IBM z/OS Connect EE V3.0

Developing RESTful APIs for MQ Services

IBM WSC Wildfire Team
IBM z Systems

Lab Version Date: April 18, 2019

Table of Contents

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

Mitch Johnson (mitchj@us.ibm.com)

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 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 **yellow** in this document should be available for copying and pasting in a file named *Development APIs CopyPaste* file on the desktop.

Mitch Johnson (mitchj@us.ibm.com)

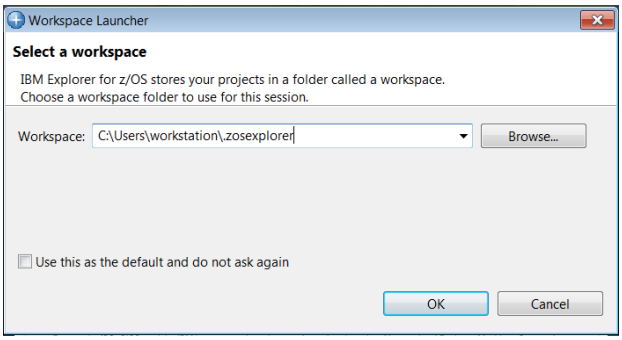
Connect to the z/OS Connect EE Server

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.

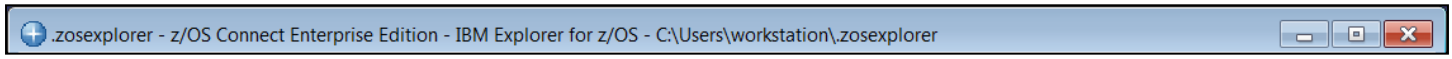
___1. On the workstation desktop, locate the *z/OS Explorer* icon and double click on it to open the Explorer.

___2. You will be prompted for a workspace:



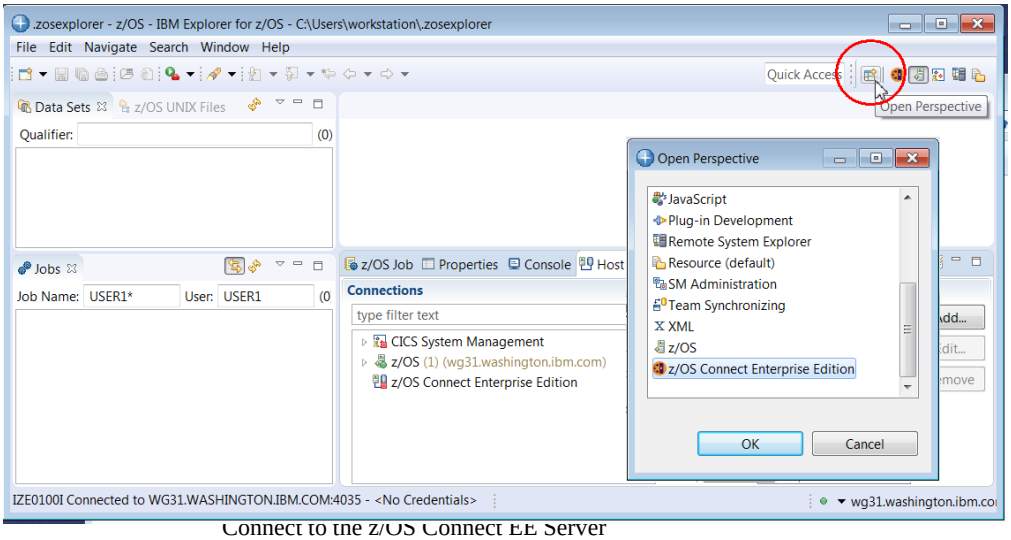
Take the default value by clicking **OK**.

___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:



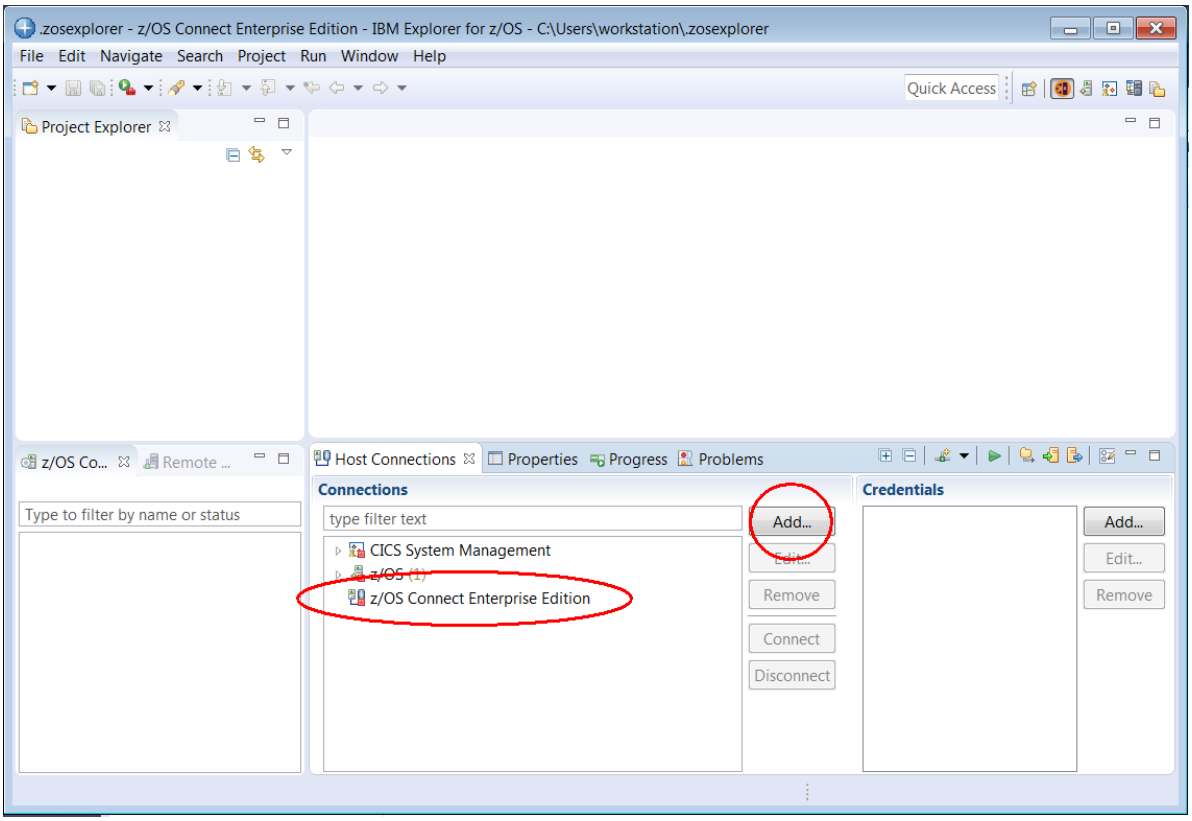
N.B. If a *Welcome* screen is displayed then click the white X beside *Welcome* to close this view.

