IBM z/OS Connect EE V3.0

Developing RESTful APIs for MQ Services

WSC Wildfire Team
IBM z Systems

Lab Version Date: June 20, 2019

Table of Contents

> **Important:** On the desktop there is a file named *Developing APIs 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.

Overview

Important – You do not need any skills with MQ to perform this exercise. Even if MQ is not relevant to your current plans performing this exercise will give additional experience using the API Toolkit with developing APIs.

The objective of these exercises is to gain experience with working with z/OS Connect EE and the API Toolkit. These two products allow the exposure of z/OS resources to JSON clients. More in-depth information about the customization of z/OS Connect EE, z/OS Connect EE security, the use of the API Toolkit and other topics is provided by the 1-day *ZCONNEE – z/OS Connect Workshop*. For information about scheduling this workshop in your area contact your IBM representative.

If you have completed either the developing APIs exercise for Db2 or MVS Batch you can start with section *MQ for z/OS Service Provider for z/OS Connect* on page 7.

General Exercise Information and Guidelines

- ✓ This exercise is based on the MQ Provider supplied by z/OS Connect EE. There is another exercise which is based on the MQ Provider supplied by the IBM MQ product.

- ✓ This exercise requires using z/OS user identity *USER1*. The password for this user will be provided by the lab instructor.

- ✓ Any time you have any questions about the use of IBM z/OS Explorer, 3270 screens, features or tools do not hesitate to ask the instructor for assistance.

- ✓ Text in **bold** and highlighted in <mark>yellow</mark> in this document should be available for copying and pasting in a file named *Development APIs CopyPaste* file on the desktop.

Connect to the z/OS Connect EE Server

Begin by establishing a connection to your z/OS Connect server 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 clicking the mouse button and then selecting the *Open* option.
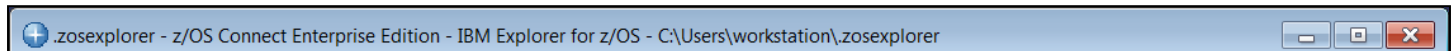
___1. On the workstation desktop, locate the *z/OS Explorer* icon and double click on it to open the Explorer.

___2. You will be prompted for a workspace:
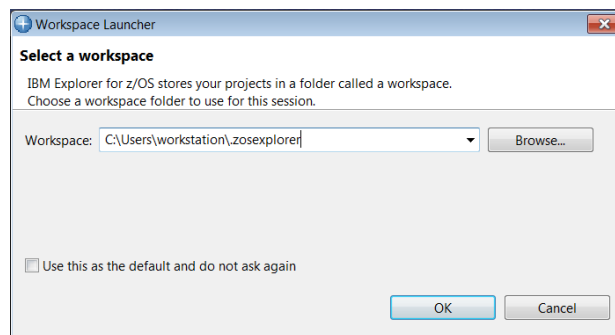
Take the default value by clicking **OK**.

___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.
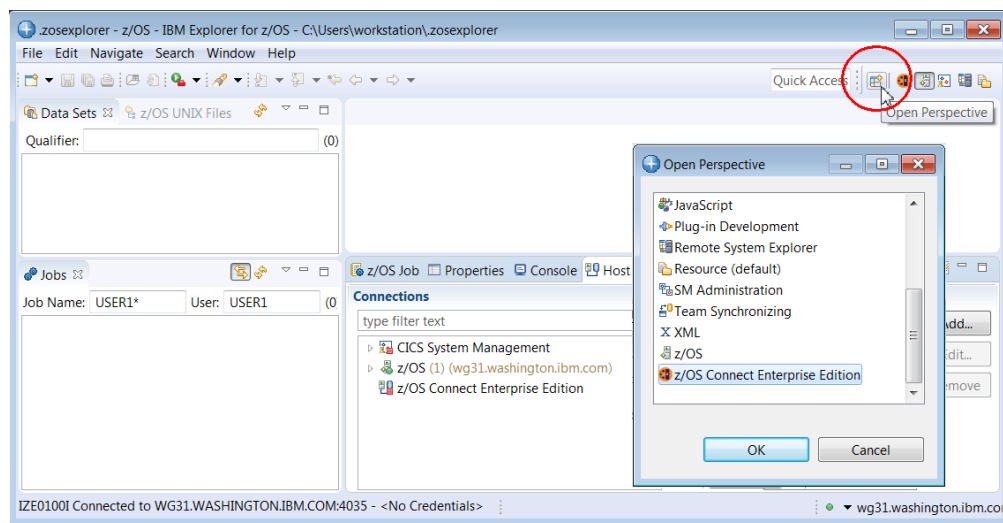
___5. To add a connection to the z/OS Connect Server select *z/OS Connect Enterprise Edition* connection in the *Host connection*s tab in the lower view and then click the **Add** button.

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, *Host Connections* and *Project Explorer* 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 Host Connections, Properties, Progress* and *Problems*. 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.

___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.

Mitch Johnson (mitchj@us.ibm.com)

MQ for z/OS Service Provider for z/OS Connect

This section provides an opportunity to develop REST interfaces to MQ queues (and topics) using the *MQ for z/OS Service Provider for z/OS Connect* feature which is shipped with z/OS Connect EE V3.0.21. This feature provides support for two types of services. The first service types are two-way services where the REST **POST** method is used to put a message on queue. A back-end application processes the message and provides a response message on a reply queue. The second service types are one-way services where REST **POST** and **GET** methods are used put and get messages from a single queue.

Generating the Service Archive (SAR) files

The first step you are required to perform is to generate the service archive (SAR) files using the z/OS Connect EE build toolkit.

Generate a service archive file for the two-way service

A *z/OS Connect EE Build Toolkit* properties file was created for this service with the contents below:

```
provider=mq
name=miniloan
version=1.0
description=MQ two-way
destination=jms/request
replyDestination=jms/response
connectionFactory=jms/qmgrCf
language=COBOL
requestStructure=miniloan.cpy
responseStructure=miniloan.cpy
operationName=miniloan
waitInterval=10
```

___ 1. Open a DOS command prompt by opening the *Command Prompt* icon on the desktop.

___ 2. Use the change directory to go to directory C:\z\MQLab, e.g. ***cd C:\z\MQLab***.

___ 3. Generate the *miniloan.sar* file by invoking the build toolkit for the *miniloan* service by entering command **miniloan**. The contents of *miniloan.bat* are shown below:

```
c:\z\zconbt\bin\zconbt.bat -p=miniloan.properties -f=miniloan.sar
```

Mitch Johnson (mitchj@us.ibm.com)

This is what your DOS session should look like.

```
c:\>cd \z\MQlab

c:\z\MQLab>miniloan

c:\z\MQLab>c:\z\software\zconbt\bin\zconbt.bat -p=miniloan.properties -f=miniloan.sar
BAQB0000I: z/OS Connect Enterprise Edition Build Toolkit Version 1.1.
BAQB0001I: Creating service archive from configuration file miniloan.properties.
BAQB1022I: IBM MQ for z/OS plugin for IBM z/OS Connect EE V3.0 build toolkit, code level is
3.0.21.0(20190514-1637).
BAQB0002I: Successfully created service archive file miniloan.sar.
```

_____ 4. Deploy the *miniloan* service by using the z/OS Connect EE RESTful administrative interface to deploy the service archive file by using the cURL command with a POST method by entering command **deployMiniloan** (the curl command is in the bat file deployMiniloan).

```
c:\z\MQLab>curl -X POST --user Fred:fredpwd --data-binary @miniloan.sar --header
"Content-Type: application/zip"  --insecure https://wg31:9453/zosConnect/services
{"zosConnect":{"serviceName":"miniloan","serviceDescription":"MQ two-
way","serviceProvider":"IBM MQ for
z/OS","serviceURL":"https://wg31:9453/zosConnect/services/miniloan","serviceInvoke
URL":"https://wg31:9453/zosConnect/services/miniloan?action=invoke","dataXformProv
ider":"zosConnectXform-1.0",
"serviceStatus":"Started"},"miniloan":{"creationTime":"2019.06.19 09:16:20 EDT",
"destination":"jms/request","replySelection":"msgIDToCorrelID",
"useCallerPrincipal":false,"waitInterval":10,"ccsid":37,"createdBy":"buildToolkit"
,"mqSarFileVersion":1,"connectionFactory":"jms/qmgrCf","persistence":false,"plugin
Level":"3.0.21.0(20190514-1637)","id":"miniloan","expiry":-
```

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

> **Tech-Tip:** If a service needs to be redeployed the service must be first stopped and then deleted. The cURL command with a PUT method can be used to stop the service:
>
> *curl -X PUT --user Fred:fredpwd --insecure*
> *https://wg31.washington.ibm.com:9453/zosConnect/services/miniloan?status=stopped*
>
> And the cURL command with a DELETE method can be used to delete the service:
>
> *curl -X DELETE --user Fred:fredpwd --insecure*
> *https://wg31.washington.ibm.com:9453/zosConnect/services/miniloan*
>
> If this is not done the service cannot be redeployed.

Mitch Johnson (mitchj@us.ibm.com)

> **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 response.json*
>
>
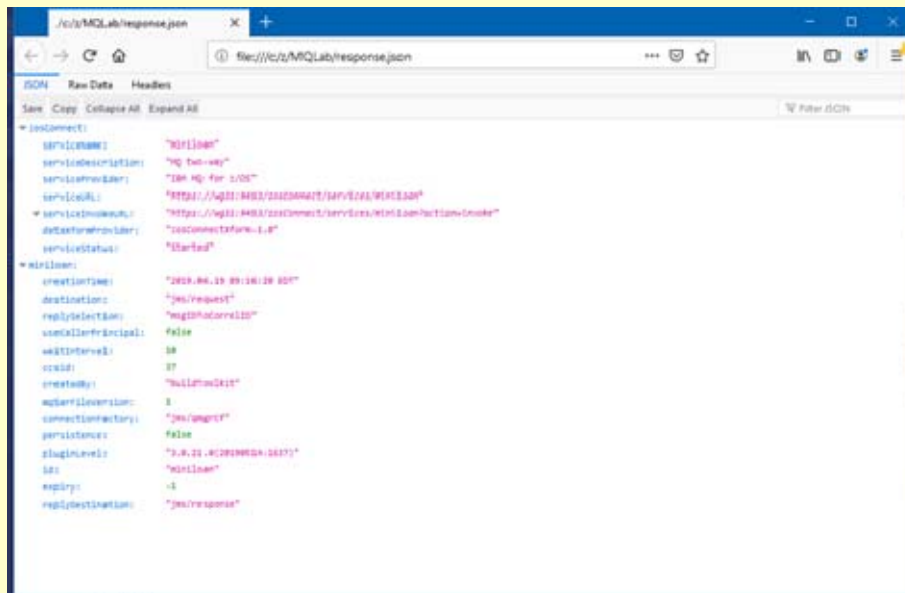
Generate a service archive file for the one-way put service

A *z/OS Connect EE Build Toolkit* properties file was created for this service with the contents below:

```
provider=mq
name=mqPut
version=1.0
description=MQ put one-way
destination=jms/default
connectionFactory=jms/qmgrCf
language=COBOL
requestStructure=fileareq.cpy
operationName=mqPut
messagingAction=mqput
```

___ 5. Generate the *mqput.sar* file by invoking the build toolkit for the *mqput* service by entering command **mqput**. The contents of *mqput.bat* are shown below:

```
c:\z\zconbt\bin\zconbt.bat -p=mqput.properties -f=mqput.sar
```

Mitch Johnson (mitchj@us.ibm.com)

This is what your DOS session should look like.

```
c:\z\MQLab>mqput

c:\z\MQLab>c:\z\software\zconbt\bin\zconbt.bat -p=mqput.properties -f=mqput.sar
BAQB0000I: z/OS Connect Enterprise Edition Build Toolkit Version 1.1.
BAQB0001I: Creating service archive from configuration file mqput.properties.
BAQB1022I: IBM MQ for z/OS plugin for IBM z/OS Connect EE V3.0 build toolkit, code level is
3.0.21.0(20190514-1637).
BAQB0002I: Successfully created service archive file mqput.sar.
```

_____ 6. Deploy the *mqput* service by using the z/OS Connect EE RESTful administrative interface to deploy the service archive file by using the cURL command with a POST method by entering command ***deployMqPut*** (the curl command is in the bat file *deployMqPut*).

```
:\z\MQLab>curl -X POST --user Fred:fredpwd --data-binary @mqput.sar --header
"Content-Type: application/zip"  --insecure https://wg31:9453/zosConnect/services
{"zosConnect":{"serviceName":"mqPut","serviceDescription":"MQ put one-
way","serviceProvider":"IBM MQ for
z/OS","serviceURL":"https://wg31:9453/zosConnect/services/mqPut","serviceInvokeURL
":"https://wg31:9453/zosConnect/services/mqPut?action=invoke","dataXformProvider":
"zosConnectXform-1.0",
"serviceStatus":"Started"},"mqPut":{"creationTime":"2019.06.19 11:16:56 EDT",
"createdBy":"buildToolkit","mqSarFileVersion":1,"connectionFactory":"jms/qmgrCf","
destination":"jms/default","messagingAction":"mqput","persistence":false,
"useCallerPrincipal":false,"pluginLevel":"3.0.21.0(20190514-1637)",
"id":"mqPut","expiry":-1,"ccsid":37}}
```

Generate a service archive file for the one-way get service

A *z/OS Connect EE Build Toolkit* properties file was created for this service with the contents below:

```
provider=mq
name=mqGet
version=1.0
description=MQ Get one-way
destination=jms/default
connectionFactory=jms/qmgrCf
waitInterval=10000
language=COBOL
responseStructure=filearsp.cpy
operationName=mqGet
messagingAction=mqget
```

___ 7. Generate the *mqget.sar* file by invoking the build toolkit by entering command ***mqget.*** The contents of *mqget.bat* are shown below:

```
c:\z\zconbt\bin\zconbt.bat -p=mqget.properties -f=mqget.sar
```

Mitch Johnson (mitchj@us.ibm.com)

This is what your DOS session should look like.

```
c:\z\MQLab>mqget

c:\z\MQLab>c:\z\software\zconbt\bin\zconbt.bat -p=mqput.properties -f=mqput.sar
BAQB0000I: z/OS Connect Enterprise Edition Build Toolkit Version 1.1.
BAQB0001I: Creating service archive from configuration file mqget.properties.
BAQB1022I: IBM MQ for z/OS plugin for IBM z/OS Connect EE V3.0 build toolkit, code level is
3.0.21.0(20190514-1637).
BAQB0002I: Successfully created service archive file mqget.sar.
```

