IBM z/OS Connect EE V3.0

Developing Outbound API Requesters Applications

IBM WSC Wildfire Team
IBM z Systems

Lab Version Date: July 24, 2019

Table of Contents

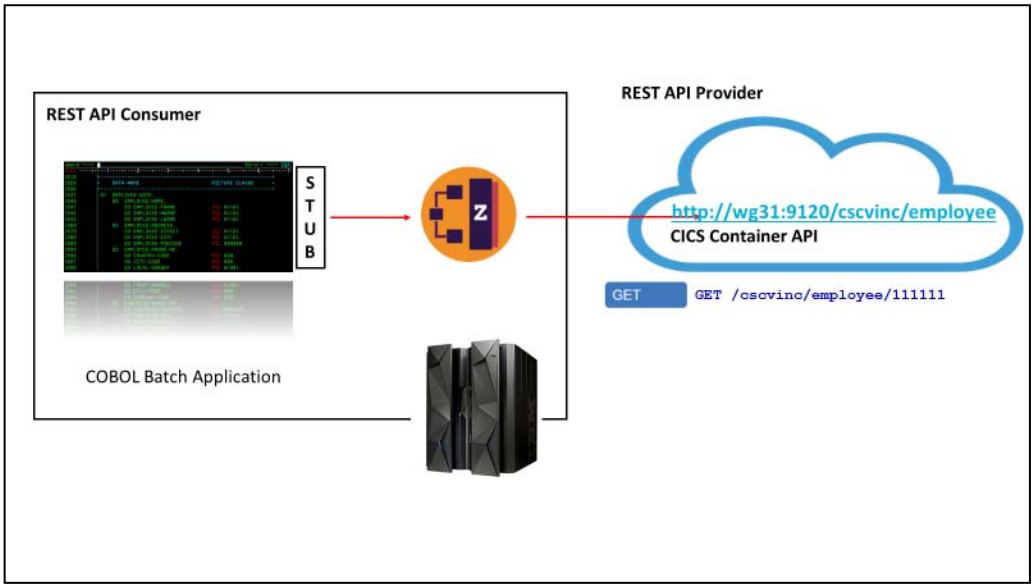
Mitch Johnson (mitchj@us.ibm.com)

Overview

These exercises demonstrate the steps required to enable COBOL programs to invoke RESTful APIs. There are two samples provided to do this. One sample uses RESTful APIs to access a CICS program and the other uses RESTful APIs to interact with an MQ queue. Note that even though these APIs are hosted in z/OS Connect EE server these same steps could be followed to access APIs hosted anywhere in the cloud.

The process starts with the Swagger document that describes the API which the COBOL program will invoke. The Swagger document is used by the z/OS Connect EE build toolkit to (1) generate the COBOL copy books included in the COBOL program to invoke the API (2) as well as an API requester archive file (ARA) which is deployed to a z/OS Connect EE server.

The diagram shows this process at a high level. The copy books generated by the z/OS Connect EE build toolkit will be integrated into a COBOL program to provide (1) the COBOL representation of the request and response messages and to identity (2) the HTTP method to be used in the RESTful request. The API requester archive file will be deployed to a z/OS Connect EE server and used to convert (1) the COBOL request and response messages to JSON messages and to invoke (2) the outbound RESTful request and wait for the response.



In this exercise the COBOL program will be compiled and executed in a batch environment. The only differences between batch and CICS and IMS is the configuration required for connecting to the z/OS Connect EE server. Consult the z/OS Connect EE Knowledge Center for information how this is done for CICS and IMS.

If you have completed either the developing APIs exercise for MVS Batch, DB2 or MQ you can start with section *RESTful API Requester to a CICS* on page 8.

Mitch Johnson (mitchj@us.ibm.com)

Important: The REST API provider applications used in this exercise access a CICS container application and MQ. These APIs were developed using the instructions provided in other exercises in this workshop. The REST API providers could have just as easily have been anywhere in the cloud on any remote system as long as the Swagger document for the REST API was available.

General Exercise Information and Guidelines

- ✓ The Windows artifacts for this exercise are in directory *c:\z\apiRequester*. Open a DOS command prompt Window and go to this directory using a change directory command, e.g. ***cd \z\apiRequester***
- ✓ Viewing or changing the contents of a file can easily be done by opening the Windows Explorer and going to directory *c:\z\apiRequester*. One this folder is display you can select a file and right mouse button click and select the *Open with EditPad Lite* option to open a file for viewing.
- ✓ This exercise requires using TSO user *USER1* and the TSO password for this user will be provided by the lab instructor.
- ✓ This exercise primarily uses data sets *USER1.ZCEE.CNTL* and *USER.ZCEE.SOURCE.*
- ✓ Any time you have any questions about the use of IBM z/OS Explorer, 3270 screens, features or tools do not hesitate to ask the instructor for assistance.

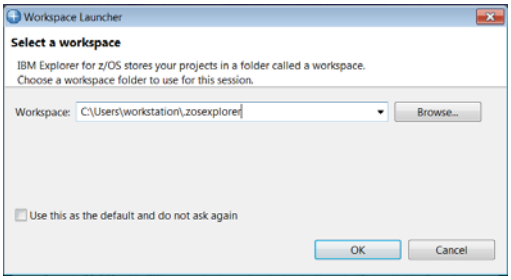
Connect the IBM z/OS Explorer to the z/OS system

Begin by establishing a connection to the z/OS system from IBM z/OS Explorer. If you have performed one of the other exercises in this series of exercises this step may not be required.

> **Tech-Tip:** Windows 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.

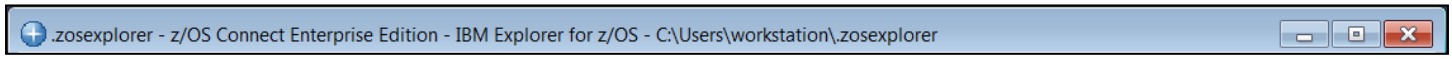
___1. On the workstation desktop, locate the *z/OS Explorer* icon and double click on it to open the Explorer.

___2. You will be prompted for a workspace:



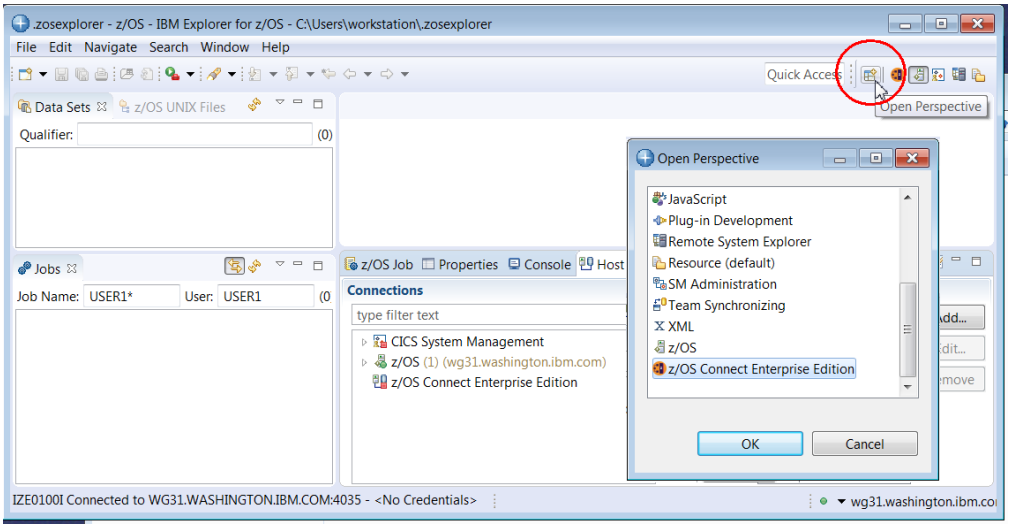
Take the default value by clicking **OK**.

___3. The Explorer should open in the *z/OS Connect Enterprise Edition* perspective. Verify this by looking in the upper left corner. You should see:



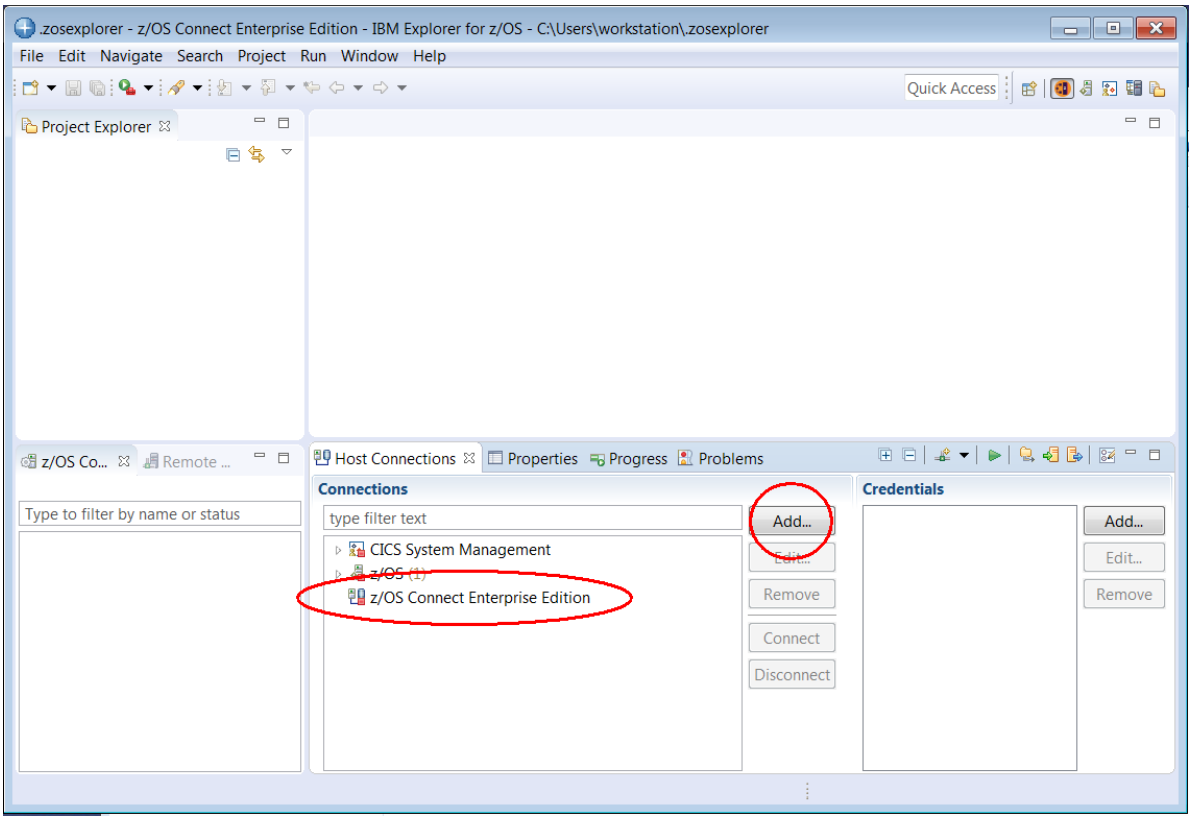
N.B. If a *Welcome* screen is displayed then click the white X beside *Welcome* to close this view.