___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.



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 connection*s tab in the lower view and then click the **Add** button.



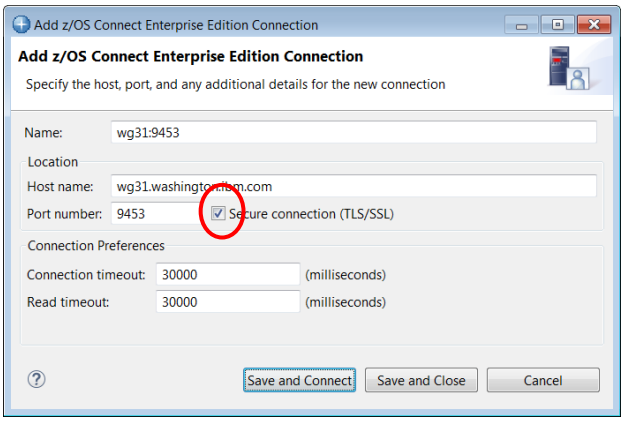
Tech-Tip: Eclipse based development tools like z/OS Explorer; provide a graphical interface consisting of multiple views within a single window.

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.

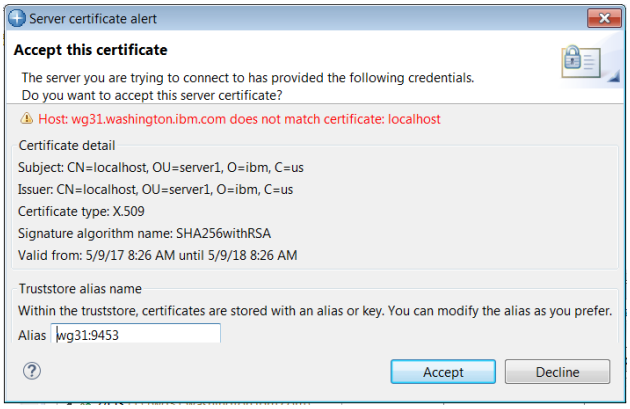
Eclipse based tools also can display multiple views based on the current role of the user. In this context, a window is known as a perspective. The contents (or views) of a perspective are based on the role the user, i.e., developer or administrator.

___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)

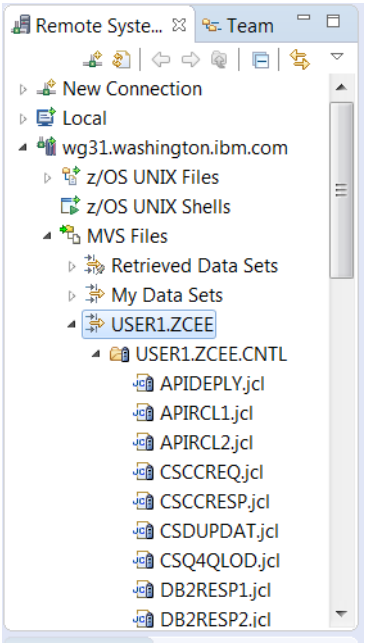
MQ for z/OS Service Provider for z/OS Connect

This unit 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 IBM MQ for z/OS V9.0.1. 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**, **GET** and **DELETE** methods are used put and get messages from a single queue.

Run the job to create the data conversion artifacts

First run the BAQLSJ2JS utility to generate the conversion artifacts required for the execution of a service, the service that will be used to access MQ The services will be referred called *FileaQueue* and *Miniloan* in this exercise.

___1. Back in the IBM z/OS Explorer session switch to the *Remote System Explorer* perspective (see page 4) and expand data set *USER1.ZCEE.CNTL* in the *Remote System view* to display a list of its members.



Mitch Johnson (mitchj@us.ibm.com)

Members *LS2JSMIN* and *LS2JSFIL* are the jobs that executes the BAQLS2JS program that will convert the COPYBOOKs for the service *Miniloand* and *Fileaqueue* services. N.B. disregard the file extensions of jcl. N.B. disregard the file extensions of jcl.