_____ 8. Deploy the *mqput* service by using the z/OS Connect EE RESTful administrative interface to deploy the service archive file by using the cURL command with a POST method by entering command ***deployMqPut*** (the curl command is in the bat file *deployMqPut*).

```
c:\z\MQLab>curl -X POST --user Fred:fredpwd --data-binary @mqget.sar --header
"Content-Type: application/zip"  --insecure https://wg31:9453/zosConnect/services
{"zosConnect":{"serviceName":"mqGet","serviceDescription":"MQ Get one-
way","serviceProvider":"IBM MQ for
z/OS","serviceURL":"https://wg31:9453/zosConnect/services/mqGet","serviceInvokeURL
":"https://wg31:9453/zosConnect/services/mqGet?action=invoke","dataXformProvider":
"zosConnectXform-1.0",
"serviceStatus":"Started"},"mqGet":{"creationTime":"2019.06.19 11:25:46 EDT",
"createdBy":"buildToolkit","mqSarFileVersion":1,"connectionFactory":"jms/qmgrCf","
destination":"jms/default","messagingAction":"mqget","useCallerPrincipal":false,
"pluginLevel":"3.0.21.0(20190514-1637)",
"id":"mqGet","waitInterval":10000,"ccsid":37}}1637)","id":"mqPut","expiry":-1,
"ccsid":37}}.
```

_____ 9. Expanding the Services folder in the IBM z/OS Explorer shows the 3 MQ services have been installed and started.
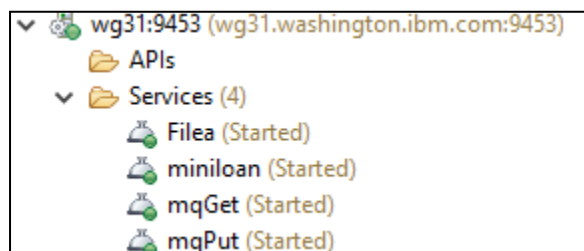
Summary
The COBOL copy books representing the layout of the messages files along with the properties provided in each service's properties files have been used by the z/OS Connect EE build toolkit to generate service archive files for each MQ service

Mitch Johnson (mitchj@us.ibm.com)

Create the MQ API project

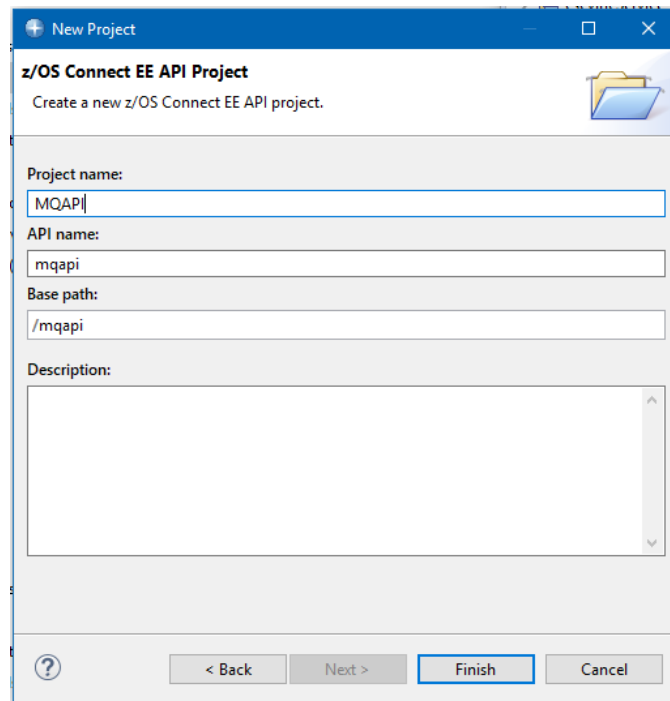In this section an API that uses both MQ two-way service and a one-way service will be developed.

___1. In the *z/OS Connect Enterprise Edition* perspective of the z/OS Explorer create a new API project by clicking the right mouse button and selecting *New → Project*:
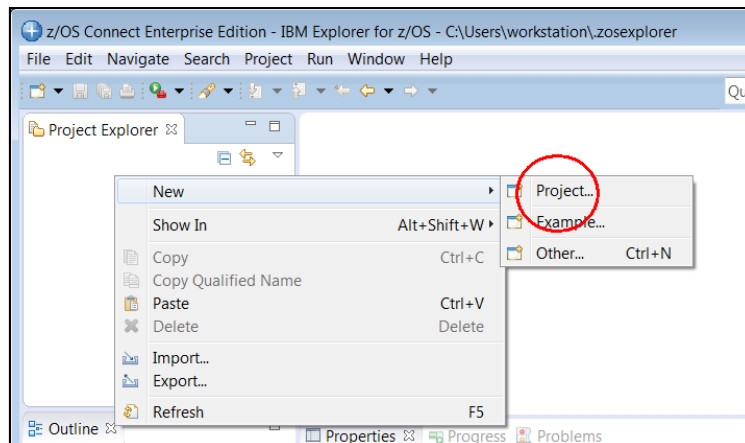
___2. In the *New Project* screen, scroll down and open the *z/OS Connect Enterprise Edition* folder and select *z/OS Connect EE API Project* and then click the **Next** button.
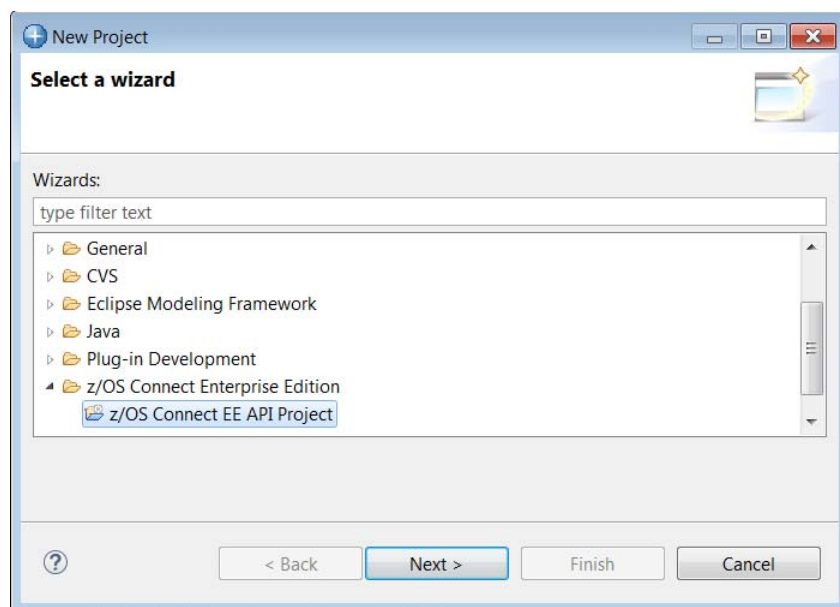
___3. Enter *MQAPI* for the *Project name.* Be sure the *API name* is set to *mqapi* and the *Base path is* set to */mqapi*. Click **Finish** to continue.

> **Important:** The values are somewhat arbitrary, but they do relate to later tasks. If you use the values and cases as supplied, then the subsequent commands and the use of subsequent URLs will work seamlessly.
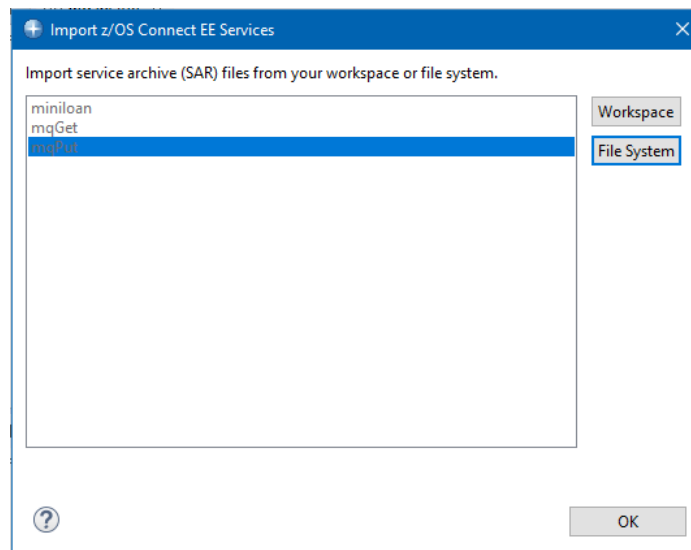
Import the SAR files

___1. In the z/OS Explorer in the *z/OS Connect Enterprise Edition* perspective in the the *Project Explorer* view (upper left), right-click on the *MQAPI* project, then select *z/OS Connect EE* and then *Import z/OS Connect EE Services* (see below):
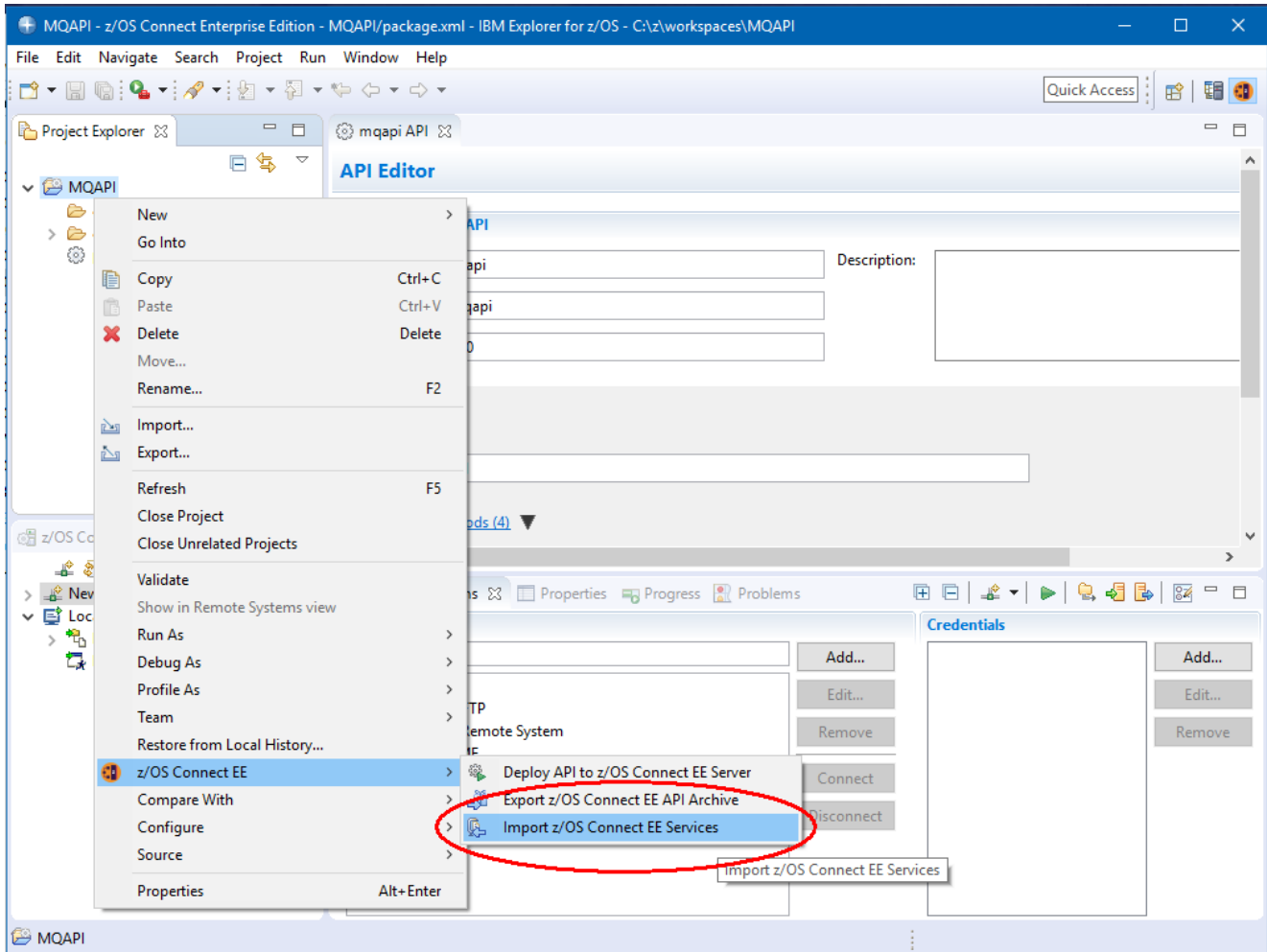
___2. In the *Import z/OS Connect EE Services* window click on the **File System** button and navigate to directory *c:\z\MQLab*. Select the three SAR files and click on the **Open** button:

```
Import Services                                                            ✕

The following services are new to this project:

Name                          Description
miniloan                      MQ two-way
mqGet                         MQ Get one-way
mqPut                         MQ put one-way




?                                                    [  OK  ]   [ Cancel ]
```
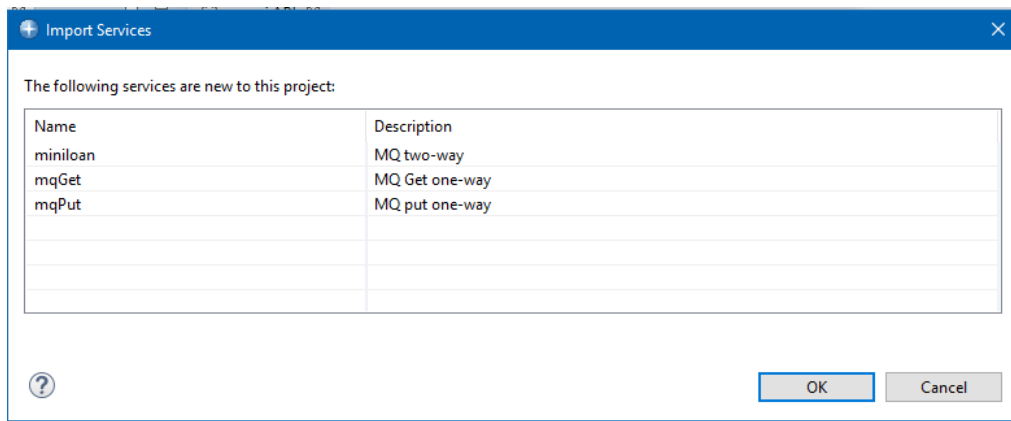
___3. The three service archive files should appear in the *Import z/OS Connect EE Services* screen. Click the **OK** button to import them into the workspace.

```
Import z/OS Connect EE Services                                           ✕

Import service archive (SAR) files from your workspace or file system.

 miniloan                                          [ Workspace ]
 mqGet
 mqPut                                             [ File System ]











?                                                            [  OK  ]
```

___4. In the *Project Explorer* view (upper left), expand the *services* folder to see the the imported service:

```
Project Explorer ⊠              ▭ ▭

                              ⊟ ⇆   ▽
∨ 📂 MQAPI
    📂 api
  > 📂 api-docs
  ∨ 📂 services
      > 📂 miniloan
      > 📂 mqGet
      > 📂 mqPut
    ⚙ package.xml
```

Mitch Johnson (mitchj@us.ibm.com)