___4. If the current perspective is not *z/OS Connect Enterprise Edition*, select the *Open Perspective* icon on the top right side to display the list of available perspectives, see below. Select **z/OS Connect Enterprise Edition** and click the **OK** button to switch to this perspective.



Mitch Johnson (mitchj@us.ibm.com)

___5. To add a connection to the z/OS Connect Server select *z/OS Connect Enterprise Edition* connection in the *Host connections* 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.

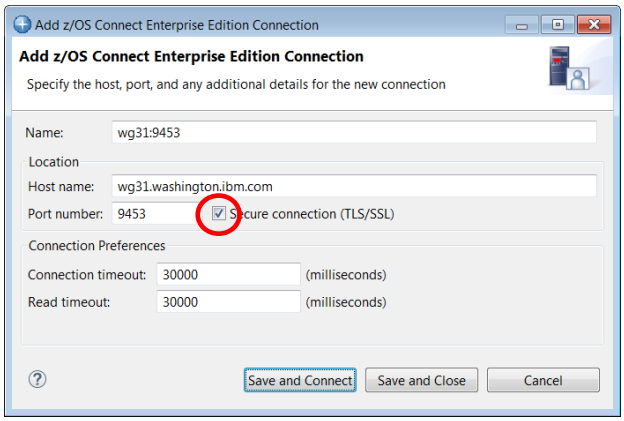
A view is an area in the window dedicated to providing a specific tool or function. For example, in the window above, *Host Connections* and *Remote Systems* are views that use different areas of the window for displaying information. At bottom on the right there is a single area for displaying the contents of four views stacked together (commonly called a *stacked views*), *z/OS Remote Systems, Tasks, Host Connections,* and *z/OS File Systems*. 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, the original arrangement can be restored by selecting Windows → Reset Perspective in the window's tool bar.

Eclipse based tools also can 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.

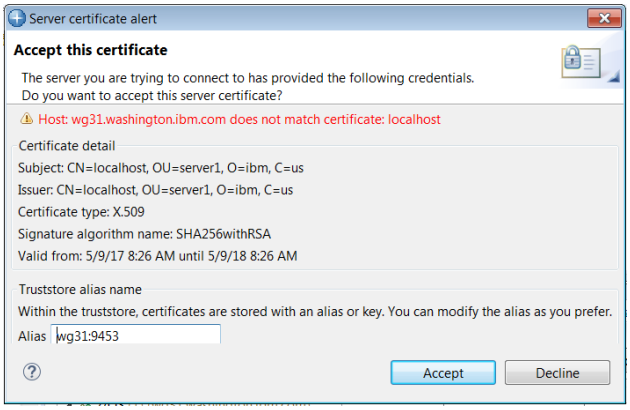
Mitch Johnson (mitchj@us.ibm.com)

___6. 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*, *9453* for the *Port Number*, check the box for *Secure connection (TLS/SSL)* and then click the **Save and Connect** button.



___7. On the *z/OS Connect Enterprise Edition – User ID* required screen 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. If SAF security had been enabled, then a valid RACF User ID and password will have to be used instead. Click **OK** to continue.

___8. 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.



___9. The status icon beside *wg31:9453* 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.

___10. A connection to the remote z/OS system was previously added. In the *Host Connection* view expand *z/OS Remote System* under *z/OS* and select *wg31.washington.ibm.com*. If the connection is not active the **Connect** button will be enabled. Click the **Connect** button and this will establish a session to the z/OS system. This step is required when submitting job for execution and viewing the output of these jobs later in this exercise.

Summary

The next step is the creation of the service and the composing and deployment of the API and then the testing of the API functions.

RESTful API Requester to a CICS RESTful API

In this section an API requester will be developed and tested where the target API accesses a CICS Container application. The batch programs will insert records (POST), retrieve records (GET), update record (PUT) and delete records (DELETE) in a VSAM data set.

Generate the API requester artifacts from a Swagger document

The first step is to use to use the API's Swagger document to generate the artifacts required for the COBOL API client application and the API requester archive file. The Swagger document was obtained from the server where the API is hosted. All the files mention in this section are in directory *c:/z/apiRequester/cscvinc* on your workstation.

The z/OS Connect EE build toolkit (ZCONBT) is a Java application shipped with z/OS Connect EE as a zip file and can be installed on a workstation or in OMVS. Anywhere a Java runtime is available. In this exercise the build toolkit will on Windows.

___1. Begin by reviewing the API's Swagger document for the CICS container-based API. Using a CICS container has implications which will be explored later. In the meantime, this document identifies the types of HTTP methods supported as well as the path of each HTTP method. It also the contents each method's request JSON message and the response JSON message field contents and field types. This is the blue print for invoking the API.

```
{

    "swagger": "2.0",
    "info": {
        "description": "",
        "version": "1.0.0",
        "title": "cscvinc"
    },
    "basePath": "/cscvinc",
    "schemes": [
        "https",
        "http"
    ],
    "consumes": [
        "application/json"
    ],
    "produces": [
        "application/json"
    ],
    "paths": {
        "/employee": {
            "post": {
                "operationId": "postCscvincService",
                "parameters": [
                    {
                        "in": "body",
                        "name": "postCscvincService_request",
                        "description": "request body",
                        "required": true,
                        "schema": {
                            "$ref": "#/definitions/postCscvincService_request"
                        }
                    },
```

Tech-Tip: Instructions for installing the z/OS Connect EE build toolkit can be found at URL https://www.ibm.com/support/knowledgecenter/en/SS4SVW_3.0.0/com.ibm.zosconnect.doc/installing/bt_install.html

___2. Next review the input parameters that will be passed to the z/OS Connect EE build toolkit in a properties file. This file is *cscvinc.properties* see its below:

```
apiDescriptionFile=./cscvinc.swagger [1]
dataStructuresLocation=./syslib [2]
apiInfoFileLocation=./syslib [3]
logFileDirectory=./logs [4]
language=COBOL [5]
connectionRef=cscvincAPI [6]
requesterPrefix=csc [7]
```

1. Identifies the Swagger document that describes the API
2. Provides the directory where the generated COBOL copybooks representing the request and response messages will be written.
3. Provides the directory where the generated COBOL copybook with invocation details of the API will be written.
4. Provides the directory where a log file will be written.
5. Identifies the target language.
6. Identifies the corresponding *zosconnect_endpointConnection* element define in the z/OS Connect EE server's server.xml file.
7. Provides a 3-character prefix to be used for all generated copybooks

___3. View file *cscvincAPI.bat* which contains this one line.

```
c:\z\zconbt\bin\zconbt.bat -p=./cscvinc.properties -f=./cscvinc.ara
```

This is batch command file that when executed invokes the z/OS Connect EE build toolkit *(zconbt*). The -p switch identifies the input properties file and the -f switch identifies the name of the generate API requester archive file. The API requester archive (ARA) file will be deployed and made available to the z/OS Connect EE server in directory *…/resources/zosconnect/apiRequesters.*

___4. On the workstation desktop, locate the *Command Prompt* icon and double click on it to open a DOS session that will allow commands to be entered.

___5. Use the change directory command (cd) to change to directory C:\z*api**Requester**cscvinc*, e.g.

cd c:\z\apiRequester\cscvinc

Mitch Johnson (mitchj@us.ibm.com)

___6. Enter command ***cscvincAPI.bat*** in the DOS command prompt window and you should see output like the below:

```
c:\z\apiRequester\cscvinc>c:\z\zconbt\bin\zconbt.bat -p=./cscvinc.properties -f=./cscvinc.ara
BAQB0000I: z/OS Connect Enterprise Edition Build Toolkit Version 1.0
BAQB0008I: Creating API requester archive from configuration file ./cscvinc.properties
BAQB0015I: Start processing operation (operationId: putCscvincService, relativePath: /employee,
method: PUT).
BAQB0016I: Successfully processed operation (operationId: putCscvincService, relativePath: /employee,
method: PUT).
BAQB0015I: Start processing operation (operationId: postCscvincService, relativePath: /employee,
method: POST).
BAQB0016I: Successfully processed operation (operationId: postCscvincService, relativePath: /employee,
method: POST).
BAQB0015I: Start processing operation (operationId: getCscvincService, relativePath: /employee/{numb},
method: GET).
BAQB0016I: Successfully processed operation (operationId: getCscvincService, relativePath:
/employee/{numb}, method: GET).
BAQB0015I: Start processing operation (operationId: deleteCscvincService, relativePath:
/employee/{numb}, method: DELETE).
BAQB0016I: Successfully processed operation (operationId: deleteCscvincService, relativePath:
/employee/{numb}, method: DELETE).

Total 4 operation(s) (success: 4, ignored: 0) defined in api description file: ./cscvinc.swagger
------ Successfully processed operation(s) ------
operationId: putCscvincService, basePath: /cscvinc, relativePath: /employee, method: PUT
- request data structure  : CSC00Q01
- response data structure : CSC00P01
- api info file           : CSC00I01

operationId: postCscvincService, basePath: /cscvinc, relativePath: /employee, method: POST
- request data structure  : CSC01Q01
- response data structure : CSC01P01
- api info file           : CSC01I01

operationId: getCscvincService, basePath: /cscvinc, relativePath: /employee/{numb}, method: GET
- request data structure  : CSC02Q01
- response data structure : CSC02P01
- api info file           : CSC02I01

operationId: deleteCscvincService, basePath: /cscvinc, relativePath: /employee/{numb}, method: DELETE
- request data structure  : CSC03Q01
- response data structure : CSC03P01
- api info file           : CSC03I01


BAQB0009I: Successfully created API requester archive file ./cscvinc.ara
```

Note that each HTTP method defined in the Swagger document will cause the generation of up to three copy books. One for the request message(Q01), one for the response message(P01) and one for the API details(I01). The copy book names are based on the *requesterPrefix* property (e.g. *csc*) and use an ascending sequence number sequence to differentiate between methods. Also note that there may be fewer than three copy books generated. For example, if there is no response message or no request message for a specific method a copy book may not be generated.