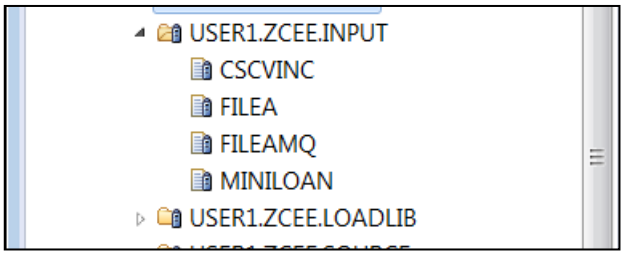
___2. Open member **LS2JSFIL** by double clicking the member**.** You will see that it is a very simple JCL job:

```
//LS2JSVSM JOB (0),MSGCLASS=X,CLASS=A,NOTIFY=&SYSUID,REGION=500M
 //ASSIST   EXEC PGM=BPXBATCH
 //STDPARM DD DSN=USER1.ZCONN2.INPUT(FILEAMQ),DISP=SHR
 /*
 //STDOUT   DD SYSOUT=*
 //STDERR   DD SYSOUT=*
 //STDENV   DD *
 JAVA_HOME=/shared/java/J8.0_64
 //
```

The key thing we want you to notice is the STDPARM DD, which points to a data set member where the input parameters for this job are located. The parameters can be included inline, but in this exercise some were longer than 80 columns wide and the utility does not yet support continuation. So we put the parameters in a data set allocated with width 256 columns.

___3. Expand data set *USER1.ZCEE.INPUT*.

___4. You will see four members:



Mitch Johnson (mitchj@us.ibm.com)

___5. Open member *FILEAMQ* member and you should see something like this:

```
PGM /shared/IBM/zosconnect/V3r0/bin/baqls2js
PDSLIB=USER1.ZCEE.CNTL
REQMEM=FILEAREQ
RESPMEM=FILEARSP
MAPPING-LEVEL=4.0
CHAR-VARYING=COLLAPSE
JSON-SCHEMA-REQUEST=/var/ats/zosconnect/servers/zceesrvr/dataXform/json/FileaQueue_request.json
JSON-SCHEMA-RESPONSE=/var/ats/zosconnect/servers/zceesrvr/dataXform/json/FileaQueue_response.jso
LANG=COBOL
LOGFILE=/var/ats/zosconnect/servers/zceesrvr/dataXform/Filea.log
PGMINT=COMMAREA
PGMNAME=ATSFILEA
WSBIND=/var/ats/zosconnect/servers/zceesrvr/dataXform/bind/FileaQueue.wsbind
SERVICE-ARCHIVE=/var/ats/zosconnect/servers/zceesrvr/dataXform/sars/FileaQueue.sar
SERVICE-NAME=FileaQueue
URI=http://zceesrvr.example.org:8080/Filea
```

1
2
3

1. Pointers to the data set where the request and response COPYBOOK members are located
2. Pointers to the directory where the request and response JSON schemas will be created
3. Pointers to the target directories for the binding file (required at runtime) and the service archive (SAR) file.

___6. Open member *MINILOAN* member and you should see something like this:

```
PGM /shared/IBM/zosconnect/V3r0/bin/baqls2js
PDSLIB=USER1.ZCEE.CNTL
REQMEM=MINILOAN
RESPMEM=MINILOAN
DATA-TRUNCATION=ENABLED
MAPPING-LEVEL=4.0
CHAR-VARYING=COLLAPSE
JSON-SCHEMA-REQUEST=/var/ats/zosconnect/servers/zceesrvr/dataXform/json/Miniloan_request.json
JSON-SCHEMA-RESPONSE=/var/ats/zosconnect/servers/zceesrvr/dataXform/json/Miniloan_request.json
LANG=COBOL
LOGFILE=/var/ats/zosconnect/servers/zceesrvr/dataXform/Miniloan.log
PGMINT=COMMAREA
PGMNAME=ATSCMINX
WSBIND=/var/ats/zosconnect/servers/zceesrvr/dataXform/bind/Miniloan.wsbind
SERVICE-ARCHIVE=/var/ats/zosconnect/servers/zceesrvr/dataXform/sars/Miniloan.sar
SERVICE-NAME=Miniloan
URI=http://zceesrvr.example.org:8080/Miniloan
```

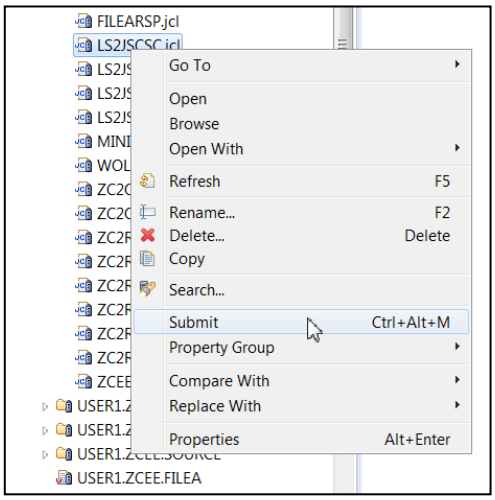
1
2
3
4

1. Pointers to the data set where the request and response COPYBOOK member is located
2. Pointers to the directory where the request and response JSON schemas will be created
3. The PGMINIT indicates that this is a *COMMAREA* application.
4. Pointers to the target directories for the binding file (required at runtime) and the service archive (SAR) file. The SAR file will be downloaded to a workstation where it will be imported in the z/OS Connect Enterprise Edition eclipse tooling.

> **Tech Tip:** The DATA-TRUNCATION=ENABLED was added for the MINILOAN member because this COPYBOOK has a structure with an *occurs depending on* variable. Enabling DATA-TRUNCATION simplifies the sending of the variable in the request and response messages by reducing the storage requirements, etc.