.
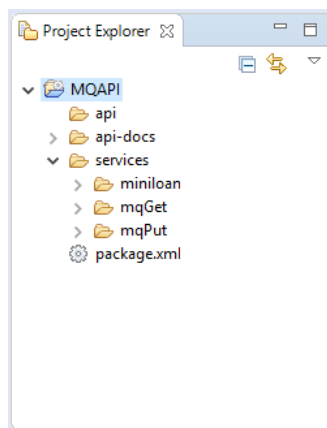
Compose the API for two-way MQ service

___1. A *MQ for z/OS Service Provider for z/OS Connect* in a two-way service requires only one method, the **POST** method ('put' a message on a queue). The **PUT** method is not supported by the *MQ for z/OS Service provider for z/OS Connect* so it should be deleted**.** The **GET** and **DELETE** methods are also not needed for a two-way service so they can be deleted also.   The view may need to be adjusted by dragging the view boundary lines.
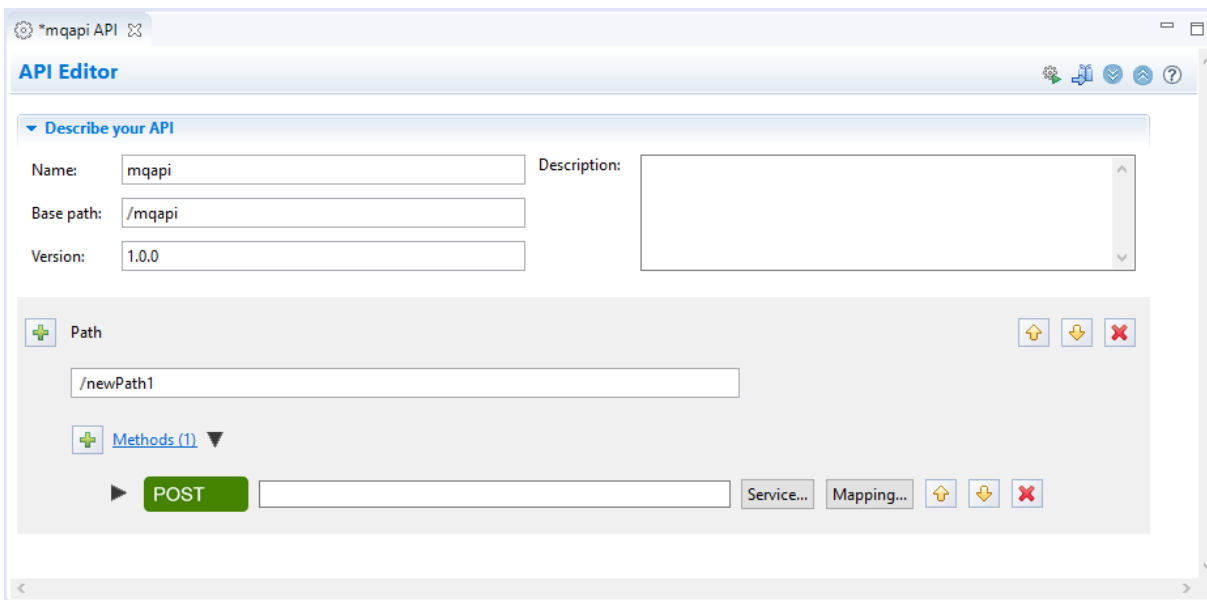
___2. Start by entering */loan* as the *Path* string in the *z/OS Connect EE API Editor* view. Click on the **Service** button to the right of the **POST** method. Then on the *Select a z/OS Connect EE Service* window select *miniloan* and click **OK**. This will populate the field to the right of the method.
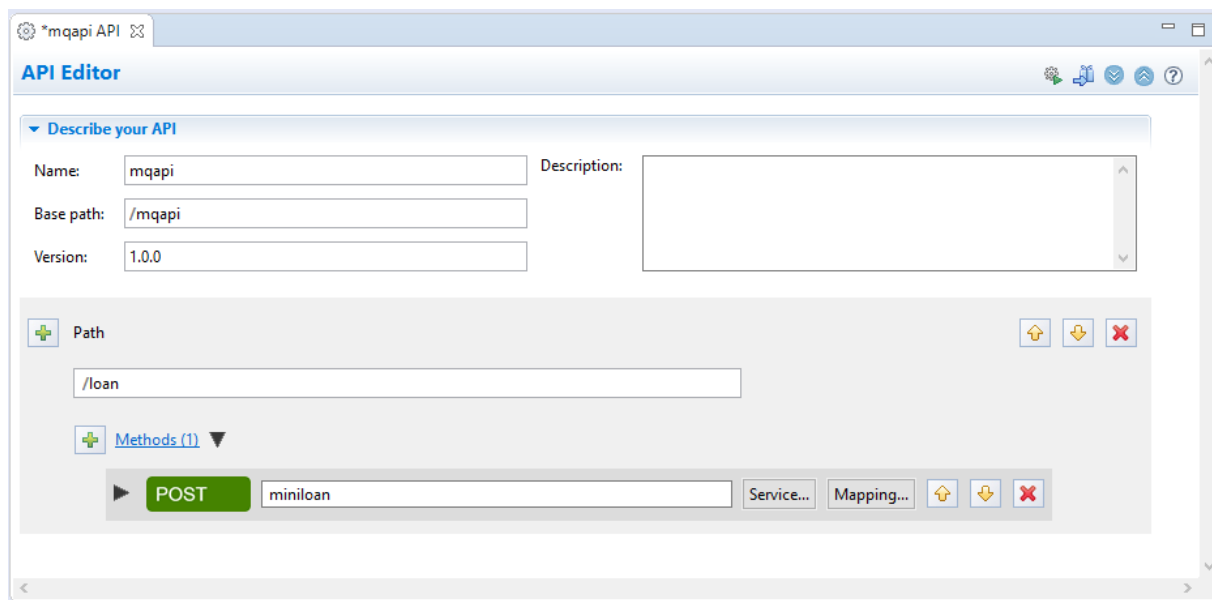
___3. Next, click on the **Mapping** button beside the **POST** method and then select *Open Request Mapping*:
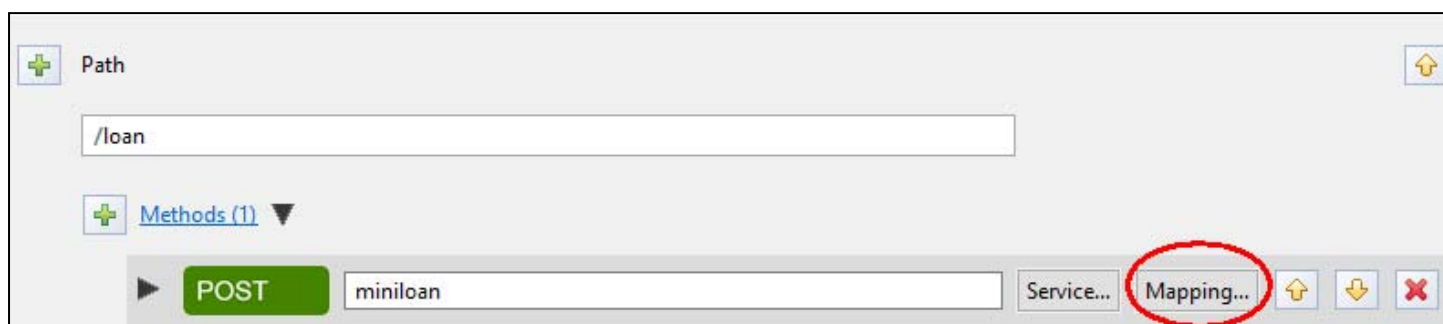
___4. In the mapping view that opens, go to the right side of the mapping (which represents the COPYBOOK fields), and click the little + signs to expand *miniloan_commarea*. You should see fields that correspond to the fields defined in the original COBOL copy book *miniloan.cpy* (see below).
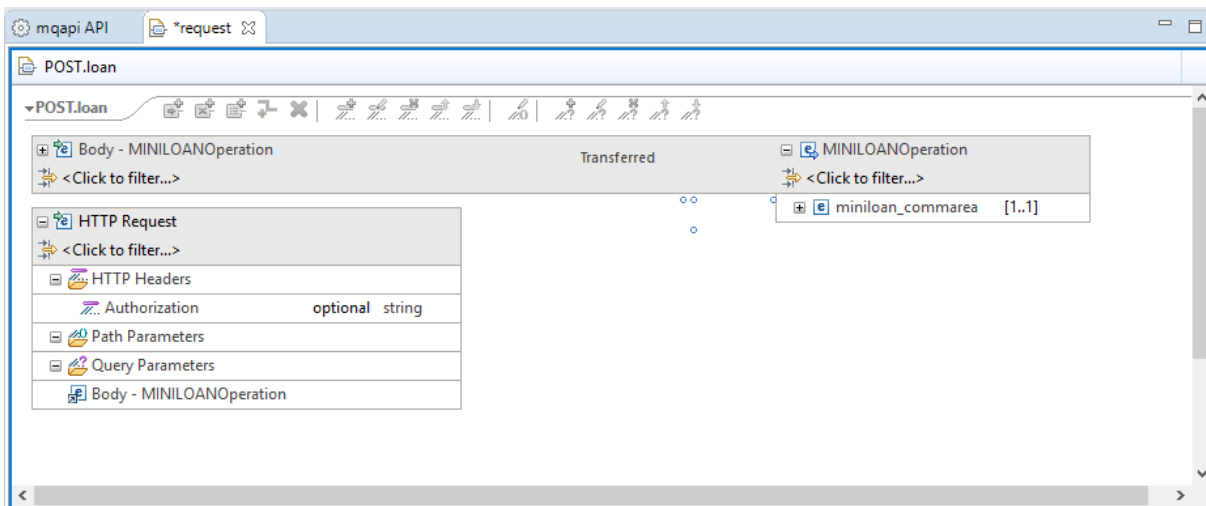
___5.

```
01 MINILOAN-COMMAREA.
     10 name pic X(20).
     10 creditScore pic 9(18).
     10 yearlyIncome pic 9(18).
     10 age pic 9(10).
     10 amount pic  9(18).
     10 approved pic X.
        88 BoolValue value 'T'.
     10 effectDate pic X(8).
     10 yearlyInterestRate pic S9(5).
     10 yearlyRepayment pic 9(18).
     10 messages-Num pic 9(9).
     10 messages pic X(60) occurs 1 to 10 times.
```

Use the slider bar to fully expose the *miniloan_commarea* structure. Right click the mouse button on *messages* and select the *Add Remove transform* option from the list of options. Repeat these steps for field *messages_Num*. Next select the *approved* field and right click the mouse button. Use the *Add Assign transform* option to enter an **F** as the value of this field in the *Properties* tab. Finally select the *yearlyInterestRate* field and right click mouse button. Use the *Add Assign Transform* option to set the value of this field to **00005** (see below). When complete the mapping should look like this:
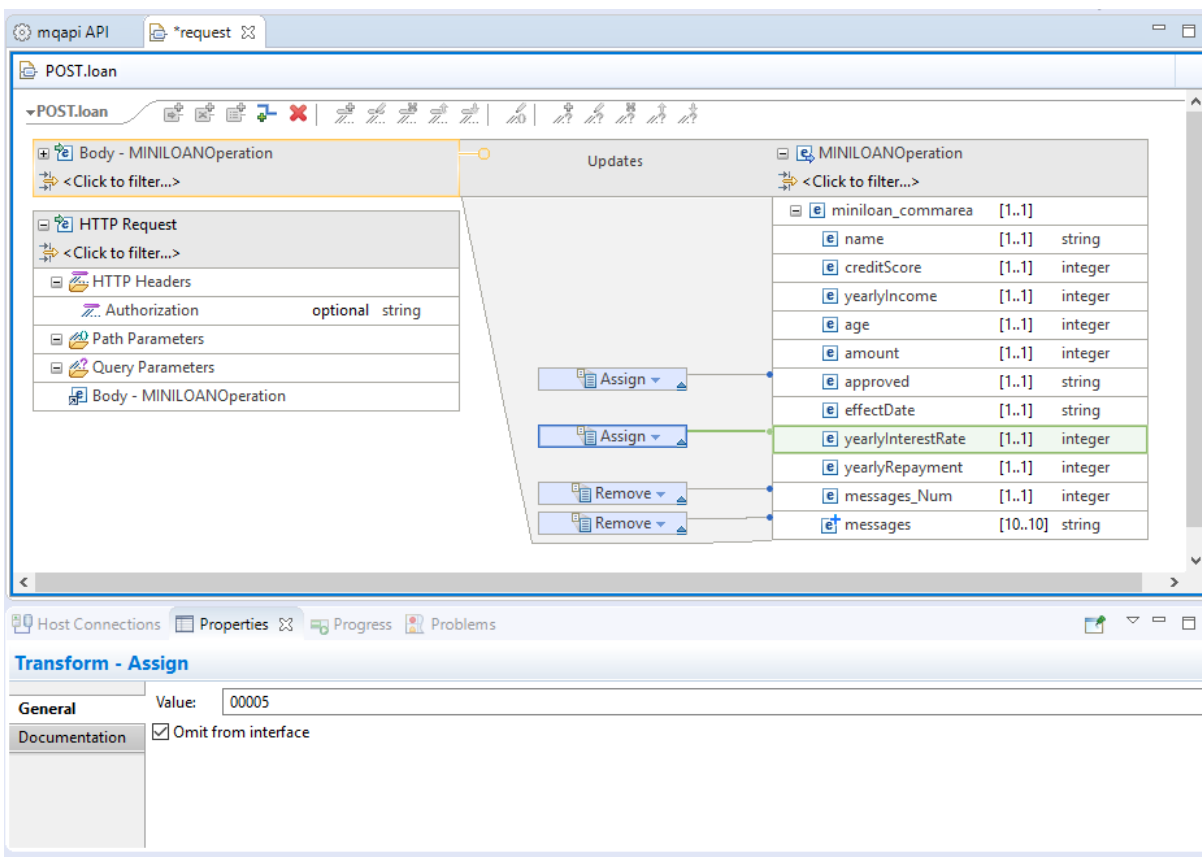
This action has omitted *messages* and *messages_Num* from the request interface and set defaults values for *approved* and *yearlyInterestRate*. Assigning values also removes these fields from the request interface.

___6. Use the **Ctrl-S** key sequence to save the changes.

Fields can be removed from the response message (i.e. the reply from the application) by using the response mapping. At this time, all fields should be returned so no response mapping changes are required. If time allows change the response mapping to remove a field or two and test the results.

Summary

You created the API, which just a base path and with the **POST** method along with the request and response mapping required for this method. This API will now be deployed into z/OS Connect EE V3.0.

Deploy the two-way API to a z/OS Connect EE Server

Review the z/OS Connect server.xml updates required for the *MQ for z/OS Connect Service Provider for z/OS Connect* before deploying the API.