___7. Now explore a sample snippet of the copy book for the request message of a PUT method. Use *EditPad Lite* to open file c:\z\apiRequester\cscvinc\syslib\CSC00Q01. Scroll down until you see something like the code below.

```
           06 ReqBody.

             09 cscvincServiceOperatio-num    PIC S9(9) COMP-5 SYNC.
             09 cscvincServiceOperation.
               12 Container1.

                 15 REQUEST-CONTAINER2-num        PIC S9(9) COMP-5 SYNC.
                 15 REQUEST-CONTAINER.

                   18 FILEA-AREA2-num            PIC S9(9) COMP-5 SYNC.
                   18 FILEA-AREA.
                     21 NUMB-num                 PIC S9(9) COMP-5 SYNC.
                     21 NUMB.
                       24 NUMB2-length           PIC S9999 COMP-5 SYNC.
                       24 NUMB2                  PIC X(6).
                     21 NAME-num                 PIC S9(9) COMP-5 SYNC.
                     21 NAME.
                       24 NAME2-length           PIC S9999 COMP-5 SYNC.
                       24 NAME2                  PIC X(20).
```

Note that each variable has an associated length (e.g. *NUMB2-length).* The length of a variable is required since there is no delimiter between the variables in storage. The runtime needs to know the size of a variable at execution time so the request and response messages can be properly converted to and from JSON name/value pairs.

For this API, the Swagger document included an additional variable for the number of occurrences of a variable, (e.g. *NUMB-num*). In this exercise we will set the number of each variable to one.

Tech-Tip: This additional variable was included for this API because the target program is a CICS container enabled program and a CICS channel can have multiple occurrences of the same container. The z/OS Connect EE API requester support needs to know number of each variable is being sent (and returned). This behavior is not unique to CICS.

The corresponding response message (*c:\z\apiRequester\cscvinc\syslib\CSC00P01*) will have a similar layout.

___8. The corresponding API information copy book (*c:\z\apiRequester\cscvinc\syslib\CSC00I01*) has contents providing the API name and the path to be used for the method:

```
           03 BAQ-APINAME                 PIC X(255)
              VALUE 'cscvinc_1.0.0'.
           03 BAQ-APINAME-LEN             PIC S9(9) COMP-5 SYNC
              VALUE 13.
           03 BAQ-APIPATH                 PIC X(255)
              VALUE '/cscvinc/employee'.
           03 BAQ-APIPATH-LEN             PIC S9(9) COMP-5 SYNC
              VALUE 17.
           03 BAQ-APIMETHOD               PIC X(255)
              VALUE 'PUT'.
           03 BAQ-APIMETHOD-LEN           PIC S9(9) COMP-5 SYNC
              VALUE 3.
```

Deploy the API Requester Archive file to the z/OS Connect EE server

Next make the API requester archive file available to the z/OS Connect EE server.

The API requester archive (ARA) file needs to be deployed to the API requester directory. In this case the target directory is */var/ats/zosconnect/servers/zceeapir/resources/zosconnect/apiRequesters*.

___1. Open a DOS command session and change to directory c:\z\apiRequester\cscvinc, e.g **cd c:\z\apiRequester\cscvinc**

___2. Use the z/OS Connect EE RESTful administrative interface to deploy the ARA files by using the cURL command embedded in command file *deployCscvinc.bat*. Invoke this command file to deploy the cscvinc API archive file.

```
c:\z\apiRequester\cscvinc>curl -X POST --user Fred:fredpwd --data-binary @cscvin
c.ara  --header "Content-Type: application/zip" --insecure https://wg31.washingt
on.ibm.com:9483/zosConnect/apiRequesters
{"name":"cscvinc_1.0.0","version":"1.0.0","description":"","status":"Started","a
piRequesterUrl":"https:\/\/wg31.washington.ibm.com:9483\/zosConnect\/apiRequeste
rs\/cscvinc_1.0.0","connection":"cscvincAPI"}
```

Tech-Tip: If a REST client tool like cURL or Postman was not available then ARA file could have been deployed using FTP to upload the file in binary mode to the *apiRequesters* directory.

Tech-Tip: If an ARA needs to be redeployed, the cURL command with a PUT method can be used to stop the API requester
curl -X PUT --user Fred:fredpwd --insecure https://wg31.washington.ibm.com:9483/zosConnect/apiRequesters/cscvinc_1.0.0?status=stopped
And the cURL command with a DELETE method can be used to delete the API requester.
curl -X DELETE --user Fred:fredpwd --insecure https://wg31.washington.ibm.com:9483/zosConnect/apiRequesters/cscvinc_1.0.0
otherwise the redeployment of the ARA file will fail.

Mitch Johnson (mitchj@us.ibm.com)

___3. The z/OS Connect EE server where the API requester will be installed has the configuration below.

```
<server description="API Requester">
   <!-- Enable features -->
   <featureManager>
       <feature>zosconnect:apiRequester-1.0</feature>
   </featureManager>

 <zosconnect_endpointConnection id="cscvincAPI"      1
        host="http://wg31.washington.ibm.com"
        port="9120"
        basicAuthRef="myBasicAuth"
        connectionTimeout="10s"
        receiveTimeout="20s" />

 <zosconnect_authData id="myBasicAuth"     2
        user="Fred"
        password="fredpwd" />

</server>
```

1. Identifies the system which hosts the target API as well as any security information and time out parameters. Note that the ID of this element matches the value *connectionRef* property used when the ARA was generated by the z/OS Connect build toolkit.
2. The target server uses basic authentication, so we are simply provided a user identity and password. The password can be encrypted in the server.xml file or TLS used for security.

Move the COBOL copy books to a MVS data set

Next make the generated COBOL copy books available for including into the COBOL program.

The generated COBOL copy books need to be available in an MVS data set which is included in the compiler SYSLIB concatenation sequence. In the data set is *USER1.ZCEE. SOURCE*. Use a DOS command prompt session and the commands below to move the copy books file to this data set. Authenticate as user USER1 (password USER1).

___1. Open a DOS command prompt and use the change directory command to go to directory C:\z\apiRequester\cscvinc\syslib, e.g. ***cd \z\apiRequester\cscvinc\syslib***

___2. Start a file transfer session with the WG31 host using the *ftp* command, e.g. **ftp wg31**

___3. Logon as USER1 and then use the *cd* command to change to directory to data set *USER1.ZCEE. SOURCE, e.g.* ***cd zcee.source***

___4. Toggle prompting off by entering command **prompt**

___5. Perform multiple put requests by using the multiple put command, **mput csc***

___6. When the last transfer has completed enter the **quit** command.

```
c:\z\apiRequester\cscvinc>cd syslib
c:\z\apiRequester\cscvinc\syslib>ftp wg31.washington.ibm.com
Connected to wg31.washington.ibm.com.
220-FTP 16:26:23 on 2018-02-15.
220 Connection will close if idle for more than 200 minutes.
User (wg31.washington.ibm.com:(none)): user1
331 Send password please. user1
Password:
230 USER1 is logged on.  Working directory is "USER1.".
ftp> cd zcee.source
250 The working directory "USER1.ZCEE.SOURCE" is a partitioned data set
ftp> prompt
Interactive mode Off .
ftp> mput CSC*
200 Port request OK.
125 Storing data set USER1.ZCEE.SOURCE(CSC00I01)
250 Transfer completed successfully.
ftp: 533 bytes sent in 0.32Seconds 1.66Kbytes/sec.
200 Port request OK.
sent in 0.10Seconds 110.89Kbytes/sec.
200 Port request OK.
…….
125 Storing data set USER1.ZCEE.SOURCE(CSC03P01)
250 Transfer completed successfully.
ftp: 7393 bytes sent in 0.32Seconds 22.89Kbytes/sec.
200 Port request OK.
125 Storing data set USER1.ZCEE.SOURCE(CSC03Q01)
250 Transfer completed successfully.
ftp: 1132 bytes sent in 0.40Seconds 2.80Kbytes/sec.
ftp> quit
```

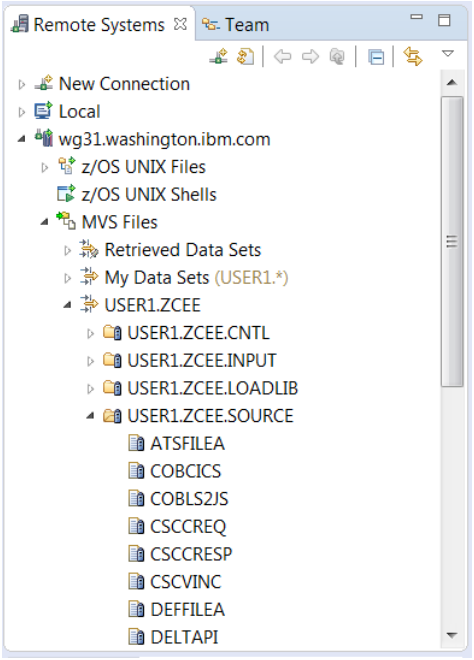
Review the application programs

Now that the copy books have been generated and are available in a data set the next step is compile and link edit the application programs. The source for the application programs are in USER1.ZCEE.SOURCE. Member DELTAPI invokes the DELETE method, member GETAPI invokes the GET method, POSTAPI invokes the POST method and PUTAPI invokes the PUT method.