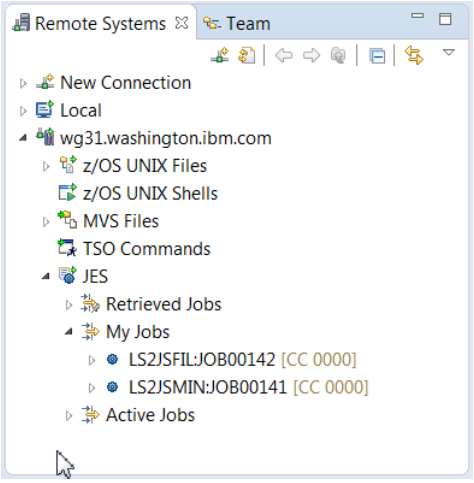
Mitch Johnson (mitchj@us.ibm.com)

___7. Submit jobs *LS2JSMIN* and *LS2JSFIL* for execution. These jobs should complete with a maximum return code of zero (e.g. MAXCC=0000).



Tech Tip: To submit a job for execution when using the IBM z/OS Explorer select the member and right click the mouse button 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 clicking the mouse button and selecting the *Open* option.

___8. Click *OK* on the next screen. After submitting the job, expand the *JES* icon and expand *My Jobs* to display the held output. The output of an entire job can be viewed by opening the spool file or output for a specific DD statement can be viewed by expanding the job and opening a specific output file.



Close any opened editor views and do not save any changes

Import the SAR files

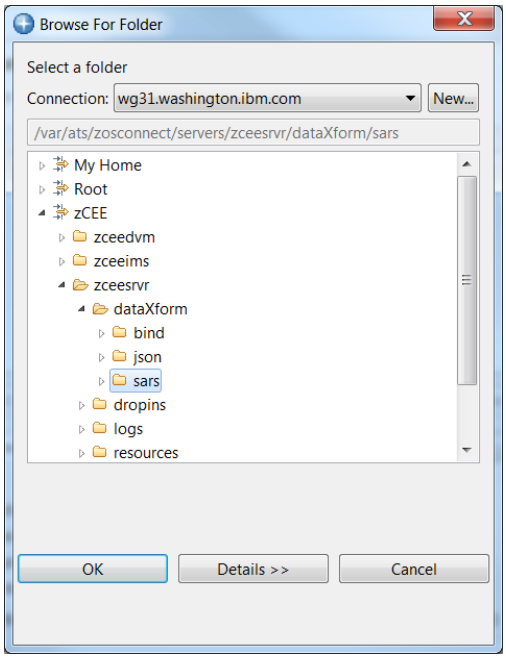
The next step is to move the SAR files created by running the jobs in the last section from the OMVS directory structure by importing them into the workspace.

___1. In the z/OS Explorer in the *z/OS Connect Enterprise Edition* perspective in the the *Project Explorer* view (upper left). Select ***File*** on the tool bar and then on the pop up select **New → Project**. Expand the *General* folder and select *Project* to create a target project for exporting the Service Archive (SAR) files. Click **Next** to continue.

N.B. The *services* folder may already exist, if it does continue with Step 3.

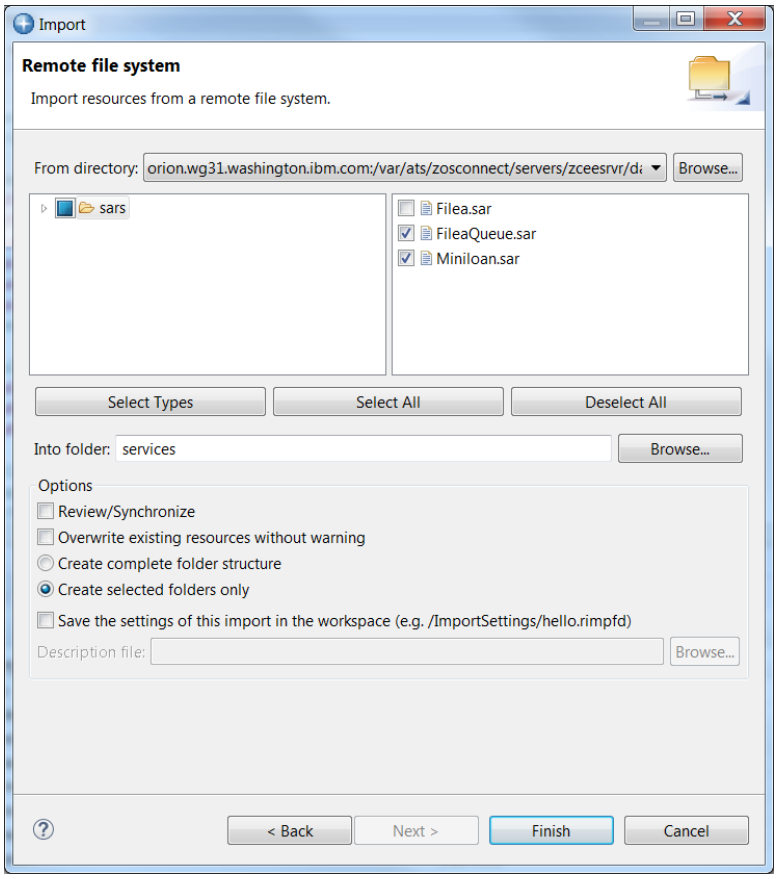
___2. On the *New Project* window enter ***services*** as the *Project name*. Click **Finish** to continue. This action will add a new project in the *Project Explorer* named *services*.