___1. Suport for this API is added to the z/OS Connect server's *server.xml* file by the including the *mq.xml* file (see below).

```
<server description="MQ Service Provider">

 <featureManager>
        <feature>zosconnect:mqService-1.0</feature>          1
</featureManager>

 <variable name="wmqJmsClient.rar.location"
         value="/shared/mqm/V9R0M1/java/lib/jca/wmq.jmsra.rar"/>
 <wmqJmsClient nativeLibraryPath="/shared/mqm/V9R0M1/java/lib"/>

  <connectionManager id="ConMgr1" maxPoolSize="5"/>

<jmsConnectionFactory id="qmgrCf" jndiName="jms/qmgrCf"
        connectionManagerRef="ConMgr1">
        <properties.wmqJMS transportType="BINDINGS"          2
               queueManager="QMZ1" />
</jmsConnectionFactory>

 <jmsQueue id="q1" jndiName="jms/default">
     <properties.wmqJms
         baseQueueName="ZCONN2.DEFAULT.MQZCEE.QUEUE"
         CCSID="37"/>
 </jmsQueue>

<jmsQueue id="request" jndiName="jms/request">
     <properties.wmqJms
         baseQueueName="ZCONN2.TRIGGER.REQUEST"
         targetClient="MQ"                                   3
         CCSID="37"/>
 </jmsQueue>

<jmsQueue id="response" jndiName="jms/response">
      <properties.wmqJms
         baseQueueName="ZCONN2.TRIGGER.RESPONSE"
         targetClient="MQ"                                   4
         CCSID="37"/>
  </jmsQueue>
</server>
```

1. The *featureManager* element identifies the Liberty features required by the *MQ for z/OS Service Provider for z/OS Connect.*

2. The *jmsConnectionFactory* element associates the JMS connection factory(*jndiName*) with the target queue manager and details on how to connect to this queue manager.

3. The jmsQueue elements provide details that associate the JMS destination (*jndiName*) with the request queue (*baseQueueName*) and its JMS/MQ properties.

4. The jmsQueue elements provide details that associate the JMS destination (*jndiName*) with the resonse queue (*baseQueueName*) and its JMS/MQ properties.

___2. In the *Project Explorer* view (upper left), right-mouse click on the *MQAPI* folder, then select *z/OS Connect EE→ Deploy API to z/OS Connect EE Server*.
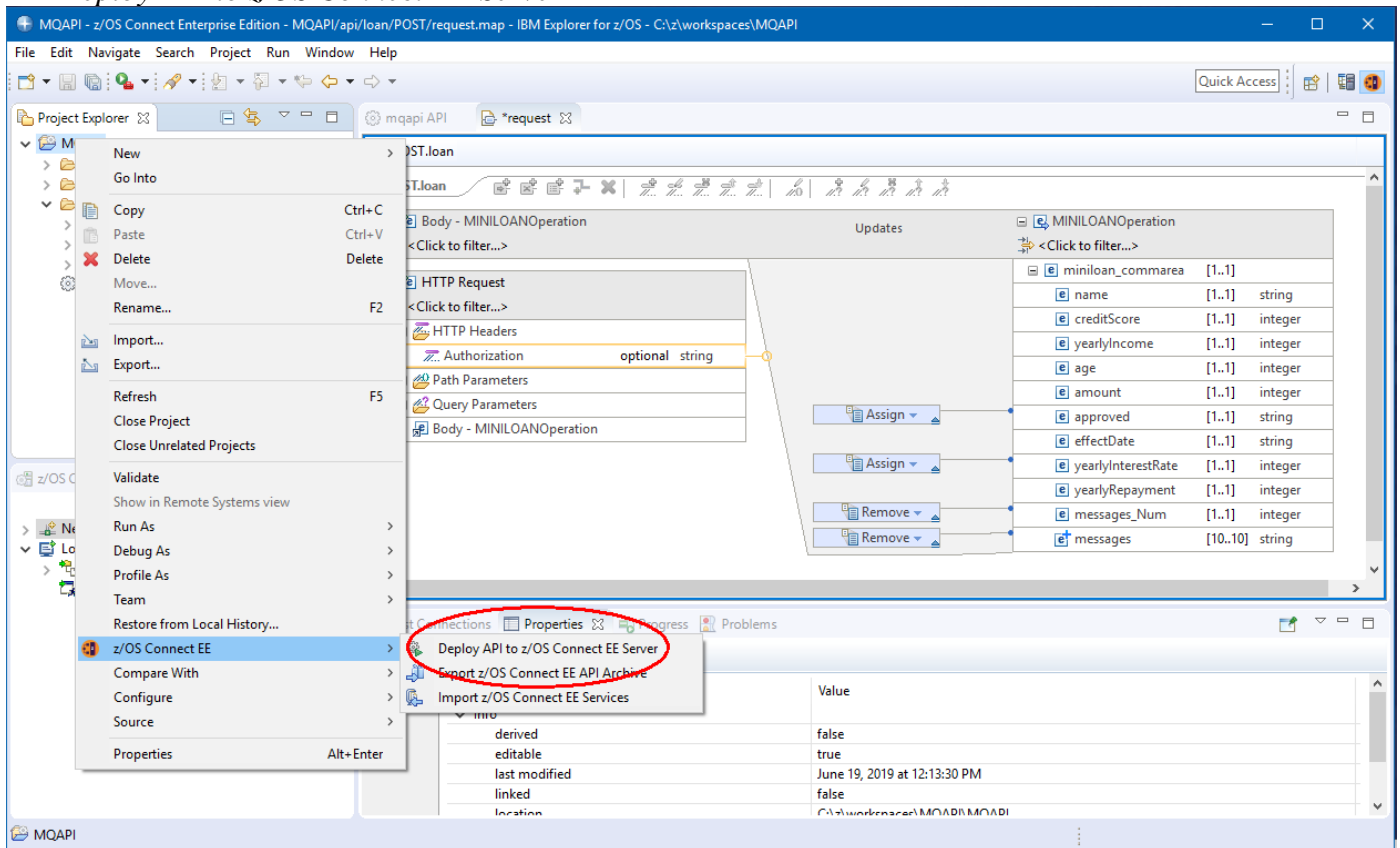
___3. Since z/OS Explorer is connected to only one z/OS Connect server there is only one choice (*wg31:9453*). If z/OS Explorer had multiple host connections to z/OS Connect servers then the pull-down arrow would allow a selection to which server to deploy. Click **OK** on this screen to continue.
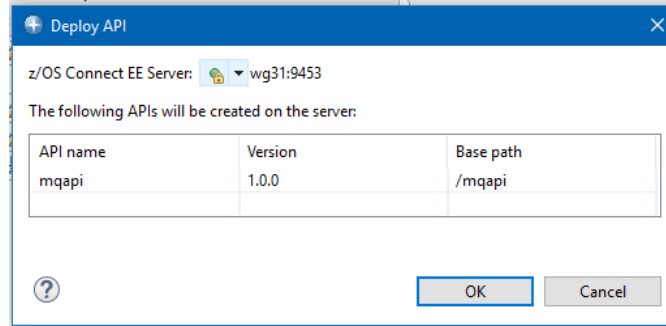
___4. The API artifacts will be transferred to z/OS and copied into the */var/ats//zosconnect/servers/zceesrv1/resources/zosconnect/apis* directory.
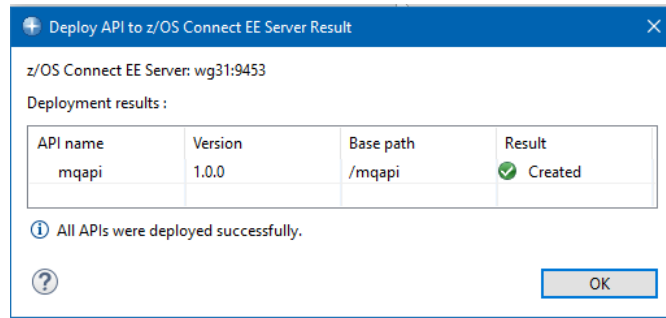
Test the MQ Miniloan Reply/Response APIs

The application used to the test the MQ API is a CICS application that uses rules for approving loan requests. The CICS transaction that invokes the application is started when a message is written to a request queue. The arrival of a message triggers the CICS transaction which starts a program which reads the message from the request queue. The application uses the information in the message to determine if a loan can be approved. The results are returned in a message in a reply queue including the explanation if the loan is denied.

The rules for rejecting a loan can be for any one of the following:

- If the credit score of the borrower is less than 300.
- If the yearly repayment amount is more than 30% of the borrower's income.
- If the income of the borrower is less than $24,000.
- If the age of the borrower is more than 65 years.
- The loan amount is more than $1,000,000.

.

___1. Enter URL ***https://wg31.washington.ibm.com:9453/zosConnect/apis*** in the Firefox browser and you should see something like the window below. The API *miniloan* is now displayed. This is because this API was just deployed to this server.
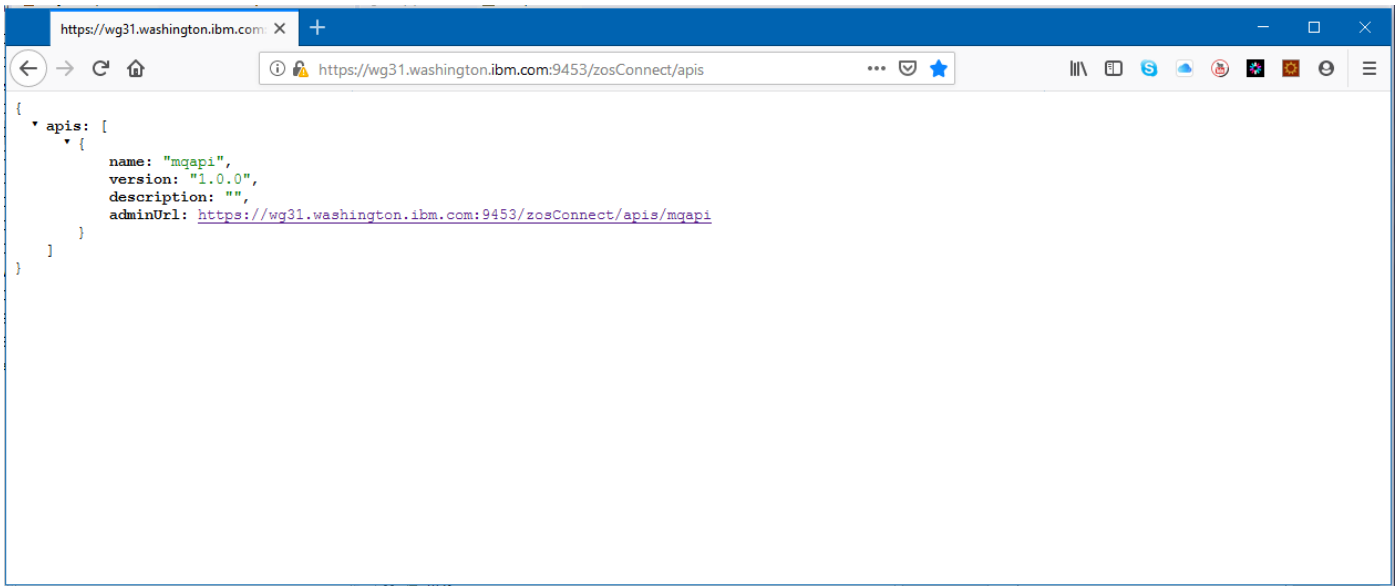
> **Tech Tip:** You may be challenged by Firefox because the digital certificate used by the Liberty z/OS server is self-signed Click the **Add Exception** button to continue. If the **Add Exception** button is not displayed click the **Advanced** button. Then click on the **Confirm Security Exception** button. Next you may see a prompt you for a userid and password.   If you do see the prompt, enter the username ***Fred*** and password **fredpwd** (case matters) and click **OK**. Remember we are using basic security and this is the user identity and password defined in the server.xml file.

> **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.

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

Mitch Johnson (mitchj@us.ibm.com)
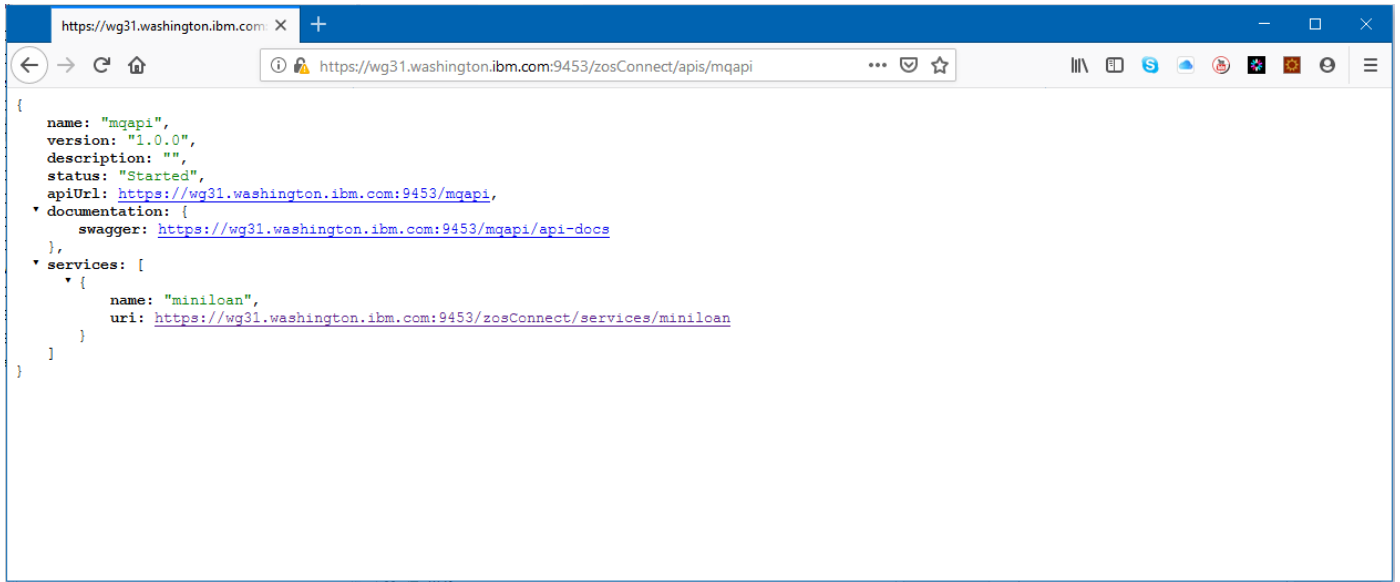
___3. Finally click on the *swagger* URL and you should see the Swagger document associated with this API.
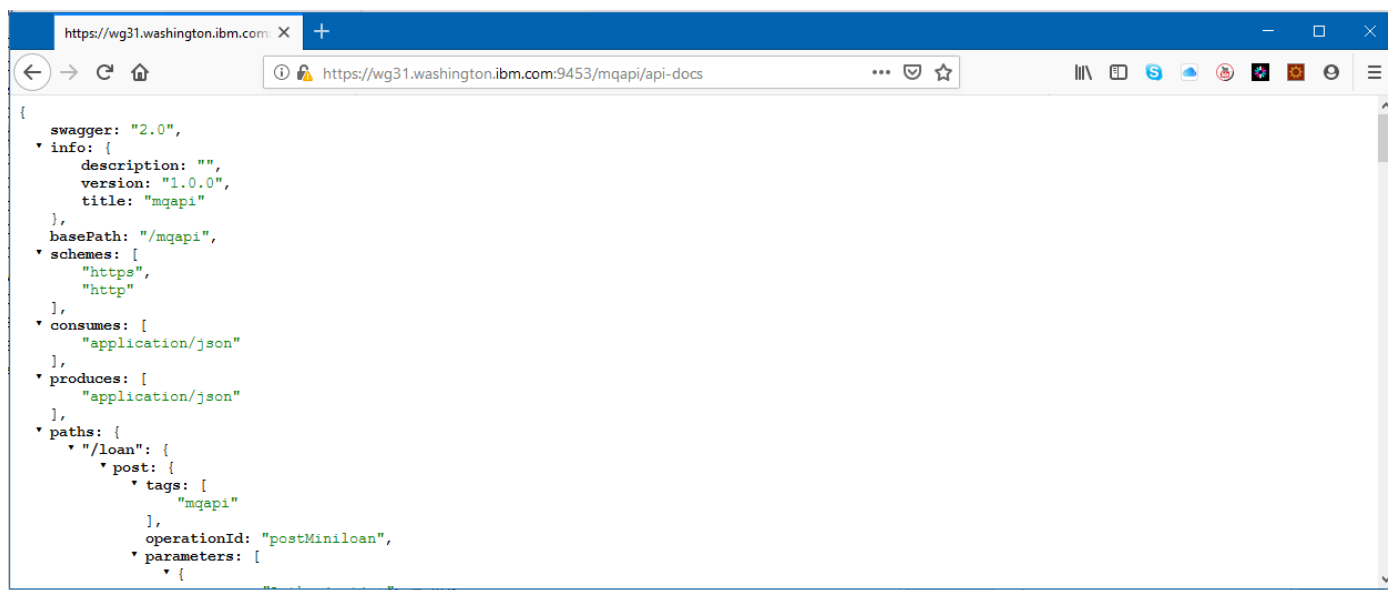
___4. Explore this Swagger document and you will see the results of the request and response mapping performed earlier. This document can be used by a developer or other tooling to develop REST clients for this specific API.