Tech-Tip: Data set USER1.ZCEE.SOURCE can be accessed either by using the IBM z/OS Explorer filter created earlier in this exercise or by using the Personal Communication icon on the desktop and logging on to TSO and use ISPF option 3.4 to access this data set. If you have any questions about either method ask the instructor.

Mitch Johnson (mitchj@us.ibm.com)

___1. Back in the IBM z/OS Explorer session switch to the *Remote System Explorer* perspective (see page 5) and expand data set *USER1.ZCEE.SOURCE* in the *Remote System view* to display a list of its members.



___2. Begin by reviewing the application programs GETAPI. Open member *GETAPI* in *USER1.ZCEE. SOURCE* and review its contents by selecting the member and right- mouse button clicking and selecting the *Open* option. Scroll down until you see these lines of code.

These lines of code copy into the source the required z/OS Connect EE copy book (BAQRINFO) and the copy books generated by the z/OS Connect EE build tool kit. If when you generated the copy book earlier in the exercise and the sequence number used for the GET method was not 02 then you need to change the COBOL source to match sequence generated when you invoke the ZCONBT tool.

```
      * Copy API Requester required copybook
       COPY BAQRINFO.

      * Request and Response
       01 API-REQUEST.
           COPY CSC02Q01.
       01 API_RESPONSE.
           COPY CSC02P01.
      * Structure with the API information
       01 API-INFO-OPER1.
           COPY CSC02I01.
```


Mitch Johnson (mitchj@us.ibm.com)

___3. Scroll down further until you find this code which provides the contents of the request message.

```
      *-------------------------------------------------------------*
      * Set up the data for the API Requester call                  *
      *------------------- ----------------------------------------*
           MOVE numb     of PARM-DATA TO numb IN API-REQUEST.
           MOVE LENGTH of numb in API-REQUEST to
               numb-length IN API-REQUEST.

      *-------------------------------------------------------------*
      * Initialize API Requester PTRs & LENs                        *
      *-------------------------------------------------------------*
      * Use pointer and length to specify the location of
      *  request and response segment.
      * This procedure is general and necessary.
           SET BAQ-REQUEST-PTR TO ADDRESS OF API-REQUEST.
           MOVE LENGTH OF API-REQUEST TO BAQ-REQUEST-LEN.
           SET BAQ-RESPONSE-PTR TO ADDRESS OF API_RESPONSE.
           MOVE LENGTH OF API_RESPONSE TO BAQ-RESPONSE-LEN.
```

The *GET* method only requires one variable and it is provided as a path parameter, e.g. /cscvinc/employee/{numb}.

___4. Scroll down further and you will the call to the z/OS Connect API requester stub module.

```
      *-------------------------------------------------------------*
      * Call the communication stub                                 *
      *-------------------------------------------------------------*
      * Call the subsystem-supplied stub code to send
      * API request to zCEE
           CALL COMM-STUB-PGM-NAME USING
                BY REFERENCE   API-INFO-OPER1
                BY REFERENCE   BAQ-REQUEST-INFO
                BY REFERENCE   BAQ-REQUEST-PTR
                BY REFERENCE   BAQ-REQUEST-LEN
                BY REFERENCE   BAQ-RESPONSE-INFO
                BY REFERENCE   BAQ-RESPONSE-PTR
                BY REFERENCE   BAQ-RESPONSE-LEN.
      * The BAQ-RETURN-CODE field in 'BAQRINFO' indicates whether this
      * API call is successful.
```

Note that the parameters are being passed by reference. This means that the z/OS Connect API requester stub will have direct access to the program storage for both accessing data and making changes (i.e. the response message).

Mitch Johnson (mitchj@us.ibm.com)

___5. Scroll down further and you will see the code where the application displays the results in the response message.

```
           IF BAQ-SUCCESS THEN
              DISPLAY "NUMB:   " numb2   of API_RESPONSE
              DISPLAY "NAME:   " name2   of API_RESPONSE
              DISPLAY "ADDRX:  " addrx2  of API_RESPONSE
              DISPLAY "PHONE:  " phone2  of API_RESPONSE
              DISPLAY "DATEX:  " datex2  of API_RESPONSE
              DISPLAY "AMOUNT: " amount2 of API_RESPONSE
              MOVE CEIBRESP of API_RESPONSE to EIBRESP
              MOVE CEIBRESP2 of API_RESPONSE to EIBRESP2
              DISPLAY "EIBRESP:    " EIBRESP
              DISPLAY "EIBRESP2:   " EIBRESP2
              DISPLAY "HTTP CODE:  " BAQ-STATUS-CODE
```

Close member *GETAPI* and open member *POSTAPI*. The code for this program is very similar to the code for *GETAPI* with the exception that all variables are provided in a JSON request message. Scroll down and find the code below.

```
      *---------------------------------------------------------------*
      * Set up the data for the API Requester call                    *
      *---------------------------------------------------------------*
           MOVE 1 to cscvincServiceOperatio-num in API-REQUEST
                     REQUEST-CONTAINER2-num in API-REQUEST
                     FILEA-AREA2-num in API-REQUEST
                     NUMB-num in API-REQUEST
                     NAME-num in API-REQUEST
                     NUMB-num in API-REQUEST
                     NAME-num in API-REQUEST
                     ADDRX-num in API-REQUEST
                     PHONE-num in API-REQUEST
                     DATEX-num in API-REQUEST
                     AMOUNT-num in API-REQUEST.

           MOVE numb of PARM-DATA TO numb2 IN API-REQUEST.
           MOVE LENGTH of numb2 in API-REQUEST to
               numb2-length IN API-REQUEST.

           MOVE "John" TO name2 IN API-REQUEST.
           MOVE LENGTH of name2 in API-REQUEST to
               name2-length IN API-REQUEST.
```

In this case the number of each occurrences of a property name must be provided (1) and then the, the length of the property value and finally the property value moved to the request copy book.

___6. Explore members *DELTAPI* and *PUTAPI* and their corresponding copy books to see the code which is common (variable housekeeping, calls to the z/OS Connect stub, etc.) regardless of the method being invoked.

Compile and link-edit the application programs

The applications programs now are ready to be compiled and link-edited.

___1. Submit the job in member **APIRCL1** in data set *USER1.ZCEE.CNTL*.

```
//COMPILE EXEC IGYWCL,LNGPRFX=SYS1.ECOBOL,PARM.COBOL='NODYNAM
//COBOL.SYSIN  DD DISP=SHR,DSN=USER1.ZCEE.SOURCE(POSTAPI)
//COBOL.SYSLIB DD DISP=SHR,DSN=USER1.ZCEE.SOURCE
//             DD DISP=SHR,DSN=ZCEE30.SBAQCOB
//LKED.SYSLMOD DD DISP=SHR,DSN=USER1.ZCEE.LOADLIB(POSTAPI)
//LKED.BAQLIB  DD DISP=SHR,DSN=ZCEE30.SBAQLIB
//LKED.SYSIN   DD *
  INCLUDE BAQLIB(BAQCSTUB)
//COMPILE EXEC IGYWCL,LNGPRFX=SYS1.ECOBOL,PARM.COBOL='NODYNAM
//COBOL.SYSIN  DD DISP=SHR,DSN=USER1.ZCEE.SOURCE(GETAPI)
//COBOL.SYSLIB DD DISP=SHR,DSN=USER1.ZCEE.SOURCE
//             DD DISP=SHR,DSN=ZCEE30.SBAQCOB
//LKED.SYSLMOD DD DISP=SHR,DSN=USER1.ZCEE.LOADLIB(GETAPI)
//LKED.BAQLIB  DD DISP=SHR,DSN=ZCEE30.SBAQLIB
//LKED.SYSIN   DD *
  INCLUDE BAQLIB(BAQCSTUB)
//COMPILE EXEC IGYWCL,LNGPRFX=SYS1.ECOBOL,PARM.COBOL='NODYNAM'
//COBOL.SYSIN  DD DISP=SHR,DSN=USER1.ZCEE.SOURCE(PUTAPI)
//COBOL.SYSLIB DD DISP=SHR,DSN=USER1.ZCEE.SOURCE
//             DD DISP=SHR,DSN=ZCEE30.SBAQCOB
//LKED.SYSLMOD DD DISP=SHR,DSN=USER1.ZCEE.LOADLIB(PUTAPI)
//LKED.BAQLIB  DD DISP=SHR,DSN=ZCEE30.SBAQLIB
//LKED.SYSIN   DD *
  INCLUDE BAQLIB(BAQCSTUB)
//COMPILE EXEC IGYWCL,LNGPRFX=SYS1.ECOBOL,PARM.COBOL='NODYNAM'
//COBOL.SYSIN  DD DISP=SHR,DSN=USER1.ZCEE.SOURCE(DELTAPI)
//COBOL.SYSLIB DD DISP=SHR,DSN=USER1.ZCEE.SOURCE
//             DD DISP=SHR,DSN=ZCEE30.SBAQCOB
//LKED.SYSLMOD DD DISP=SHR,DSN=USER1.ZCEE.LOADLIB(DELTAPI)
//LKED.BAQLIB  DD DISP=SHR,DSN=ZCEE30.SBAQLIB
//LKED.SYSIN   DD *
  INCLUDE BAQLIB(BAQCSTUB)
```

Note that the CALL statement in each of this program uses a variable for the program name z/OS Connect EE stub program (BAQCSTUB). This means that this program will by dynamically linked at execution. The load module library containing BAQSCTUB (ZCEE30.SBAQLIB) must be available in the STEPLIB concatenation sequence when the program is executed.