___3. Select the *services* folder. Right click the mouse button and select *Import* from the list of options. On the *Import – Select* window expand *Remote Systems* and select *Remote file system* and click **Next** to continue. On the *Import – Remote file system* press the *Browse* button. On Browse for Folder window use the pull-down allow to select *wg31.washington.ibm.com*. Expand the zCEE folder (this is the filter created earlier) and expand the *zceesrvr* folder, then expand the *dataXform* folder and then select the *sars* folder. Click **OK** to continue.



Mitch Johnson (mitchj@us.ibm.com)

___4. On the *Import – Remote file system* window expand the *sars* folder and check the boxes beside both *FileaQueue.sar* and *Miniloan.sar* files and click the **Finish** button:

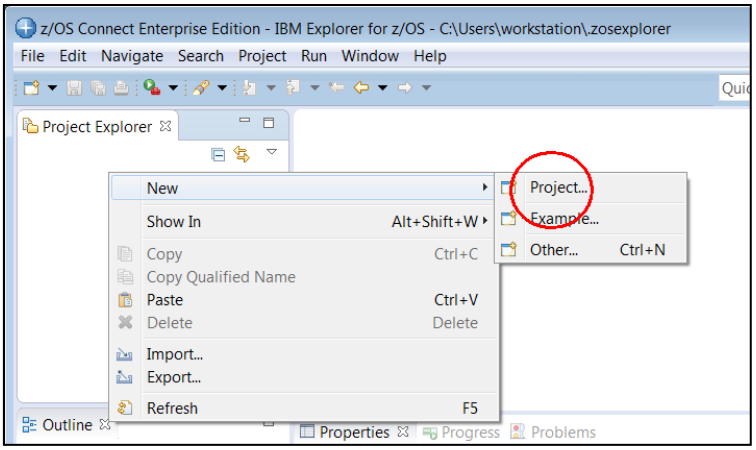


In the *Project Explorer* view (upper left), expand the *services* folder to see the the imported SAR file

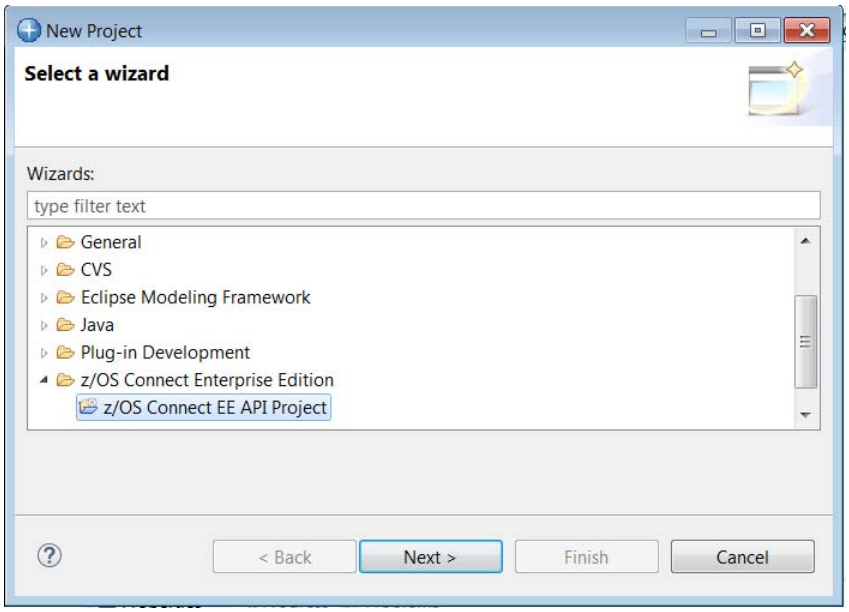
Create the Miniloan API project (a MQ Two-Way Service)

In this section an API that uses a MQ two-way service will be developed. A MQ two-service only supports the **POST** method. A JSON request message is put on a request queue after being converted to the format expected by the back-end application. The application put the response message on the reply queue and this response message is used to build the JSON response message.

___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*:

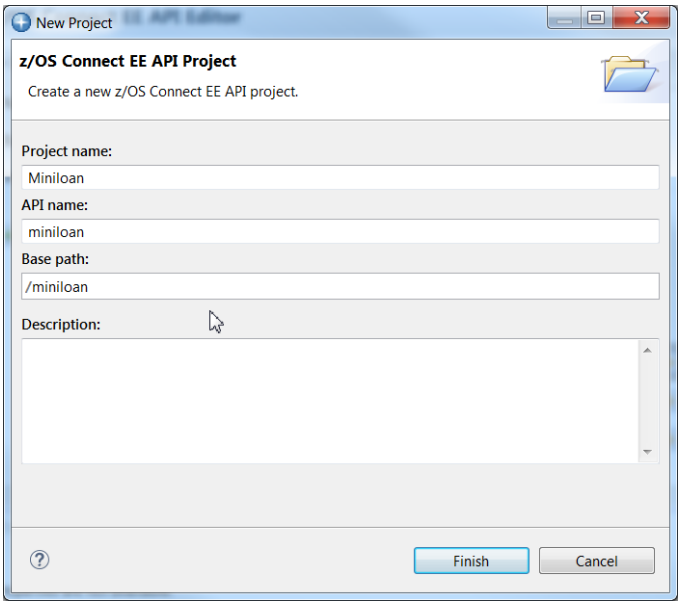


___2. In the *New Project* screen, scroll down and open the *z/OS Connect Enterprise Edition* folder and select *z/OS Connect EE API Project* and then click the **Next** button.