___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.
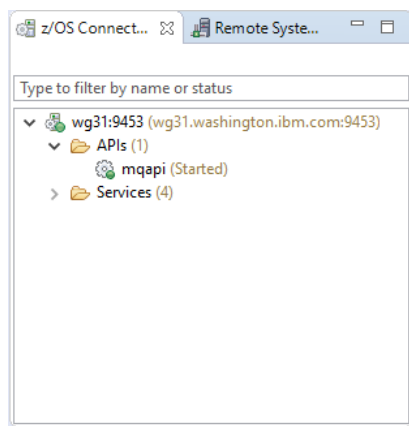
___6. Right click the mouse button on *mqapi* and select *Open in Swagger UI*. Click OK if an informational
prompt appears. This will open a Firefox window showing a *Swagger* test client (see below).
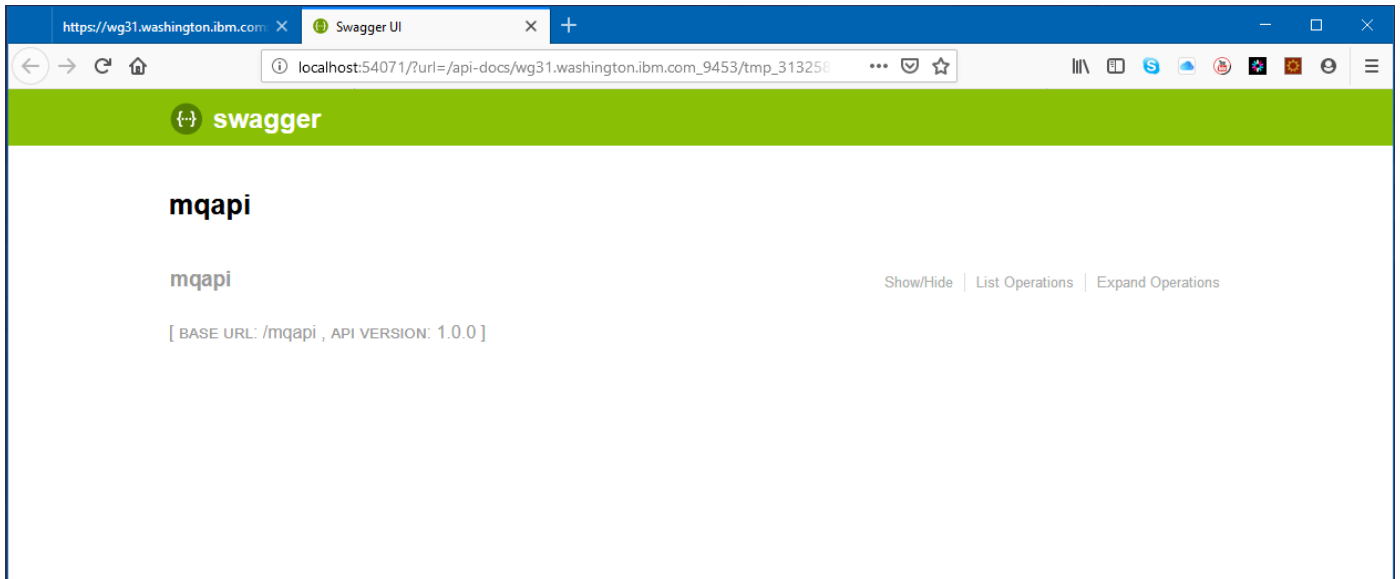
___7. Click the *List Operations* and the browser should show a list of the available HTTP methods like this:
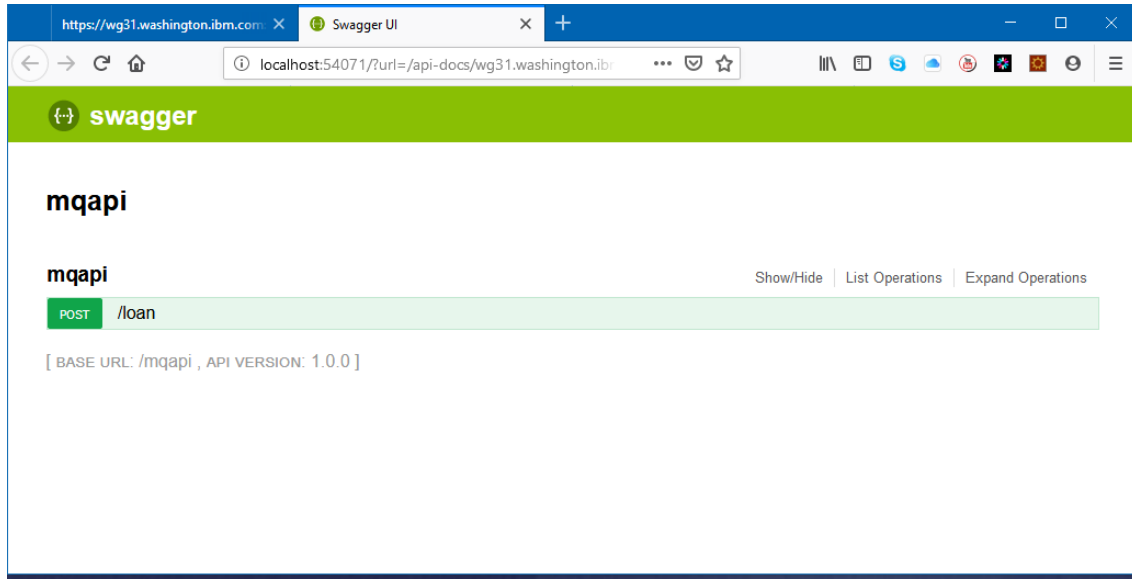
___8. Expand the *Post* method by clicking on the path beside it (e.g. */loan)* and scroll down until the method *Parameters* are displayed as shown below:

Mitch Johnson (mitchj@us.ibm.com)
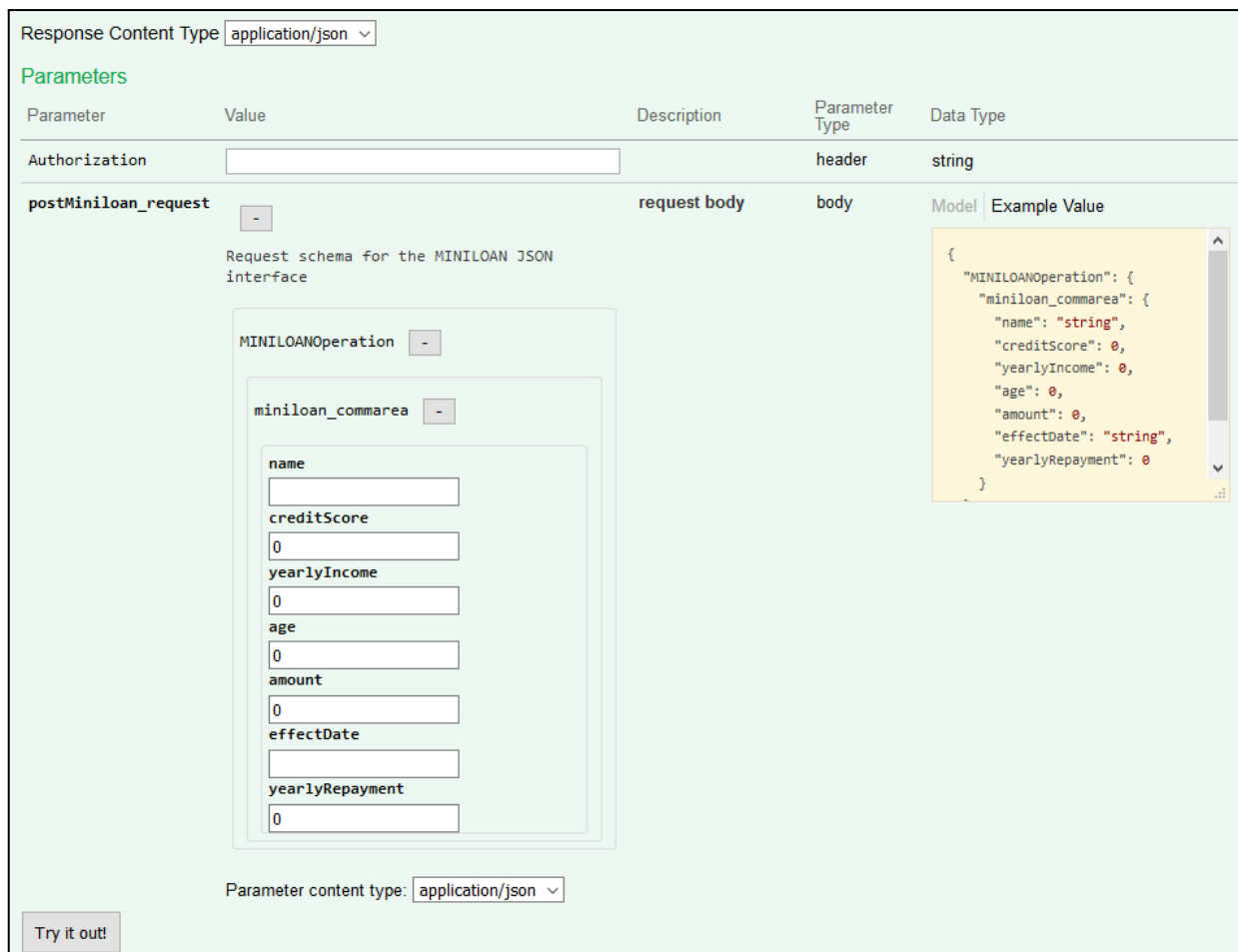
___9. Enter *John* in the area under *name*, *100* in the area under *creditScore*, *10000* in the area under *yearlyIncome*, *99* in the area under *age*, *1000000* in the area under *amount*, *12/12/21* in the area under *effectDate* and *1000* in the area under *yearlyRepayment*. Finally enter ***Basic RnJlZDpmcmVkcHdk*** in the area beside *Authorization*.

Response Content Type	application/json ∨				
Parameters					
Parameter	Value	Description	Parameter Type	Data Type	
Authorization	Basic RnJlZDpmcmVkcHdk		header	string	
postMiniloan_request	-	request body	body	Model Example Value	

Request schema for the MINILOAN JSON interface

MINILOANOperation -

   miniloan_commarea -

    name
    [John]
    creditScore
    [100]
    yearlyIncome
    [10000]
    age
    [99]
    amount
    [10000000]
    effectDate
    [12/12/21]
    yearlyRepayment
    [1000]

```
{
  "MINILOANOperation": {
    "miniloan_commarea": {
      "name": "string",
      "creditScore": 0,
      "yearlyIncome": 0,
      "age": 0,
      "amount": 0,
      "effectDate": "string",
      "yearlyRepayment": 0
    }
}
```

Parameter content type: application/json ∨

[Try it out!]

Compose the API for two-way MQ service

Mitch Johnson (mitchj@us.ibm.com)

___10. Click **Try it out!**. You should see a *200* in the *Response Code* area and a display of the response from the loan application in the *Response Body*. This is the message from the application returned in the reply queue. The loan was not approved (*F* in the *approved* field) and an explanation in the *messages* array.

Response Body

```
{
    "MINILOANOperationResponse": {
      "miniloan_commarea": {
        "creditScore": 100,
        "amount": 10000000,
        "approved": "F",
        "effectDate": "12/12/21",
        "messages_Num": 4,
        "name": "John",
        "messages": [
          "The age exceeds the maximum.",
          "The loan cannot exceed 1000000",
          "Credit score below 200",
          "The yearly income is lower than the basic request",
          "",
          "",
          "",
          "",
          "",
          ""
```

Response Code

```
200
```

___11. Change the request field as shown below and press **Try it Out!** again.

___12. This time the loan should be approved based on the value in the *approved* field.

ZCONEE - z/OS Connect EE

___13. (Optional) Start a CICS execution diagnostic trace on transaction MINC, e.g. **CEDX MINC** and trace the flow in CICS. Note that the Swagger-UI client will eventually will fail with a timed-out message:

> {
>
>   *"errorMessage": "BAQR0405I: The asynchronous request under URL*
>   *https://wg31.washington.ibm.com:9453/miniloan/loan has timed out after 30,000 milliseconds."*
>
>   }

Mitch Johnson (mitchj@us.ibm.com)

Compose the API project for a One-Way Service

In this section the MQAPI API will be enhanced by add support for MQ one-way services. MQ one-way service only supports the **POST** and **GET** methods. The POST method puts JSON request message on a queue after converting the JSON request message to a non-JSON format. The **GET** method is used get messages from a queue and convert the contents to a JSON response message.