This job should complete with a zero-return code.

> **Tech Tip:** To submit a job when using the IBM z/OS Explorer select the member and right mouse button click and select the *Submit* option. Or if the member is currently opened simply right mouse button click and select the *Submit* option. Click the **Locate Job** button on the *Job Confirmation* pop-up. This will display the job output in the *Retrieved Job* section under *JES* in the *Remote Systems* view. The job's output can be viewed right mouse button clicking and selecting the *Open* option.

Mitch Johnson (mitchj@us.ibm.com)

Test the API requester application programs

The JCL to execute these programs can be found in USER1.ZCEE.CNTL, the member names in this data set match the program names. The members are DELTAPI, GETAPI, POSTAPI and PUTAPI.

___1. Submit the job in member **GETAPI** in *USER1.ZCEE.CNTL*.

```
//GETAPI EXEC PGM=GETAPI,PARM='111111'
//STEPLIB  DD DISP=SHR,DSN=USER1.ZCEE.LOADLIB
//         DD DISP=SHR,DSN=ZCEE30.SBAQLIB
//SYSPRINT DD SYSOUT=*
//CEEOPTS  DD *
 POSIX(ON),
 ENVAR("BAQURI=wg31.washington.ibm.com",
 "BAQPORT=9120")
```

___2. The job should complete with a condition of zero and have the following output for SYSPRINT.

```
NUMB:   111111
NAME:   C. BAKER
ADDRX:  OTTAWA, ONTARIO
PHONE:  51212003
DATEX:  26 11 81
AMOUNT: $0011.00
EIBRESP:    00000000
EIBRESP2:   00000000
HTTP CODE:  0000000200
```

___3.

Tech-Tip: An HTTP code of 200 indicates success. For an explanation of HTTP codes see URL https://en.wikipedia.org/wiki/List_of_HTTP_status_codes

___3. Change the PARM=’111111’ to PARM=’000000’ and resubmit. This time the output should look like this:

```
NUMB:   000000
NAME:
ADDRX:
PHONE:
DATEX:
AMOUNT:
EIBRESP:    00000013
EIBRESP2:   00000080
HTTP CODE:  0000000200
```

The EIBRESP and EIBRESP2 values are from a CICS program and the response codes received when a EXEC CICS READ fails with a *Not Found* condition.

___4. Submit the job in member **POSTAPI** in *USER1.ZCEE.CNTL*.

```
//POSTAPI EXEC PGM=POSTAPI,PARM='323232'
//STEPLIB  DD DISP=SHR,DSN=USER1.ZCEE.LOADLIB
//         DD DISP=SHR,DSN=ZCEE30.SBAQLIB
//SYSPRINT DD SYSOUT=*
//CEEOPTS  DD *
 POSIX(ON),
 ENVAR("BAQURI=wg31.washington.ibm.com",
 "BAQPORT=9120")
```

___5. The job should complete with a condition of zero and have the following output for SYSPRINT.

```
NUMB:   323232
NAME:   John
ADDRX:  Apex
PHONE:  0065
DATEX:  11 22 65
AMOUNT: $1000.65
EIBRESP:    00000000
EIBRESP2:   00000000
HTTP CODE:  0000000200
```

___6.

___7. Member **DELTAPI** in *USER.ZCEE.CNTL* will delete a record and **PUTAPI** will update an existing record. Submit these jobs and observe the results. Or if you want to modify the request messages in POSTAPI and PUTAPI and recompile and relink these programs and observe the results when different values are provided.

Mitch Johnson (mitchj@us.ibm.com)

The available records are listed below:

numb	name	addrx	Phone	datex	amount
000100	S. D. BORMAN	SURREY, ENGLAND	32156778	26 11 81	$0100.11
000102	J. T. CZAYKOWSI	WARWICK, ENGLAND	98356183	26 11 81	$1111.11
000104	M. B. DOMBEY	LONDON,ENGLAND	12846293	26 11 81	$0999.99
000106	A. I. HICKSON	CROYDON, ENGLAND	19485673	26 11 81	$0087.71
000111	ALAN TULIP	SARATOGA,CALIFORNIA	46120753	01 02 74	$0111.11
000762	SUSAN MALAIKA	SAN JOSE,CALIFORNIA	22312121	01 06 74	$0000.00
000983	J. S. TILLING	WASHINGTON, DC	34512120	21 04 75	$9999.99
001222	D.J.VOWLES	BOBLINGEN, GERMANY	70315551	10 04 73	$3349.99
001781	TINA J YOUNG	SINDELFINGEN,GERMANY	70319990	21 06 77	$0009.99
003210	B.A. WALKER	NICE, FRANCE	12345670	26 11 81	$3349.99
003214	PHIL CONWAY	SUNNYVALE, CAL.	34112120	00 06 73	$0009.99
003890	BRIAN HARDER	NICE FRANCE	00000000	28 05 74	$0009.99
004004	JANET FOUCHE	DUBLIN, IRELAND	71112121	02 11 73	$1259.99
004445	DR. P. JOHNSON	SOUTH BEND, S.DAK.	61212120	26 11 81	$0009.99
004878	ADRIAN JONES	SUNNYVALE, CALIF.	32212120	10 06 73	$5399.99
005005	A. E. DALTON	SAN FRANCISCO, CA.	00000001	01 08 73	$0009.99
005444	ROS READER	SARATOGA, CALIF.	67712120	20 10 74	$0809.99
005581	PETE ROBBINS	BOSTON, MASS.	41312120	11 04 74	$0259.99
006016	SIR MICHAEL ROBERTS	NEW DELHI, INDIA	70331211	21 05 74	$0009.88
006670	IAN HALL	NEW YORK, N.Y.	21212120	31 01 75	$3509.88
006968	J.A.L. STAINFORTH	WARWICK, ENGLAND	56713821	26 11 81	$0009.88
007007	ANDREW WHARMBY	STUTTGART, GERMANY	70311000	10 10 75	$5009.88
007248	M. J. AYRES	REDWOOD CITY, CALF.	33312121	11 10 75	$0009.88
007779	MRS. A. STEWART	SAN JOSE, CALIF.	41512120	03 01 75	$0009.88
009000	P. E. HAVERCAN	WATERLOO, ONTARIO	09876543	21 01 75	$9000.00
100000	M. ADAMS	TORONTO, ONTARIO	03415121	26 11 81	$0010.00
111111	C. BAKER	OTTAWA, ONTARIO	51212003	26 11 81	$0011.00
200000	S. P. RUSSELL	GLASGOW, SCOTLAND	63738290	26 11 81	$0020.00
222222	DR E. GRIFFITHS	FRANKFURT, GERMANY	20034151	26 11 81	$0022.00
300000	V. J. HARRIS	NEW YORK, U.S.	64739801	26 11 81	$0030.00
333333	J.D. HENRY	CARDIFF, WALES	78493020	26 11 81	$0033.00
400000	C. HUNT	MILAN, ITALY	25363738	26 11 81	$0040.00
444444	D. JACOBS	CALGARY, ALBERTA	77889820	26 11 81	$0044.00
500000	P. KINGSLEY	MADRID, SPAIN	44454640	26 11 81	$0000.00
555555	S.J. LAZENBY	KINGSTON, N.Y.	39944420	26 11 81	$0005.00
600000	M.F. MASON	DUBLIN, IRELAND	12398780	26 11 81	$0010.00
666666	R. F. WALLER	LA HULPE, BRUSSELS	42983840	26 11 81	$0016.00
700000	M. BRANDON	DALLAS, TEXAS	57984320	26 11 81	$0002.00
777777	L.A. FARMER	WILLIAMSBURG, VIRG.	91876131	26 11 81	$0027.00
800000	P. LUPTON	WESTEND, LONDON	24233389	26 11 81	$0030.00
888888	P. MUNDY	NORTHAMPTON, ENG.	23691639	26 11 81	$0038.00
900000	D.S. RENSHAW	TAMPA, FLA.	35668120	26 11 81	$0040.00
999999	ANJI STEVENS	RALEIGH, N.Y.	84591639	26 11 81	$0049.00

Summary

You have use the z/OS Connect build toolkit to generate an API requester archive file and the COBOL copy books that must be included in the COBOL application program when accessing the API requester archive file in a z/OS Connect server. The COBOL applications have been compiled and link edited and the target API has been tested using various RESTful methods

RESTful API Requester to a MQ One-Way RESTful API

In this section an API requester will be developed and tested where the target API accesses a MQ queue. The batch programs will be put messages (POST) to the queue, do non-destructive gets (GET) and destructive gets (DELETE) of messages from the same queue.

Generate the API requester artifacts from a Swagger document

The first step is to use to use the API's Swagger document to generate the artifacts required for the COBOL API client application and the API requester archive file. The Swagger document was obtained from the server where the API is hosted. All the files mention in this section are in directory *c:/z/apiRequester/fileaqueue* on your workstation.

The z/OS Connect EE build toolkit (ZCONBT) is a Java application shipped with z/OS Connect EE as a zip file and can be installed on a workstation or in OMVS. Anywhere a Java runtime is available. In this exercise the build toolkit will on Windows.

___1. Begin by reviewing the API's Swagger document for the MQ one-way API. This document identifies the types of HTTP methods supported as well as the path of each HTTP method. It also the contents each method's request JSON message and the response JSON message field contents and field types. This is the blue print for invoking the API.

```
{

    "swagger": "2.0",
    "info": {
        "description": "",
        "version": "1.0.0",
        "title": "fileaqueue"
    },
    "basePath": "/fileaqueue",
    "schemes": [
        "https",
        "http"
    ],
    "consumes": [
        "application/json"
    ],
    "produces": [
        "application/json"
    ],
    "paths": {
        "/mq": {
            "get": {
                "operationId": "getFilea",
                "parameters": [
                    {
                        "name": "Authorization",
                        "in": "header",
                        "required": false,
                        "type": "string"
                    }
                ],
```

Tech-Tip: Instructions for installing the z/OS Connect EE build toolkit can be found at URL https://www.ibm.com/support/knowledgecenter/en/SS4SVW_3.0.0/com.ibm.zosconnect.doc/installing/bt_install.html

___2. Next review the input parameters that will be passed to the z/OS Connect EE build toolkit in a properties file. This file is *fileaqueue.properties* see its below:

```
apiDescriptionFile=./fileaqueue.swagger [1]
dataStructuresLocation=./syslib [2]
apiInfoFileLocation=./syslib [3]
logFileDirectory=./logs [4]
language=COBOL [5]
connectionRef=fileaqueueAPI [6]
requesterPrefix=imq [7]
```

1. Identifies the Swagger document that describes the API

2. Provides the directory where the generated COBOL copybooks representing the request and response messages will be written.