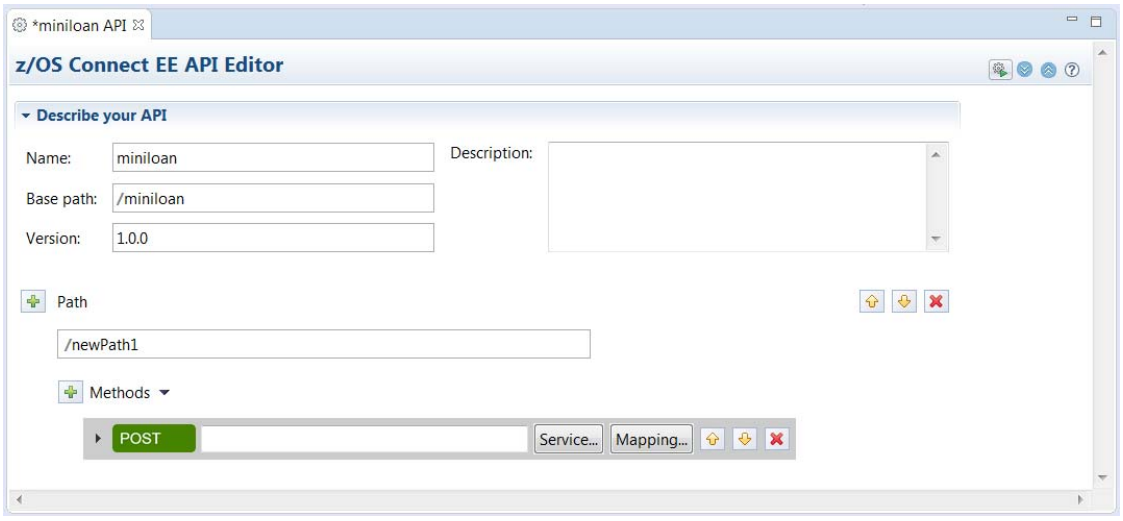
Mitch Johnson (mitchj@us.ibm.com)

___3. Enter ***Miniloan*** for the *Project name.* Be sure the *API name* is set to ***miniloan*** and the *Base path is* set to ***/miniloan***. 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.

___4. The *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** (nondestructive 'get' of a message), and **DELETE** (destructive 'get' of a message) methods are 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.

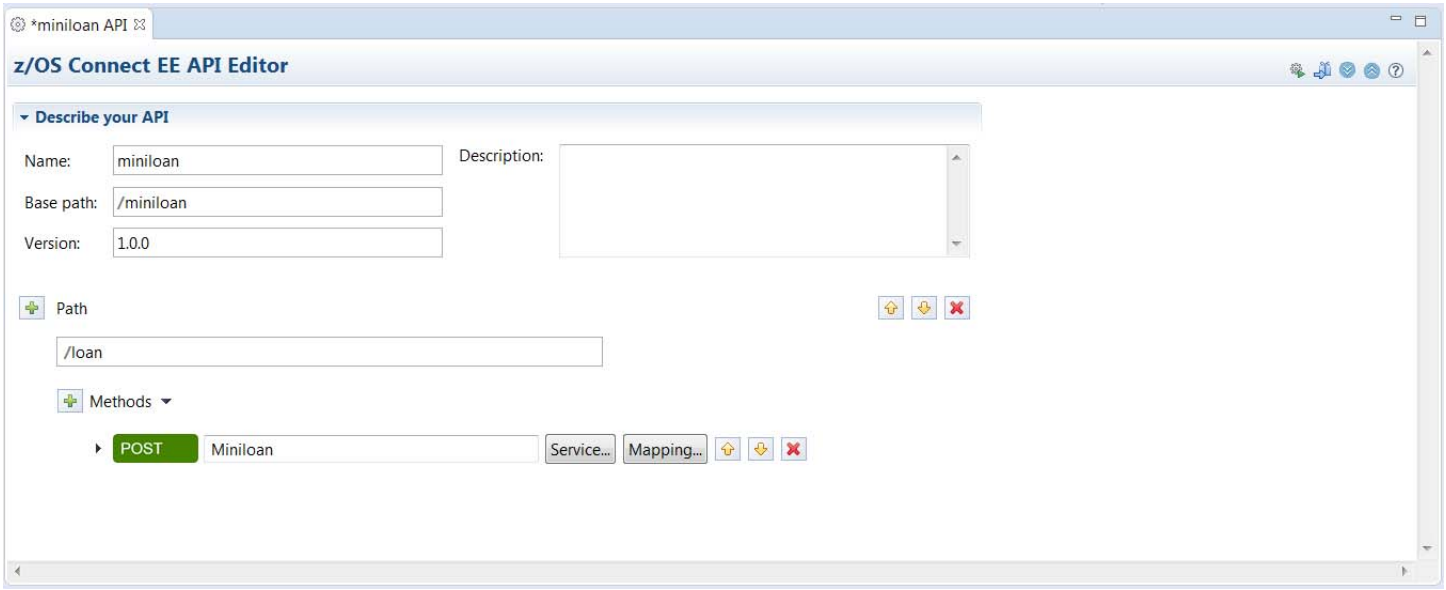


Summary

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

Compose the API for the CICS MQ Trigger Monitor Application

___1. 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 click on the **Workspace** button. Expand the *services* folder and select *Miniloan.sar*. Click **OK** three times. This will populate the field to the right of the method. This has imported this SAR file into the *Miniloan* project (expand the new *services* folder in *Miniloan*). Select this service for the **POST** method and remove the other methods as shown below:



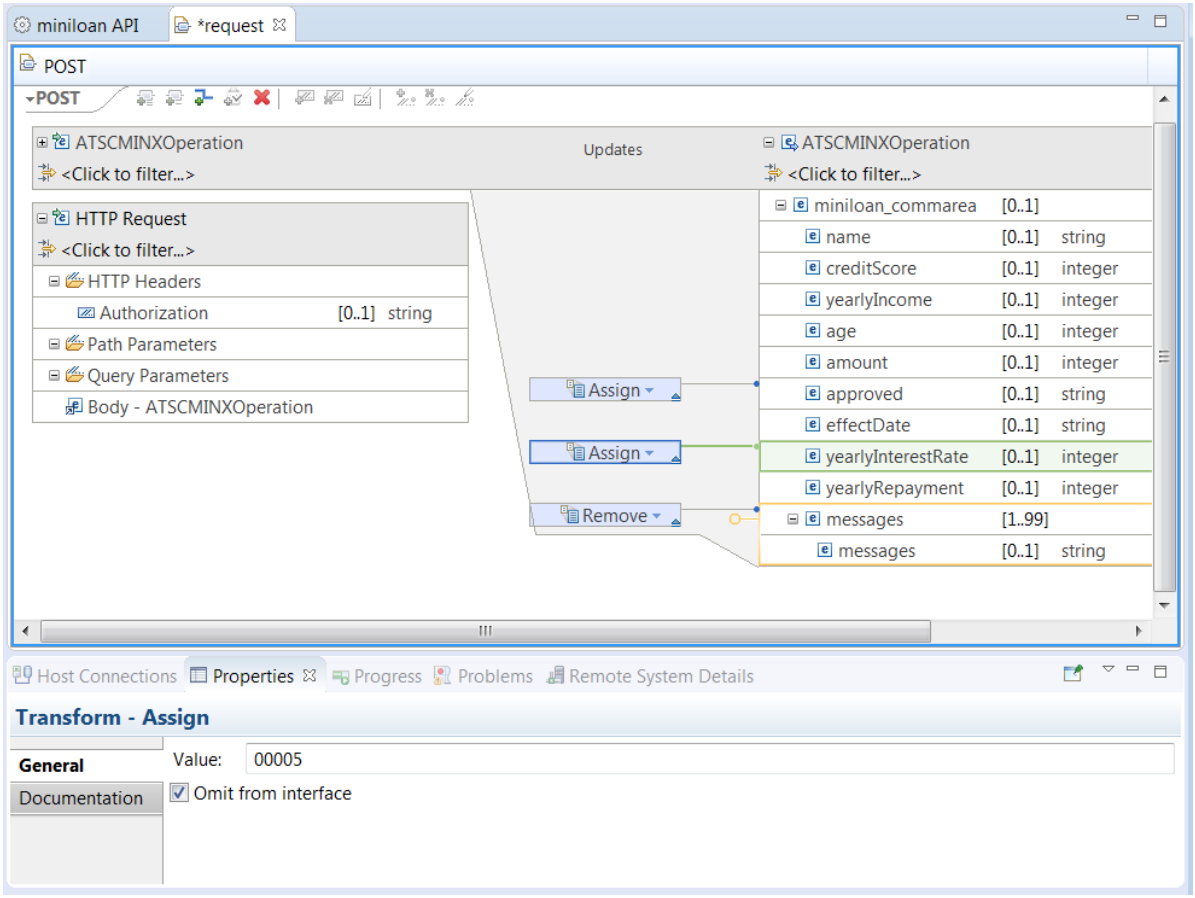
Mitch Johnson (mitchj@us.ibm.com)

___2. Next, click on the **Mapping** button beside the **POST** method and then select *Open Request Mapping*:

___3. 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* in *USER1.ZCEE.CNTL*.

```
      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 99 times
                        depending on messages-Num.
```

___5. 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. 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:



___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 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. The ***Miniloan*** service is added to the z/OS Connect server's *server.xml* file by the including the *miniloan.xml* file (see below).

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

 <featureManager>
       <feature>jms-2.0</feature>
       <feature>mqzosconnect:zosConnectMQ-2.0</feature>
       <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"/>

 <zosconnect_zosConnectService id="fileaqueue"
         invokeURI="/FileaQueue"
         dataXformRef="xformJSON2Byte"
         serviceName="FileaQueue"
         serviceDescription="MQ Oneway Service"
         serviceRef="FileaQueue" />

 <zosconnect_zosConnectService id="miniloan"
         dataXformRef="xformJSON2Byte"
         serviceName="Miniloan"
         serviceDescription="MQ Reply/Response Service"
         serviceRef="Miniloan" />

  <mqzosconnect_mqzOSConnectService id="FileaQueue"
         connectionFactory="jms/qmgrCf"
         destination="jms/default" />

  <mqzosconnect_mqzOSConnectService id="Miniloan"
         connectionFactory="jms/qmgrCf"
         waitInterval="30000"
         destination="jms/request"
         replyDestination="jms/response"/>