___1. The *MQ for z/OS Service Provider for z/OS Connect* in a one-way service which supports the **POST** ('put' a message on a queue) and **GET** (a 'get' of a message from a queue). The view may need to be adjusted by dragging the view boundary lines. Start by click in plus sign beside the *Path* for the */loan* request.

.
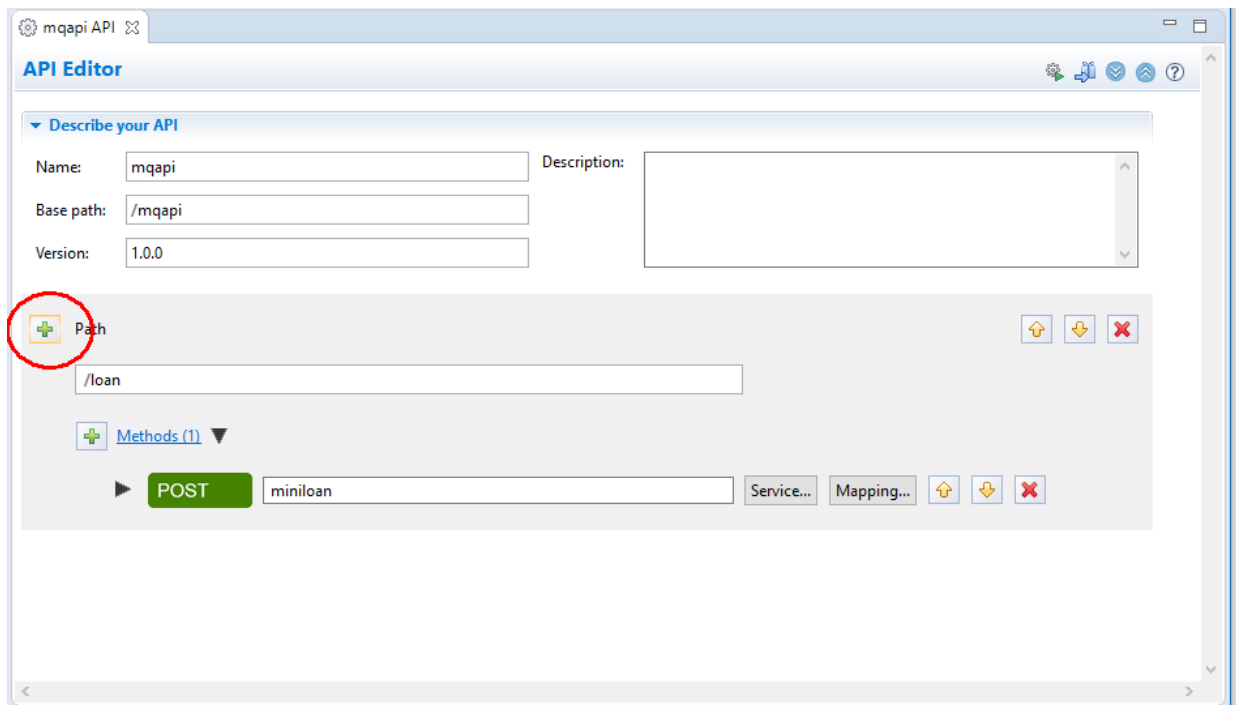
___2. Start by entering */queue* as the *Path* string in the *z/OS Connect EE API Editor* view and delete the PUT and DELETE methods. Click on the **Service** button to the right of the **POST** method. Then on the *Select a z/OS Connect EE Service* window select *mqPut*. Click **OK** to continue. This will populate the field to the right of the method.

___3. Click on the **Service** button to the right of the **GET** method. Then on the *Select a z/OS Connect EE Service* window select *mqGet*. Click **OK** to continue. This will populate the field to the right of the method.

___4. When completed the API editor should look something like this.

Mitch Johnson (mitchj@us.ibm.com)

___5. Next, click on the **Mapping** button beside the **POST** method and then select *Open Request Mapping*:

___6. In the mapping view that opens, go to the right side of the mapping (which represents the COPYBOOK fields), and click the little + signs to expand *mqmessage*. You should see fields that correspond to the fields defined in the original COBOL copy book *FILEAMQ.cpy* (see below).


___6.
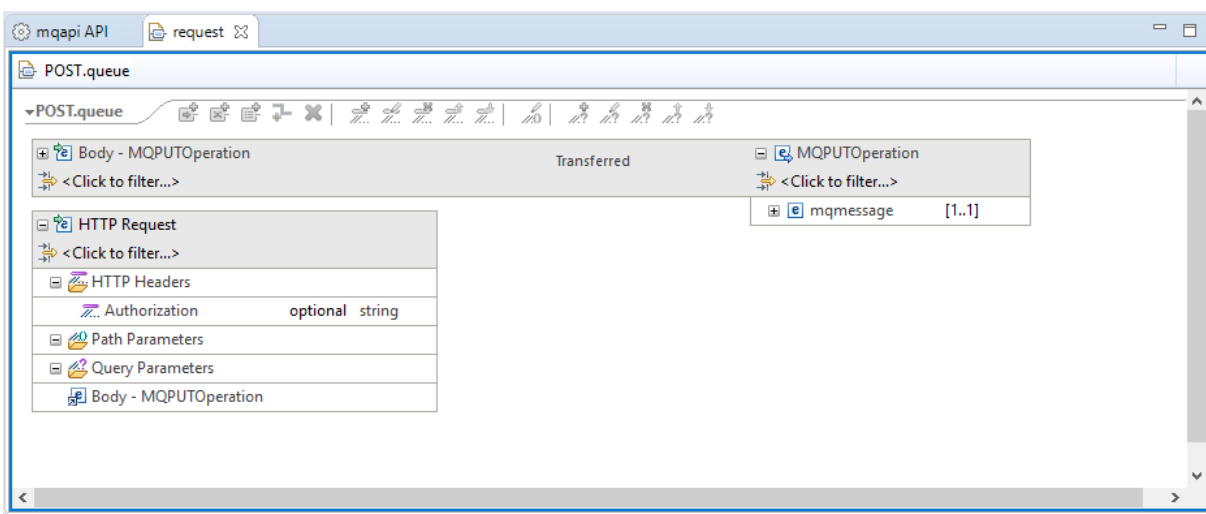
```
01 MQMESSAGE.
        10  stat        PIC X(1).
        10  numb        PIC X(6).
        10  name        PIC X(20).
        10  addrx       PIC X(20).
        10  phone       PIC X(8).
        10  datex       PIC X(8).
        10  amount      PIC X(8).
        10  comment     PIC X(9).
```

Mitch Johnson (mitchj@us.ibm.com)

Use the slider bar to fully expose the *mqmessage* structure. No mapping or changes are required since every field will be supplied in the JSON request message.

This is where values can be assigned to fields and/or fields can be omitted from the interface.
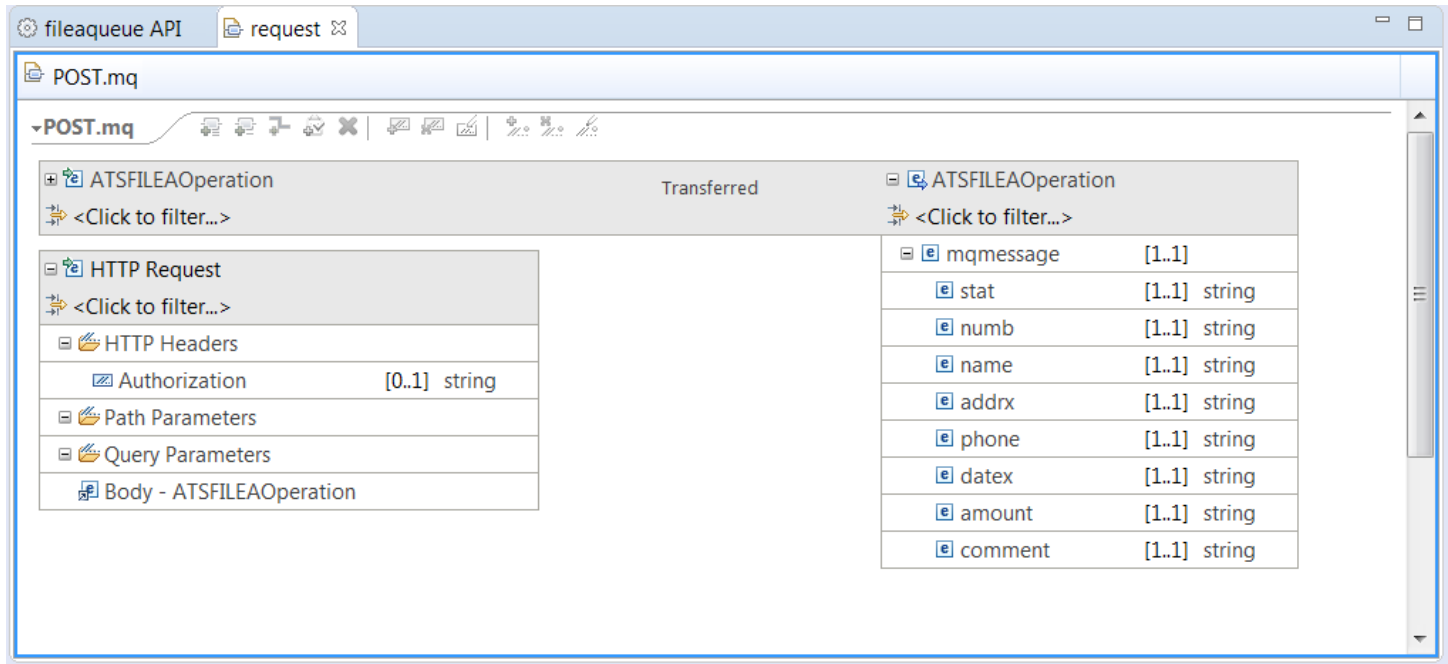
___7. Use the **Ctrl-S** key sequence to close this view.

___8. Click on the **Mapping** button beside the **POST** method and then select *Open Default Response Mapping*. Note that the response body has no content. A POST (MQPUT) does not a response JSON message.

THIS LINE SHOULD NOT APPEAR
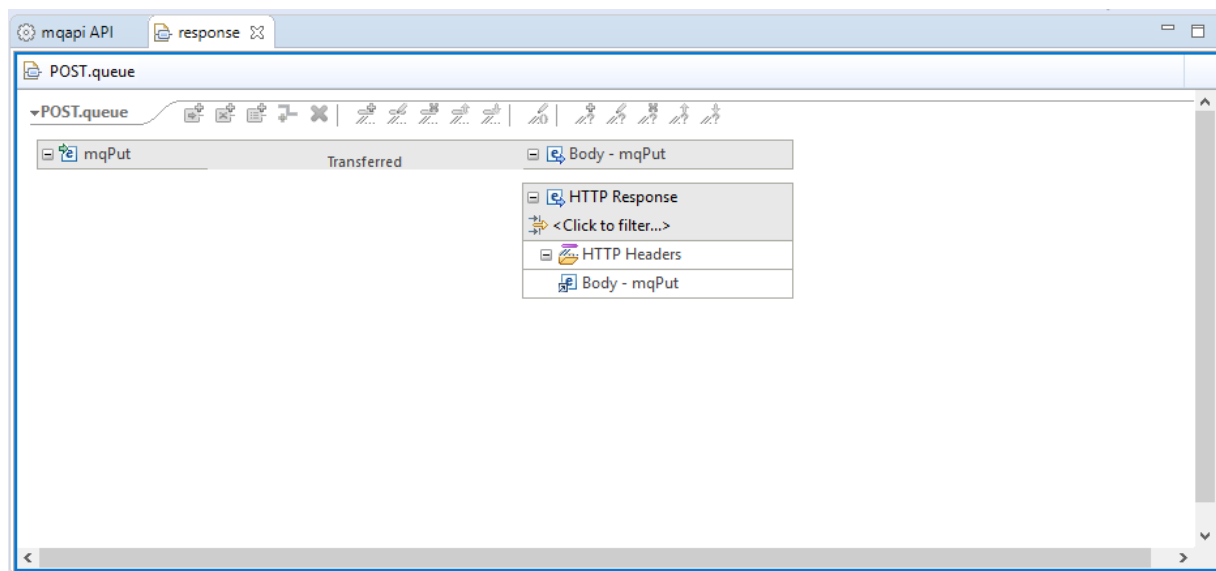
___9. Next, click on the **Mapping** button beside the **GET** method and then select *Open Request Mapping*. Note that there are no fields displayed in the request message.
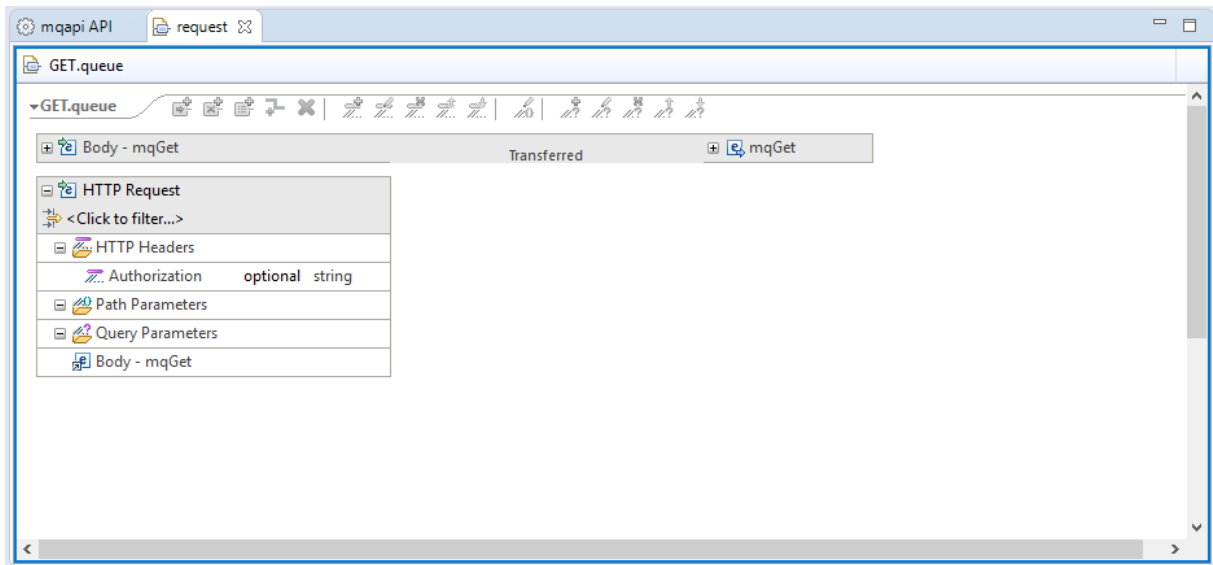
This is because a GET request not use require a JSON request message, only a JSON response message is returned.
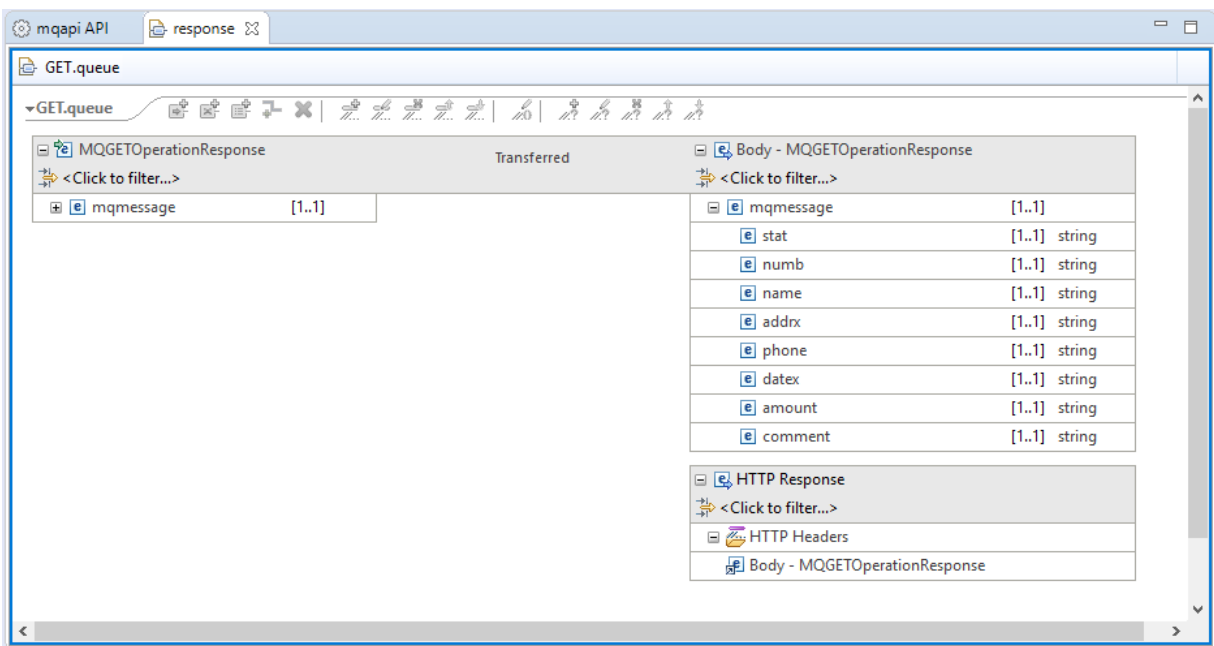
Summary