3. Provides the directory where the generated COBOL copybook with invocation details of the API will be written.

4. Provides the directory where a log file will be written.

5. Identifies the target language.

6. Identifies the corresponding *zosconnect_endpointConnection* element define in the z/OS Connect EE server's server.xml file.

7. Provides a 3-character prefix to be used for all generated copybooks

___3. View file *fileaqueueAPI.bat* which contains this one line.

```
c:\z\zconbt\bin\zconbt.bat -p=./fileaqueue.properties -f=./fileaqueue.ara
```

This is batch command file that when executed invokes the z/OS Connect EE build toolkit *(zconbt*). The -p switch identifies the input properties file and the -f switch identifies the name of the generate API requester archive file. The API requester archive (ARA) file will be uploaded and make available to the z/OS Connect EE server in directory *…/resources/zosconnect/apiRequsters.*

___4. On the workstation desktop, locate the *Command Prompt* icon and double click on it to open a DOS session that will allow commands to be entered.

___5. Use the change directory command (cd) to change to directory *apiRequester*/*fileaqueue*, e.g.

cd c:\z\apiRequester\fileaqueue

Mitch Johnson (mitchj@us.ibm.com)

___6. Enter command ***fileaqueueAPI.bat*** in the DOS command prompt window and you should see output like the below:

```
c:\z\apiRequester\fileaqueue>c:\z\zconbt\bin\zconbt.bat -p=./fileaqueue.properties -f=./fileaqueue.ara
BAQB0000I: z/OS Connect Enterprise Edition Build Toolkit Version 1.0
BAQB0008I: Creating API requester archive from configuration file ./fileaqueue.properties
BAQB0015I: Start processing operation (operationId: getFilea, relativePath: /mq, method: GET).
BAQB0016I: Successfully processed operation (operationId: getFilea, relativePath: /mq, method: GET).
BAQB0015I: Start processing operation (operationId: postFileaQueue, relativePath: /mq, method: POST).
BAQB0016I: Successfully processed operation (operationId: postFileaQueue, relativePath: /mq, method:
POST).
BAQB0015I: Start processing operation (operationId: deleteFilea, relativePath: /mq, method: DELETE).
BAQB0016I: Successfully processed operation (operationId: deleteFilea, relativeP
ath: /mq, method: DELETE).

Total 3 operation(s) (success: 3, ignored: 0) defined in api description file: .
/fileaqueue.swagger
------ Successfully processed operation(s) ------
operationId: getFilea, basePath: /fileaqueue, relativePath: /mq, method: GET
- request data structure  : No data structure needs to be generated.
- response data structure : IMQ00P01
- api info file           : IMQ00I01

operationId: postFileaQueue, basePath: /fileaqueue, relativePath: /mq, method: POST
- request data structure  : IMQ01Q01
- response data structure : No data structure needs to be generated.
- api info file           : IMQ01I01

operationId: deleteFilea, basePath: /fileaqueue, relativePath: /mq, method: DELETE
- request data structure  : No data structure needs to be generated.
- response data structure : IMQ02P01
- api info file           : IMQ02I01


BAQB0009I: Successfully created API requester archive file ./fileaqueue.ara
```

Note that each HTTP method defined in the Swagger document will normally cause the generation of up to three copy books. One for the request message(Q01), one for the response message(P01) and one for the API details(I01). For this API only two copy books were generated for the messages. One for either the request message or one for the response message. In this case the API did not require both types for any method. Also note that the copy book names are based on the *requesterPrefix* property (e.g. *imq*) and use an ascending sequence number sequence to differentiate between methods.

Important: The sequence numbers assigned to a copy book seem to be arbitrary. When you compile the sample programs later in this exercise verify the copy books included for a particular method has the same sequence numbers as generated when you invoke the toolkit.

Mitch Johnson (mitchj@us.ibm.com)

___7. Now explore a sample snippet of the copy book for the request message of a PUT method. Use *EditPad Lite* to open file c:\z\apiRequester\fileaqueue\syslib\IMQ01Q01. Scroll down until you see something like the code below.

```
           06 ReqBody.
             09 ATSFILEAOperation.
               12 mqmessage.
                 15 stat-length                 PIC S9999 COMP-5           SYNC.
                 15 stat                        PIC X(1).
                 15 numb-length                 PIC S9999 COMP-5           SYNC.
                 15 numb                        PIC X(6).
                 15 name-length                 PIC S9999 COMP-5           SYNC.
                 15 name                        PIC X(20).
                 15 addrx-length                PIC S9999 COMP-5           SYNC.
                 15 addrx                       PIC X(20).
                 15 phone-length                PIC S9999 COMP-5           SYNC.
                 15 phone                       PIC X(8).
                 15 datex-length                PIC S9999 COMP-5           SYNC.
                 15 datex                       PIC X(8).
                 15 amount-length               PIC S9999 COMP-5           SYNC.
                 15 amount                      PIC X(8).
                 15 comment-length              PIC S9999 COMP-5           SYNC.
                 15 comment                     PIC X(9).
```

Note that each variable has an associated length (e.g. *stat-length).* The length of a variable is required since there is no delimiter between the variables in storage. The runtime needs to know the size of a variable at execution time so the request and response messages can be properly converted to and from JSON name/value pairs.

A response message (*c:\z\apiRequester\fileaqueue\syslib\IMQ02P01*) will have a similar layout.

___8. An API information copy book (*c:\z\apiRequester\fileaqueue\syslib\IMQ00I01*) has contents providing the API name and the path to be used for the method:

```
          03 BAQ-APINAME                PIC X(255)
             VALUE 'fileaqueue_1.0.0'.
          03 BAQ-APINAME-LEN            PIC S9(9) COMP-5 SYNC
             VALUE 16.
          03 BAQ-APIPATH                PIC X(255)
             VALUE '/fileaqueue/mq'.
          03 BAQ-APIPATH-LEN            PIC S9(9) COMP-5 SYNC
             VALUE 14.
          03 BAQ-APIMETHOD              PIC X(255)
             VALUE 'POST'.
          03 BAQ-APIMETHOD-LEN          PIC S9(9) COMP-5 SYNC
             VALUE 4.
```

Deploy the API Requester Archive file to the z/OS Connect EE server

Next make the API requester archive file available to the z/OS Connect EE server.

The API requester archive (ARA) file needs to be deployed to the API requester directory. In this case the target directory is */var/ats/zosconnect/servers/zceeapir/resources/zosconnect/apiRequesters*.

___1. Open a DOS command session and change to directory C:\z\apiRequester\fileaqueue, e.g
cd \z\apiRequester\fileaqueue

___4. Use the z/OS Connect EE RESTful administrative interface to deploy the ARA files by using the cURL command embedded in command file *deployFileaqueue.bat*. Invoke this command file to deploy the fileaaueue API archive file.

```
c:\z\apiRequester\fileaqueue>curl -X POST --user Fred:fredpwd --data-binary @fil
eaqueue.ara  --header "Content-Type: application/zip" --insecure
https://wg31.washington.ibm.com:9483/zosConnect/apiRequesters
{"name":"fileaqueue_1.0.0","version":"1.0.0","description":"","status":"Started"
,"apiRequesterUrl":"https:\/\/wg31.washington.ibm.com:9483\/zosConnect\/apiReque
sters\/fileaqueue_1.0.0","connection":"fileaqueueAPI"}
```

Tech-Tip: If a REST client tool like cURL or Postman was not available then ARA file could have been deployed using FTP to upload the file in binary mode to the *apiRequesters* directory.

Tech-Tip: If an ARA needs to be redeployed, the cURL command with a PUT method can be used to stop the API requester
curl -X PUT --user Fred:fredpwd --insecure
https://wg31.washington.ibm.com:9483/zosConnect/apiRequesters/fileaqueue_1.0.0?status=stopped
And the cURL command with a DELETE method can be used to delete the API requester.
curl -X DELETE --user Fred:fredpwd --insecure
https://wg31.washington.ibm.com:9483/zosConnect/apiRequesters/fileaqueue_1.0.0
otherwise the redeployment of the ARA file will fail.

Mitch Johnson (mitchj@us.ibm.com)

___2. The z/OS Connect EE server where the API requester will be installed has the configuration below in its server.xml file.

```
<server description="API Requester">
   <!-- Enable features -->
   <featureManager>
       <feature>zosconnect:apiRequester-1.0</feature>
   </featureManager>

 <zosconnect_endpointConnection id="fileaqueueAPI"       1
        host="http://wg31.washington.ibm.com"
        port="9120"
        basicAuthRef="myBasicAuth"
        connectionTimeout="10s"
        receiveTimeout="20s" />

 <zosconnect_authData id="myBasicAuth"      2
        user="Fred"
        password="fredpwd" />

</server>
```

1. Identifies the system which hosts the target API as well as any security information and time out parameters. Note that the ID of this element matches the value *connectionRef* property used when the ARA was generated by the z/OS Connect build toolkit.
2. The target server uses basic authentication, so we are simply provided a user identity and password. The password can be encrypted or TLS can be used for security.

Move the COBOL copy books to a MVS data set

Next make the generated COBOL copy books available for including into the COBOL program.

The generated COBOL copy books need to be available in an MVS data set which is included in the compiler SYSLIB concatenation sequence. In the data set is *USER1.ZCEE.SOURCE*. Use a DOS command prompt session and the commands below to move the copy books file to this data set. Authenticate as user USER1 (password USER1).

