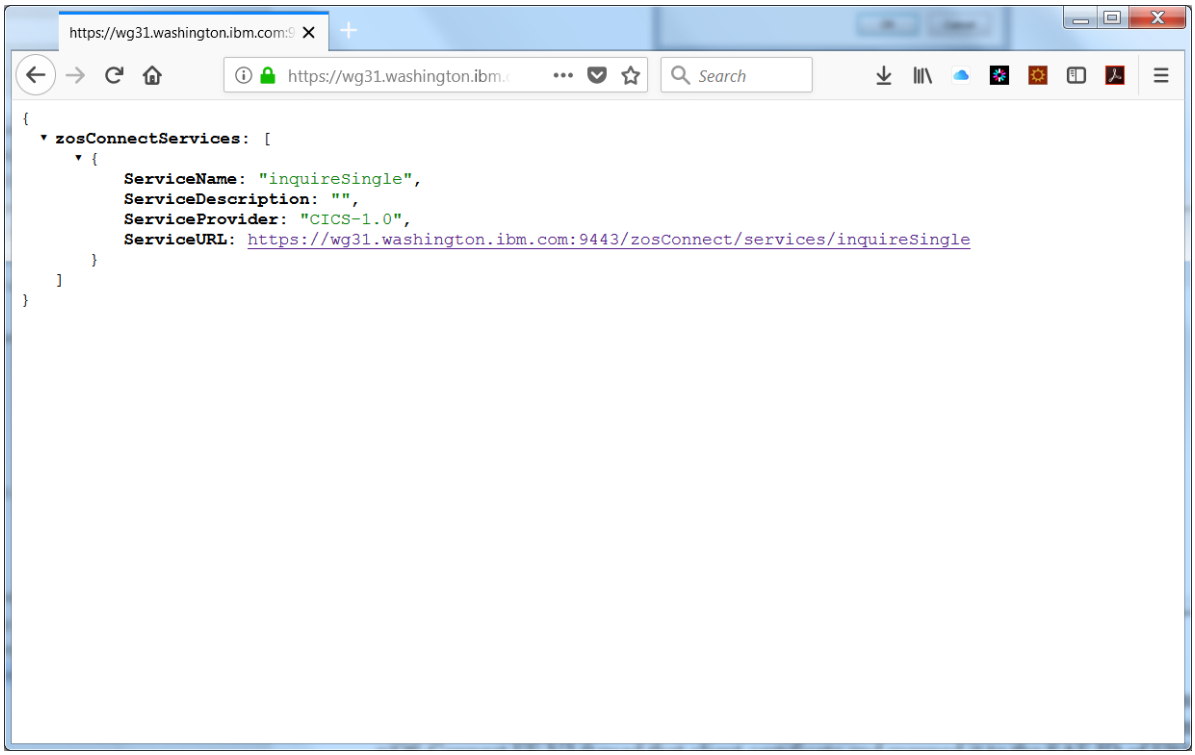
Mitch Johnson (mitchj@us.ibm.com)

___29. You will be prompted for which client certificate you wish to use:



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

___30. You should see the list of installed services:



___31. Use the *Command Prompt* icon on the desktop to open a DOS command prompt session.

___32. In the session use the change directory (cd) command to go to directory c:\z\CICSLab, e.g.

cd c:\z\CICSLAB

Mitch Johnson (mitchj@us.ibm.com)

___33. 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

___36. 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.

___37. 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

___37. 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.

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). z/OS Connect EE V3.0 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 example the client certificate specified by the *–cert* flag determined which identity was used for authorization checking in z/OS Connect EE because *clientAuthenication* was enabled.

Mitch Johnson (mitchj@us.ibm.com)