You created the API, which just a base path and with the **POST** and **GET** HTTP methods along with the request and response mapping required for each method. This API will now be deployed into z/OS Connect EE V3.0.

Deploy the API to a z/OS Connect EE Server

Review the z/OS Connect server.xml updates required for the *MQ for z/OS Connect Service Provider for z/OS Connect* before deploying the API.

___5. The *queue* service is added to the z/OS Connect server's *server.xml* file by the including the *mq.xml* file (see below).

```
<server description="MQ Service Provider">

  <featureManager>
        <feature>jms-2.0</feature>
        <feature>mqzosconnect:zosConnectMQ-2.0</feature>          1
        <feature>wmqJmsClient-2.0</feature>
        <feature>zosTransaction-1.0</feature>
  </featureManager>

  <variable name="wmqJmsClient.rar.location"
            value="/shared/mqm/V9R0M1/java/lib/jca/wmq.jmsra.rar"/>
  <wmqJmsClient nativeLibraryPath="/shared/mqm/V9R0M1/java/lib"/>

  <connectionManager id="ConMgr1" maxPoolSize="5"/>

<jmsConnectionFactory id="qmgrCf" jndiName="jms/qmgrCf"
        connectionManagerRef="ConMgr1">
        <properties.wmqJMS transportType="BINDINGS"          2
                queueManager="QMZ1" />
</jmsConnectionFactory>

<jmsQueue id="q1" jndiName="jms/default">
     <properties.wmqJms
        baseQueueName="ZCONN2.DEFAULT.MQZCEE.QUEUE"          3
        CCSID="37"/>
</jmsQueue>


<jmsQueue id="request" jndiName="jms/request">
     <properties.wmqJms
        baseQueueName="ZCONN2.TRIGGER.REQUEST"
        targetClient="MQ"
        CCSID="37"/>
</jmsQueue>

<jmsQueue id="response" jndiName="jms/response">
      <properties.wmqJms
         baseQueueName="ZCONN2.TRIGGER.RESPONSE"
         targetClient="MQ"
         CCSID="37"/>
  </jmsQueue>
</server>
```

1. The *featureManager* element identifies the Liberty features required by the *MQ for z/OS Service Provider for z/OS Connect.*

2. The *jmsConnectionFactory* element associates the JMS connection factory(*jndiName*) with the target queue manager and details on how to connect to this queue manager.

3. The jmsQueue elements provide details that associate the JMS destination (*jndiName*) with the target queue (*baseQueueName*) and its JMS/MQ properties.

___6. In the *Project Explorer* view (upper left), right-mouse click on the *MQAPI* folder, then select *z/OS Connect EE→ Deploy API to z/OS Connect EE Server*.
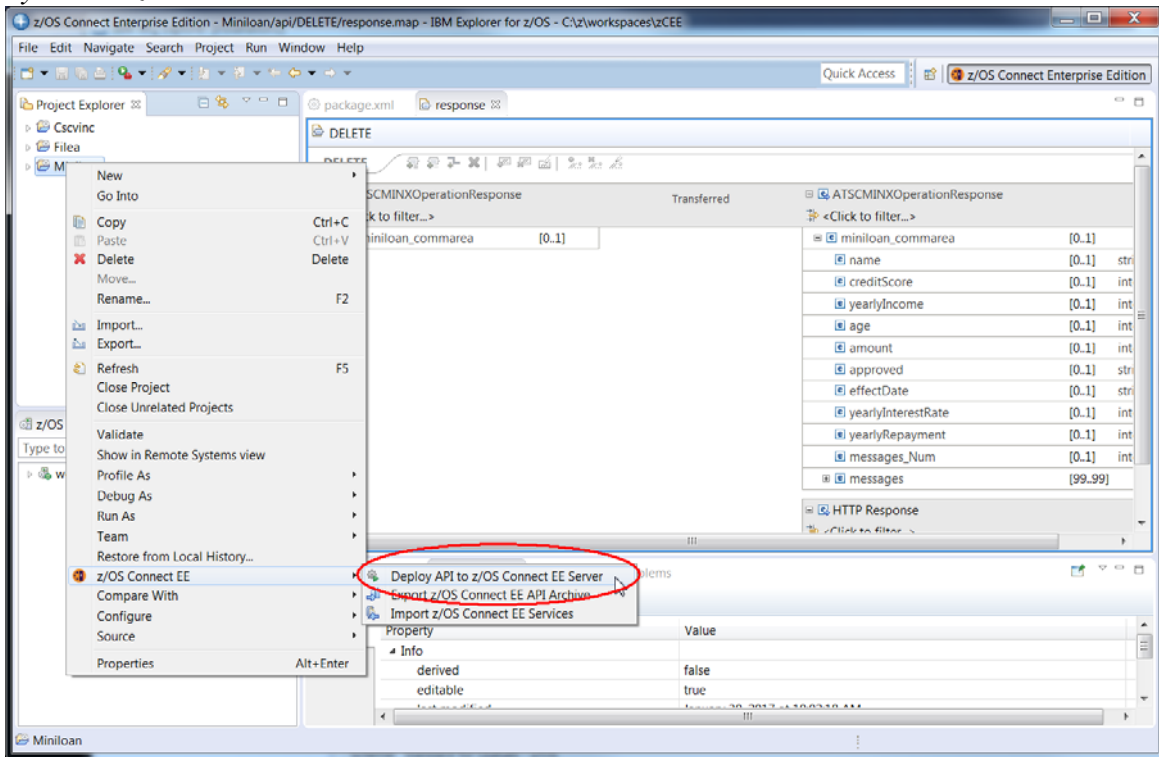
___7. Since z/OS Explorer is connected to only one z/OS Connect server there is only one choice (*wg31:9453*). If z/OS Explorer had multiple host connections to z/OS Connect servers then the pull-down arrow would allow a selection to which server to deploy. Check the box beside *Update existing APIs* and click **OK** on this screen to continue.
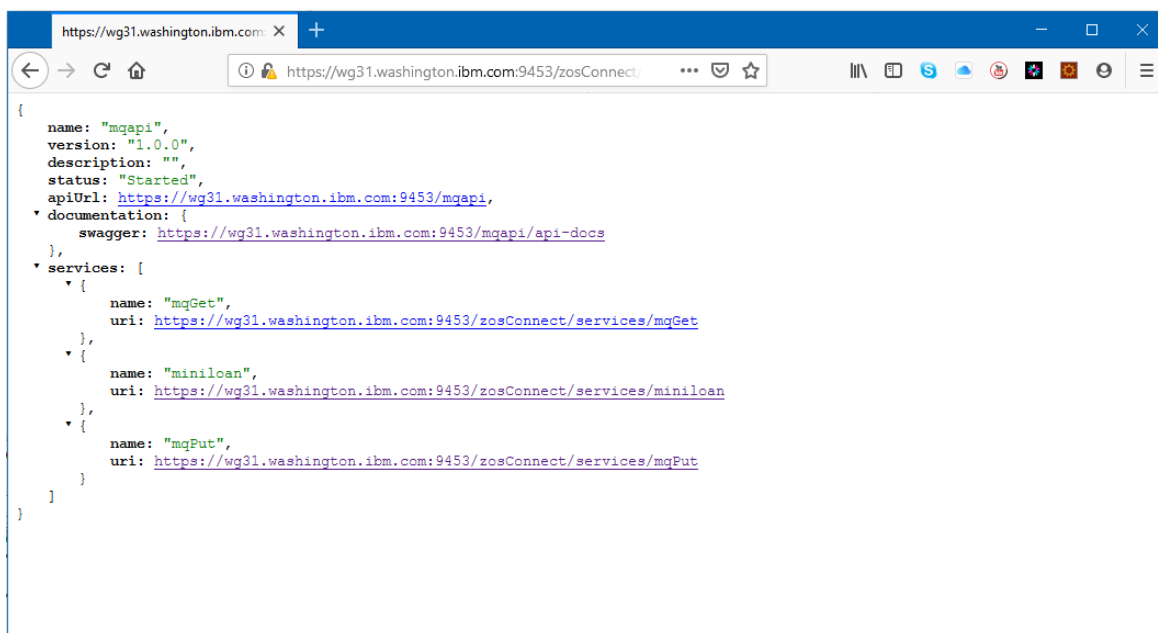
ZCONEE - z/OS Connect EE
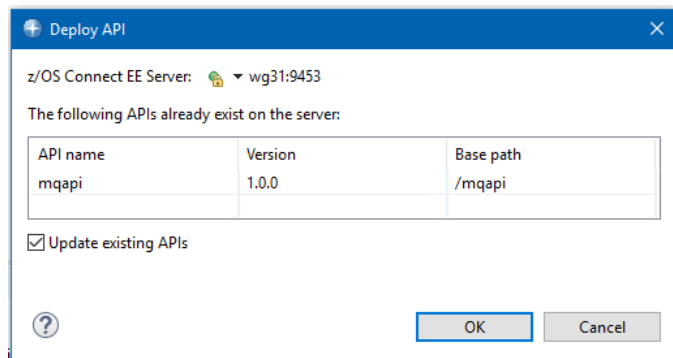
___8. The API artifacts will be transferred to z/OS and copied into the */var/ats//zosconnect/servers/zceesrv1/resources/zosconnect/apis* directory.
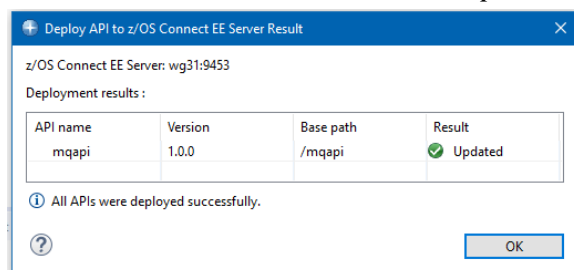
Test the MQ One Way Service API

A MQ "one way" service provides a REST interface for putting to and getting messages from a queue (or topic). The supported REST methods are; **POST** ('put' a message), **GET** ('get' a message).

___1. Enter URL ***https://wg31.washington.ibm.com:9453/zosConnect/apis/mqapi*** in the Firefox browser and you should see the window below. The new services should not be displayed.


Tech Tip: You may be challenged by Firefox because the digital certificate used by the Liberty z/OS server is self-signed Click the **Add Exception** button to continue. If the **Add Exception** button is not displayed click the **Advanced** button. Then click on the **Confirm Security Exception** button. Next you may see a prompt you for a userid and password. If you do see the prompt, enter the username *Fred* and password **fredpwd** (case matters) and click **OK**. Remember we are using basic security, and this is the user identity and password defined in the server.xml file.

Compose the API project for a One-Way Service

© Copyright IBM Corporation 2017, 2019 All rights reserved

Mitch Johnson (mitchj@us.ibm.com)

38 of 45

> **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.

___2. Finally click on the *swagger* URL and you should see the Swagger document associated with this API.

___3. Explore this Swagger document and you will see the results of the request and response mapping performed earlier. This document can be used by a developer or other tooling to develop REST clients for this specific API.