___1. Open a DOS command prompt and use the change directory command to go to directory C:\z\apiRequester\fileaqueue\syslib, e.g. ***cd \z\apiRequester\fileaqueue\syslib***

___2. Start a file transfer session with the WG31 host using the *ftp* command, e.g. **ftp wg31**

___3. Logon as USER1 and then use the *cd* command to change to directory to data set *USER1.ZCEE.SOURCE, e.g.* ***cd zcee.source***

___4. Toggle prompting off by entering command **prompt**

___5. Perform multiple put request by using the multiple put command, **mput imq***

___6. When the last transfer has completed enter the **quit** command.

```
c:\z\apiRequester\cscvinc>cd syslib
c:\z\apiRequester\cscvinc\syslib>ftp wg31.washington.ibm.com
Connected to wg31.washington.ibm.com.
220-FTP 16:26:23 on 2018-02-15.
220 Connection will close if idle for more than 200 minutes.
User (wg31.washington.ibm.com:(none)): user1
331 Send password please. user1
Password:
230 USER1 is logged on.  Working directory is "USER1.".
ftp> cd zcee.source
250 The working directory "USER1.ZCEE.SOURCE" is a partitioned data set
ftp> prompt
Interactive mode Off .
ftp> mput IMQ*
200 Port request OK.
125 Storing data set USER1.ZCEE.SOURCE(IMQ00I01)
250 Transfer completed successfully.
ftp: 533 bytes sent in 0.32Seconds 1.66Kbytes/sec.
200 Port request OK.
sent in 0.10Seconds 110.89Kbytes/sec.
200 Port request OK.
…….
125 Storing data set USER1.ZCEE.SOURCE(IMQ02I01)
250 Transfer completed successfully.
ftp: 7393 bytes sent in 0.32Seconds 22.89Kbytes/sec.
200 Port request OK.
125 Storing data set USER1.ZCEE.SOURCE(IMQ02P01)
250 Transfer completed successfully.
ftp: 1132 bytes sent in 0.40Seconds 2.80Kbytes/sec.
ftp> quit
```


Mitch Johnson (mitchj@us.ibm.com)

Review the application programs

Now that the copy books have been generated and are available in a data set the next step is compile and link edit the application programs. The source for the application programs are in *USER1.ZCEE. SOURCE*. Member DELMQAPI invokes the DELETE method, member GETMQAPI invokes the GET method and PSTMQAPI invokes the POST method. No changes need to be made to these programs.

Tech-Tip: Data set USER1.ZCEE.SOURCE can be accessed either by using the IBM z/OS Explorer filter created earlier in this exercise or by using the Personal Communication icon on the desktop and logging on to TSO and use ISPF option 3.4 to access this data set. If you have any questions about either method ask the instructor.

___1. Begin by reviewing the application programs GETMQAPI. Browse data set *USER1.ZCEE. SOURCE* and open member GETMQAPI. Scroll down until you see these lines of code.

```
      * Copy API Requester required copybook
       COPY BAQRINFO.

      * Request and Response
       01 API-REQUEST.
           10 FILLER              PIC X(1).
       01 API_RESPONSE.
           COPY IMQ00P01.
      * Structure with the API information
       01 API-INFO-OPER1.
           COPY IMQ00I01.
```

These lines of code copy into the source the required z/OS Connect EE copy book (BAQRINFO) and the copy books generated by the z/OS Connect EE build tool kit. If when you generated the copy book earlier in the exercise and the sequence number used for the GET method was not 02 then you need to change the COBOL source to match sequence generated when you invoke the ZCONBT tool.

___2. Scroll down further and you will the call to the z/OS Connect API requester stub module.

```
      *---------------------------------------------------------------*
      * Call the communication stub                                   *
      *---------------------------------------------------------------*
      * Call the subsystem-supplied stub code to send
      * API request to zCEE
           CALL COMM-STUB-PGM-NAME USING
                BY REFERENCE   API-INFO-OPER1
                BY REFERENCE   BAQ-REQUEST-INFO
                BY REFERENCE   BAQ-REQUEST-PTR
                BY REFERENCE   BAQ-REQUEST-LEN
                BY REFERENCE   BAQ-RESPONSE-INFO
                BY REFERENCE   BAQ-RESPONSE-PTR
                BY REFERENCE   BAQ-RESPONSE-LEN.
      * The BAQ-RETURN-CODE field in 'BAQRINFO' indicates whether this
      * API call is successful.
```

Note that the parameters are being passed by reference. This means that the z/OS Connect API requester stub will have direct access to the program storage for both accessing data and making changes (i.e. the response message).

___3. Scroll down further and you will see the code where the application displays the results in the response message.

```
           IF BAQ-SUCCESS THEN
              DISPLAY "NUMB:   " numb    of API_RESPONSE
              DISPLAY "NAME:   " name    of API_RESPONSE
              DISPLAY "ADDRX:  " addrx   of API_RESPONSE
              DISPLAY "PHONE:  " phone   of API_RESPONSE
              DISPLAY "DATEX:  " datex   of API_RESPONSE
              DISPLAY "AMOUNT: " amount  of API_RESPONSE
              DISPLAY "HTTP CODE: " BAQ-STATUS-CODE
```

___4. Exit member *GETMQAPI* and open member *PSTMQAPI*. The code for this program is very similar to the code for *GETMQAPI* with the exception that all variables are provided in a JSON request message. Scroll down and find the code below.

```
      *-------------------------------------------------------------*
      * Set up the data for the API Requester call                  *
      *-------------------------------------------------------------*

           MOVE "837367" TO numb IN API-REQUEST.
           MOVE LENGTH of numb in API-REQUEST to
               numb-length IN API-REQUEST.

           MOVE "John" TO name IN API-REQUEST.
           MOVE LENGTH of name in API-REQUEST to
               name-length IN API-REQUEST.

           MOVE "Apex" TO addrx IN API-REQUEST.
           MOVE LENGTH of addrx in API-REQUEST to
               addrx-length IN API-REQUEST.

           MOVE "0065" TO phone IN API-REQUEST.
           MOVE LENGTH of phone in API-REQUEST to
               phone-length IN API-REQUEST.

           MOVE "11 22 65" TO datex IN API-REQUEST.
           MOVE LENGTH of datex in API-REQUEST to
               datex-length IN API-REQUEST.

           MOVE "$1000.65" TO amount IN API-REQUEST.
           MOVE LENGTH of amount in API-REQUEST to
               amount-length IN API-REQUEST.
```

In this case the length of the property value and finally the property value moved to the request copy book.

___5. Explore members *DLTMQAPI* and their corresponding copy books to see the code which is common (variable housekeeping, calls to the z/OS Connect stub, etc.) regardless of the method being invoked.

Compile and link-edit the application programs

The applications programs now are ready to be compiled and link-edited.

___1. Submit the job in member **APIRCL2** in data set *USER1.ZCEE.CNTL*.

```
//USER1S JOB MSGCLASS=H,NOTIFY=&SYSUID
//* JCLLIB ORDER=SYS1.ECOBOL.SIGYPROC
//COMPILE EXEC IGYWCL,LNGPRFX=SYS1.ECOBOL,PARM.COBOL='NODYNAM'
//COBOL.SYSIN  DD DISP=SHR,DSN=USER1.ZCEE.SOURCE(GETMQAPI)
//COBOL.SYSLIB DD DISP=SHR,DSN=USER1.ZCEE.SOURCE
//             DD DISP=SHR,DSN=ZCEE30.SBAQCOB
//LKED.SYSLMOD DD DISP=SHR,DSN=USER1.ZCEE.LOADLIB(GETMQAPI)
//LKED.BAQLIB  DD DISP=SHR,DSN=ZCEE30.SBAQLIB
//LKED.SYSIN   DD *
  INCLUDE BAQLIB(BAQCSTUB)
//COMPILE EXEC IGYWCL,LNGPRFX=SYS1.ECOBOL,PARM.COBOL='NODYNAM'
//COBOL.SYSIN  DD DISP=SHR,DSN=USER1.ZCEE.SOURCE(DLTMQAPI)
//COBOL.SYSLIB DD DISP=SHR,DSN=USER1.ZCEE.SOURCE
//             DD DISP=SHR,DSN=ZCEE30.SBAQCOB
//LKED.SYSLMOD DD DISP=SHR,DSN=USER1.ZCEE.LOADLIB(DLTMQAPI)
//LKED.BAQLIB  DD DISP=SHR,DSN=ZCEE30.SBAQLIB
//LKED.SYSIN   DD *
  INCLUDE BAQLIB(BAQCSTUB)
//COMPILE EXEC IGYWCL,LNGPRFX=SYS1.ECOBOL,PARM.COBOL='NODYNAM'
//COBOL.SYSIN  DD DISP=SHR,DSN=USER1.ZCEE.SOURCE(PSTMQAPI)
//COBOL.SYSLIB DD DISP=SHR,DSN=USER1.ZCEE.SOURCE
//             DD DISP=SHR,DSN=ZCEE30.SBAQCOB
//LKED.SYSLMOD DD DISP=SHR,DSN=USER1.ZCEE.LOADLIB(PSTMQAPI)
//LKED.BAQLIB  DD DISP=SHR,DSN=ZCEE30.SBAQLIB
//LKED.SYSIN   DD *
  INCLUDE BAQLIB(BAQCSTUB)
```

Note that the CALL statement in each of this program uses a variable for the program name z/OS Connect EE stub program (BAQCSTUB). This means that this program will by dynamically linked at execution. The load module library containing BAQSCTUB (ZCEE30.SBAQLIB) must be available in the STEPLIB concatenation sequence when the program is executed.

The execution of this job should complete with a zero-return code.

Mitch Johnson (mitchj@us.ibm.com)

Test the API requester application programs

The JCL to execute these programs can be found in USER1.ZCEE.CNTL, the member names in this data set match the program names. The members are DLTMQAPI, GETMQAPI and PSTMQAPI.