___4. 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.

___5. Right click the mouse button on *mqapi* and select *Open in Swagger UI*. Click **OK** if an informational prompt appears. This will open a Firefox window showing a *Swagger* test client (see below).
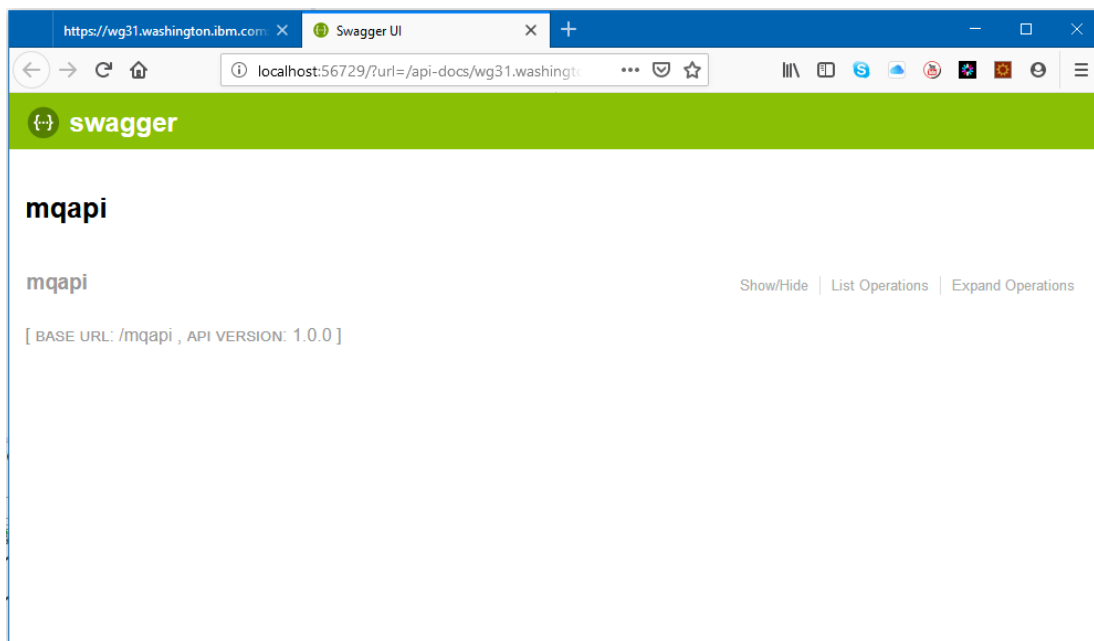
___6. Click the *List Operations* and the browser should show a list of the available HTTP methods for this API.
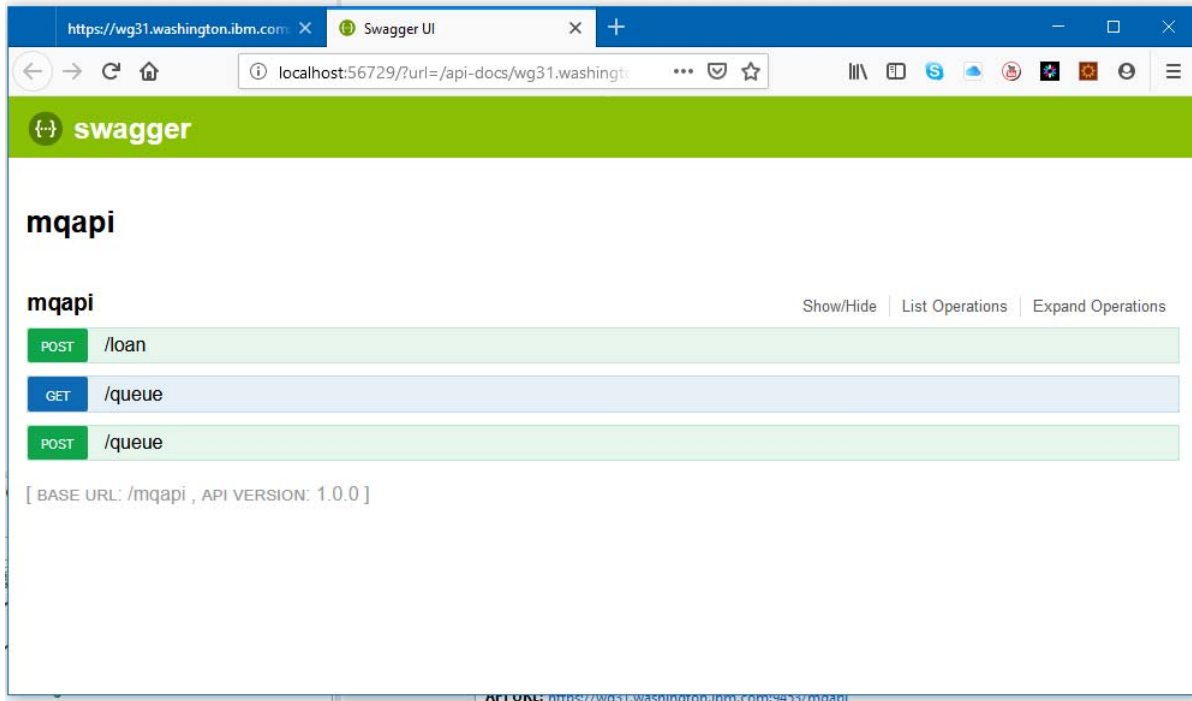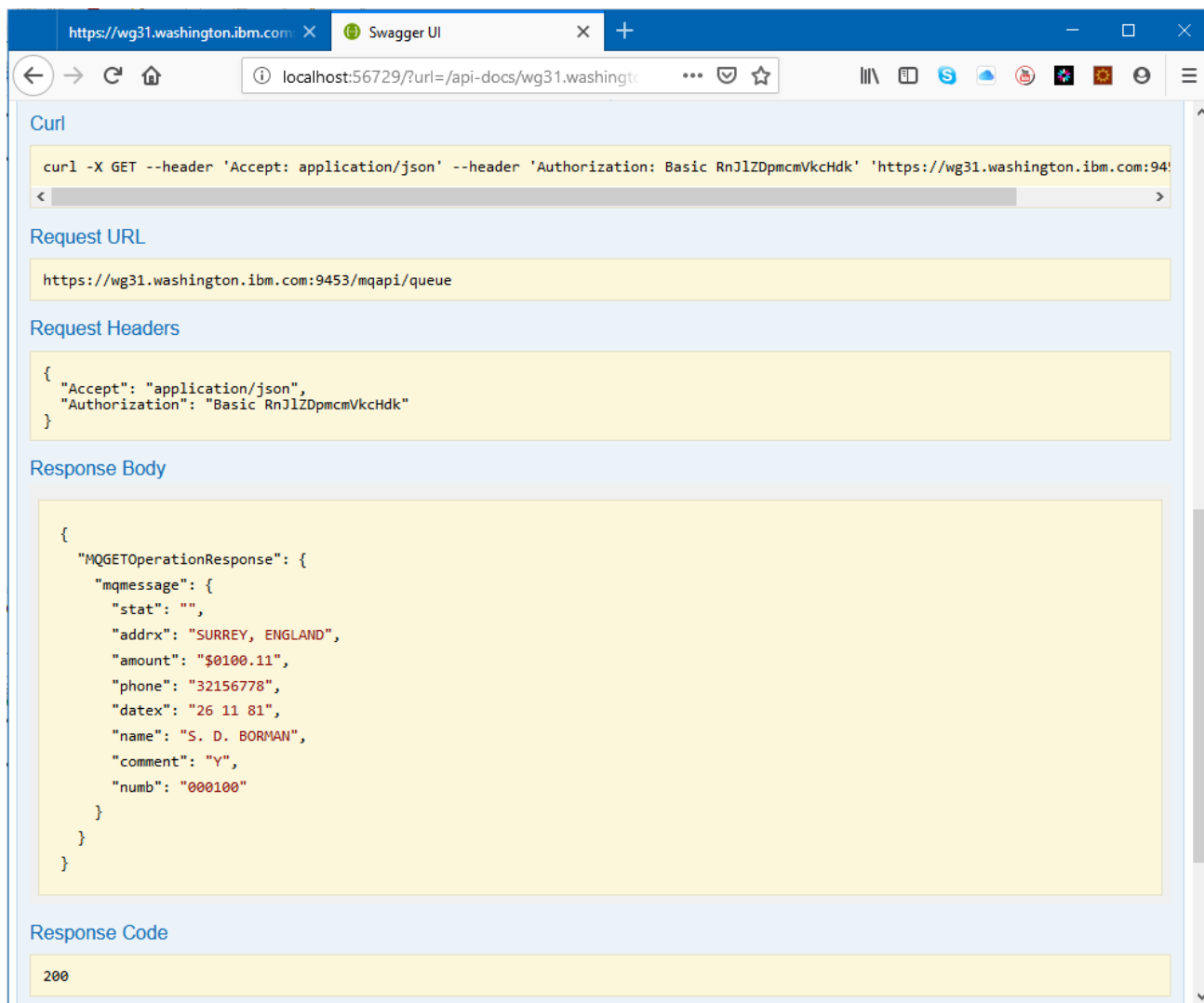

___7. Expand the **GET** method by clicking on the path beside it (e.g. */mq*) and scroll down until the method *Parameters* are displayed as shown below:
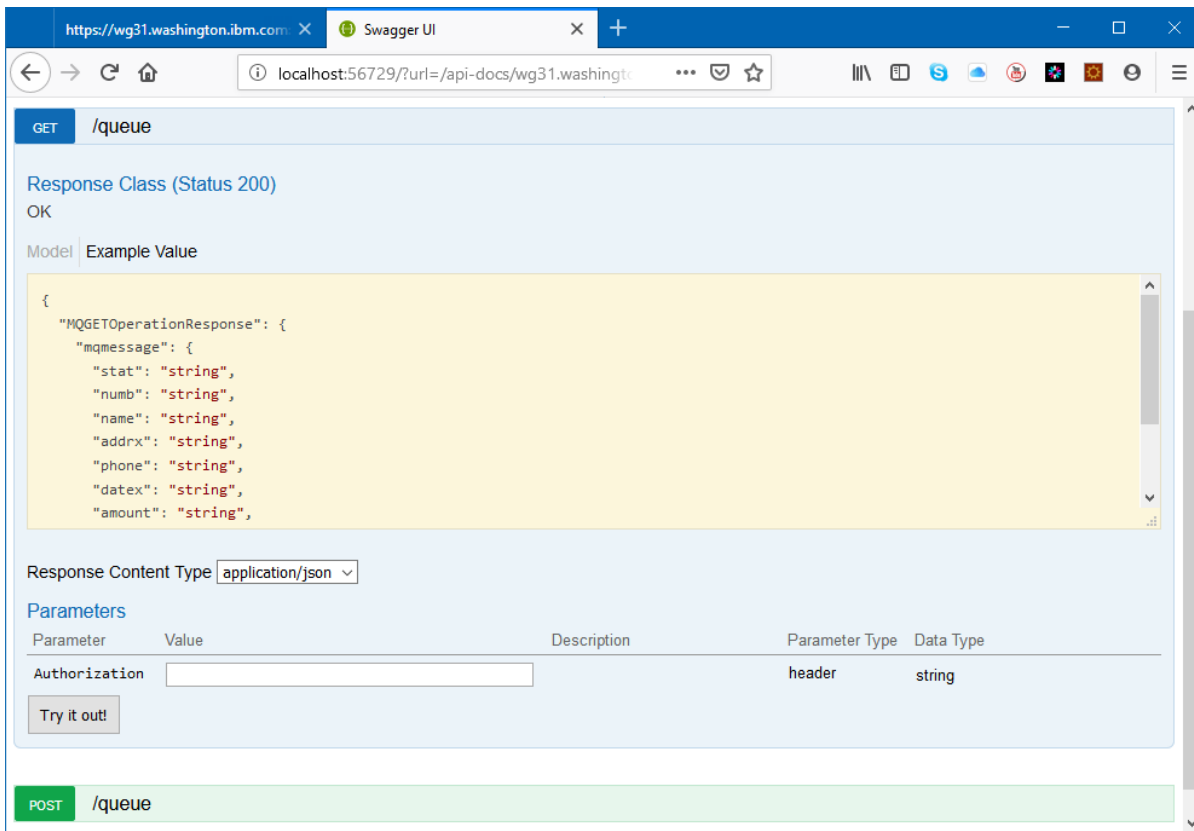
___8. Enter **_Basic RnJlZDpmcmVkcHdk_** in the area beside *Authorization* and click **Try it out!**. Scroll down to display the response in the *Response Body*. This is the message from the application returned in the reply queue. This was the message retrieved by a non-destructive get. If you continue to click the **Try it out!** button you will see the same message over and over.

___9. Expand the **POST** method by clicking on the path beside it (e.g. */queue)* and scroll down until the method *Parameters* are displayed as shown below. Enter values for each field (see examples below) and press the **Try it Out!** button.

POST	/queue

Parameters

Parameter	Value	Description	Parameter Type	Data Type
Authorization			header	string
postMqPut_request	-	request body	body	Model Example Value

Request schema for the MQPUT JSON interface

MQPUTOperation   -

mqmessage   -

stat
Y

numb
948478

name
Don Bagwell

addrx
Raleigh NC

phone
0065

datex
26 11 81

amount
$100.00

comment
****

```
{
    "MQPUTOperation": {
        "mqmessage": {
            "stat": "string",
            "numb": "string",
            "name": "string",
            "addrx": "string",
            "phone": "string",
            "datex": "string",
            "amount": "string",
            "comment": "string"
```

Parameter content type: application/json ∨

Response Messages

HTTP Status Code	Reason	Response Model	Headers
200	OK		

Try it out!

Mitch Johnson (mitchj@us.ibm.com)

___10. Scroll down to display the response in the *Response Body*. Note that there is no response message. The MQ Service provider does not return a reply message for the *POST* method. If you continue to click the **Try it out!** button you will see the same results over and over.

| Try it out! | Hide Response |

Curl

```
curl -X POST --header 'Content-Type: application/json' --header 'Accept: application/json' --header 'Authorization: Basic RnJlZDpmcmcm
```

◄ ▐▐▐ ►

Request URL

```
https://wg31.washington.ibm.com:9453/fileaqueue/mq
```

Request Headers

```
{
  "Accept": "application/json",
  "Authorization": "Basic RnJlZDpmcmcmVkcHdk"
}
```

Response Body

```
no content
```

Response Code

```
204
```

Response Headers

```
{
  "content-language": "en-US",
  "expires": "Thu, 01 Dec 1994 16:00:00 GMT",
  "cache-control": "no-cache=\"set-cookie, set-cookie2\"",
  "content-type": null
}
```

Tech-Tip: An HTTP code of 204 the server processed the request but returned no content. For an explanation of HTTP codes see URL
https://en.wikipedia.org/wiki/List_of_HTTP_status_codes

___11. Confirm by using the *MQExplorer* tool on the desktop and browse the messages in the *ZCONN2.DEFAULT.MQZCEE.QUEUE* queue.

- Select *QMZ1* under **Queue Managers** and right click the mouse button

- Select the *Connect* option.

- Once connected, expand the *Queues* folder and select *ZCONN2.DEFAULT.MQZEE.QUEUE* and right click the mouse button.

Tech Tip: The message is first in the list because it has a higher priority than the other messages.

___12. Do a few more **GET** and **POST** requests and observe the changes in the queue.

Summary

You have added the *MQ for z/OS Service Provider for z/OS Connect* to the z/OS Connect EE server and configured two services. One service (Miniloan) supports a reply/response application using two queues and the other service (Queue)) is a one-way service where the same queue is used for POSTs and GETs.