___1. Submit the job in member **GETMQAPI** in *USER1.ZCEE.CNTL*.

```
//GETAPI EXEC PGM=GETMQAPI
//STEPLIB  DD DISP=SHR,DSN=USER1.ZCEE.LOADLIB
//         DD DISP=SHR,DSN=ZCEE30.SBAQLIB
//SYSPRINT DD SYSOUT=*
//CEEOPTS  DD *
 POSIX(ON),
 ENVAR("BAQURI=wg31.washington.ibm.com",
 "BAQPORT=9120")
```

___2. The job should complete with a condition of zero and have the following output for SYSPRINT.

```
NUMB:   000100
NAME:   S. D. BORMAN
ADDRX:  SURREY, ENGLAND
PHONE:  32156778
DATEX:  26 11 81
AMOUNT: $0100.11
HTTP CODE: 0000000200
```

Tech-Tip: An HTTP code of 200 indicates success. For an explanation of HTTP codes see URL https://en.wikipedia.org/wiki/List_of_HTTP_status_codes

___3. Submit **GETMQAPI** again and you should see the same results, a GET HTTP method is doing a non-destructive get of messages from the queue.

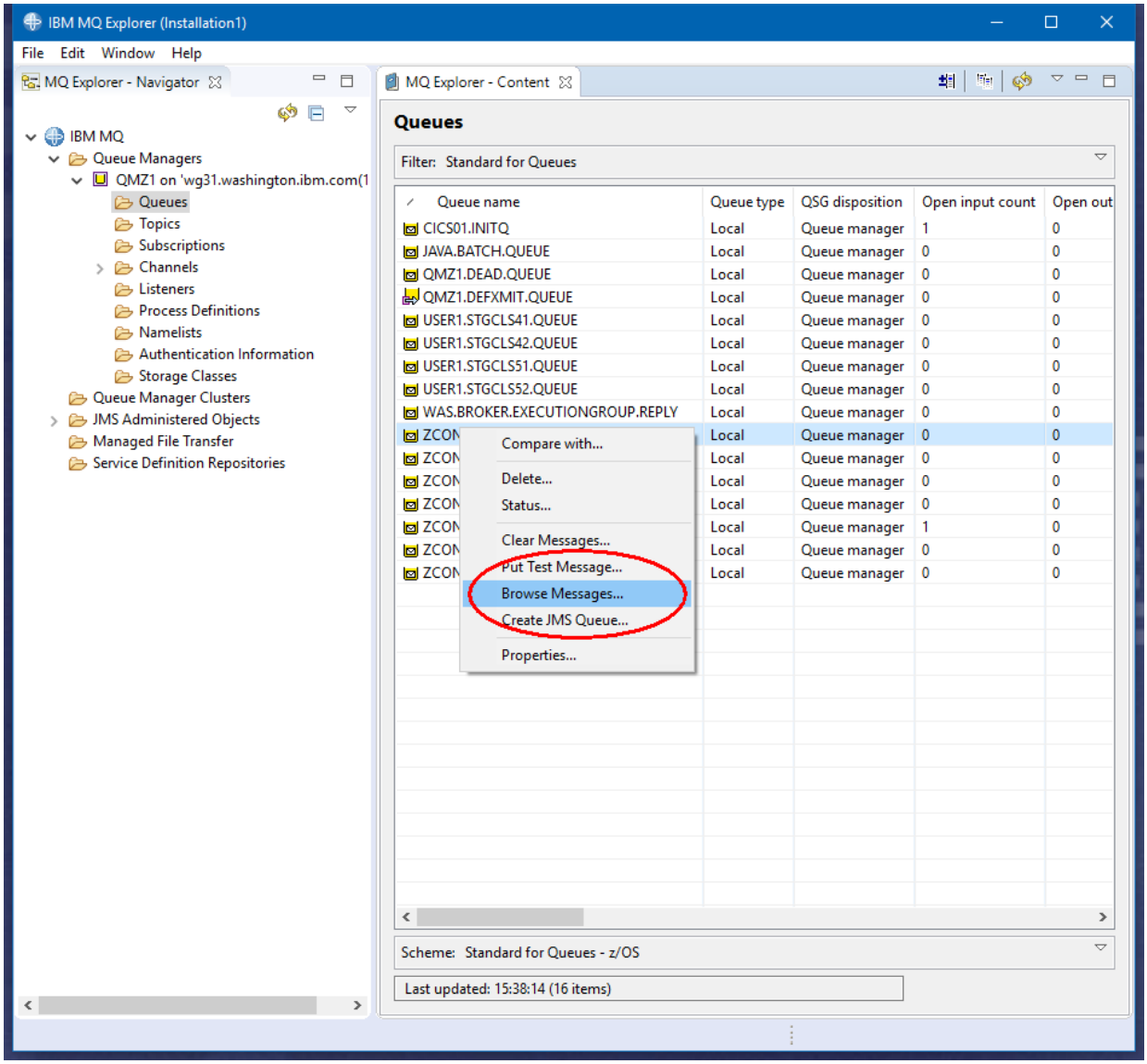
___4. Submit the job in member **DLTMQAPI** in *USER1.ZCEE.CNTL* multiple times. The results should be different each time, a DELETE HTP method is doing a destructive get of messages from the queue.

___5. Member **PSTMQAPI** in *USER.ZCEE.CNTL* will invokes a POST HTTP method which will put a message on the queue. Submit this job for execution and observe the results. The POST method for this API does not return a response message.

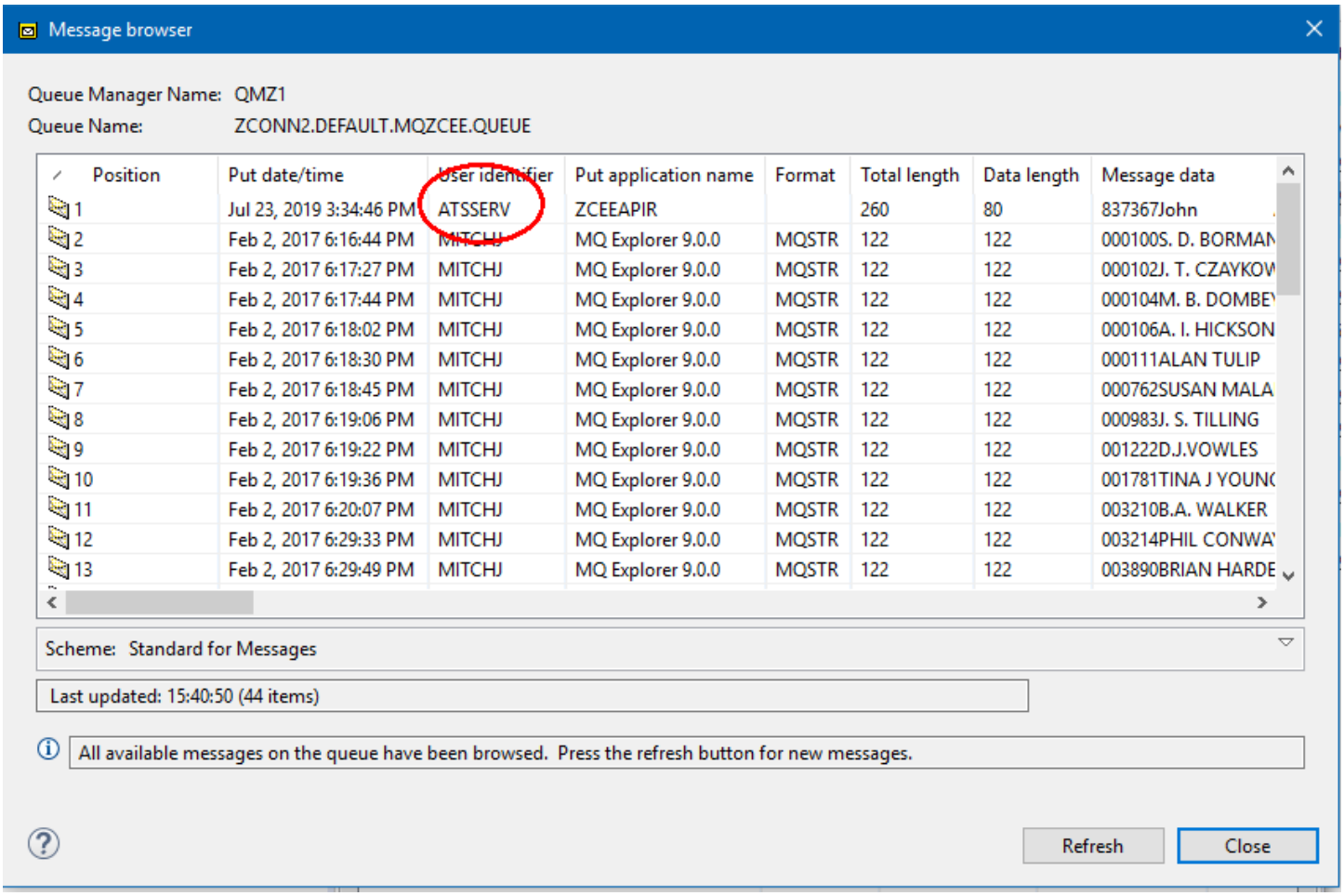
```
HTTP CODE: 0000000200
```

___6. Confirm by opening the *MQExplorer* icon on the desktop and browsing the messages in the *ZCONN2.DEFAULT.MQZCEE.QUEUE* queue.

- Select *QMZ1* under **Queue Managers** and right mouse button click
- Select the *Connect* option.
- Once connected, expand the *Queues* folder and select *ZCONN2.DEFAULT.MQZEE.QUEUE* and right mouse button click and select the *Browse Messages* option.



___7. Your see a list of the messages currently in the queue, something like the list below:



Summary

You have use the z/OS Connect build toolkit to generate an API requester archive file and the COBOL copy books that must be included in the COBOL application program when accessing the API requester archive file in a z/OS Connect server. The COBOL applications have been compiled and link edited, and the target API has been tested using various RESTful methods

Mitch Johnson (mitchj@us.ibm.com)