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

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

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

1

2

3

4

```
<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>
```

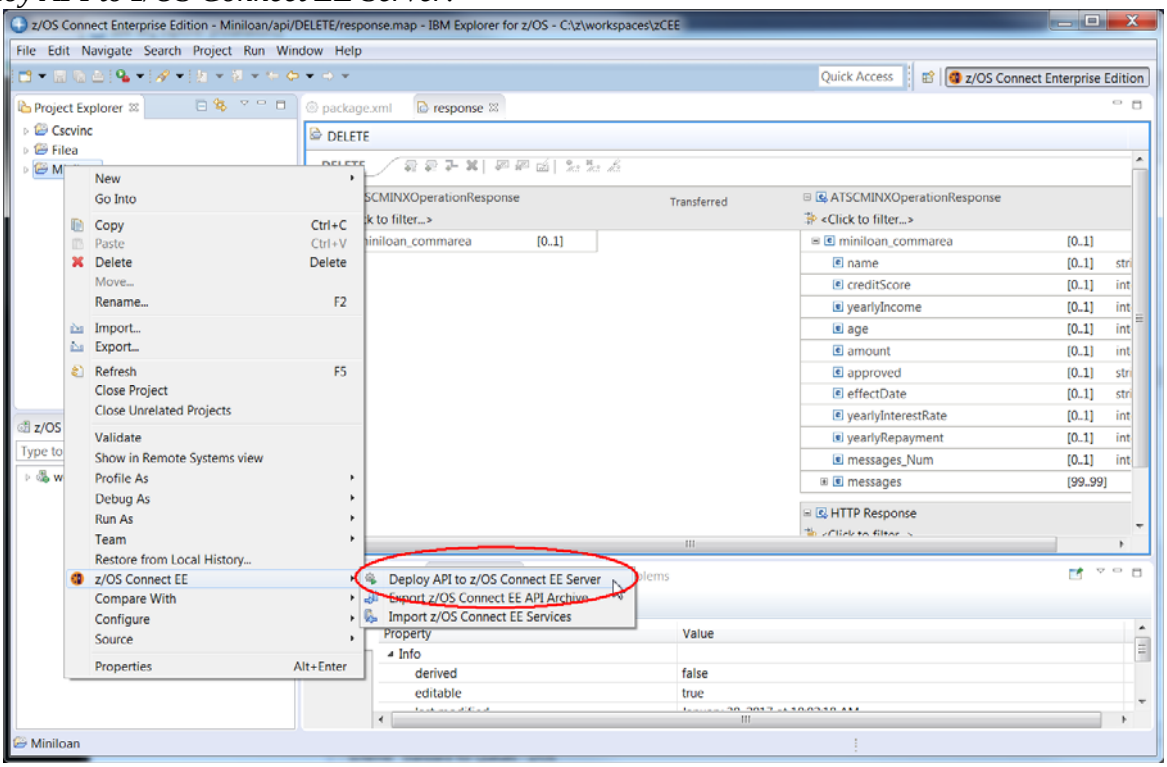
5

Figure 1 - miniloan.xml

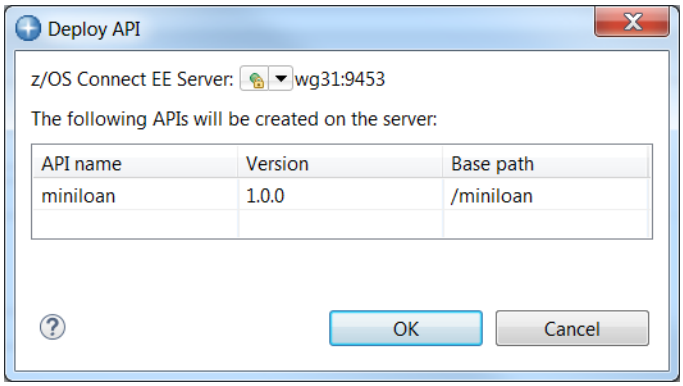
Tech-Tip: The *mqzosconnect_mqzOSConnectService* elements identify which queue is being access by each service using the JNDI names in the *destination* and/or the *replyDestination* attributes.

1. The *featureManager* element identifies the Liberty features required by the *MQ for z/OS Service Provider for z/OS Connect.*
2. The *zosConnectService* element provides security constraints (none in this case) and the identity of the *dataXformRef* element that provides the location of API artifacts.
3. The *mqzOSConnectServices* element identifies the JMS connection factories, the JMS destinations (queues) by JNDI name and other JMS characteristics required by each service.
4. The *jmsConnectionFactory* element associates the JMS connection factory(*jndiName*) with the target queue manager and details on how to connect to this queue manager.
5. The jmsQueue elements provide details that associate the JMS destination (*jndiName*) with the target queue (*baseQueueName*) and its JMS/MQ properties.

___2. In the *Project Explorer* view (upper left), right-mouse click on the *Miniloan* folder, then select *z/OS Connect EE→ Deploy API to z/OS Connect EE Server*.

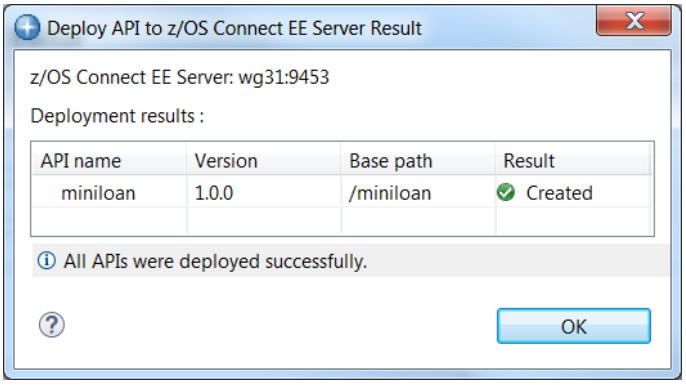


___3. Since z/OS Explorer is connected to only one z/OS Connect server there is only one choice (*wg31:9453*). If z/OS Explorer had multiple host connections to z/OS Connect servers then the pull-down arrow would allow a selection to which server to deploy. Click **OK** on this screen to continue.



Mitch Johnson (mitchj@us.ibm.com)

___4. The API artifacts will be transferred to z/OS and copied into the */var/ats//zosconnect/servers/zceesrvr/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 Operational Decision Management 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.

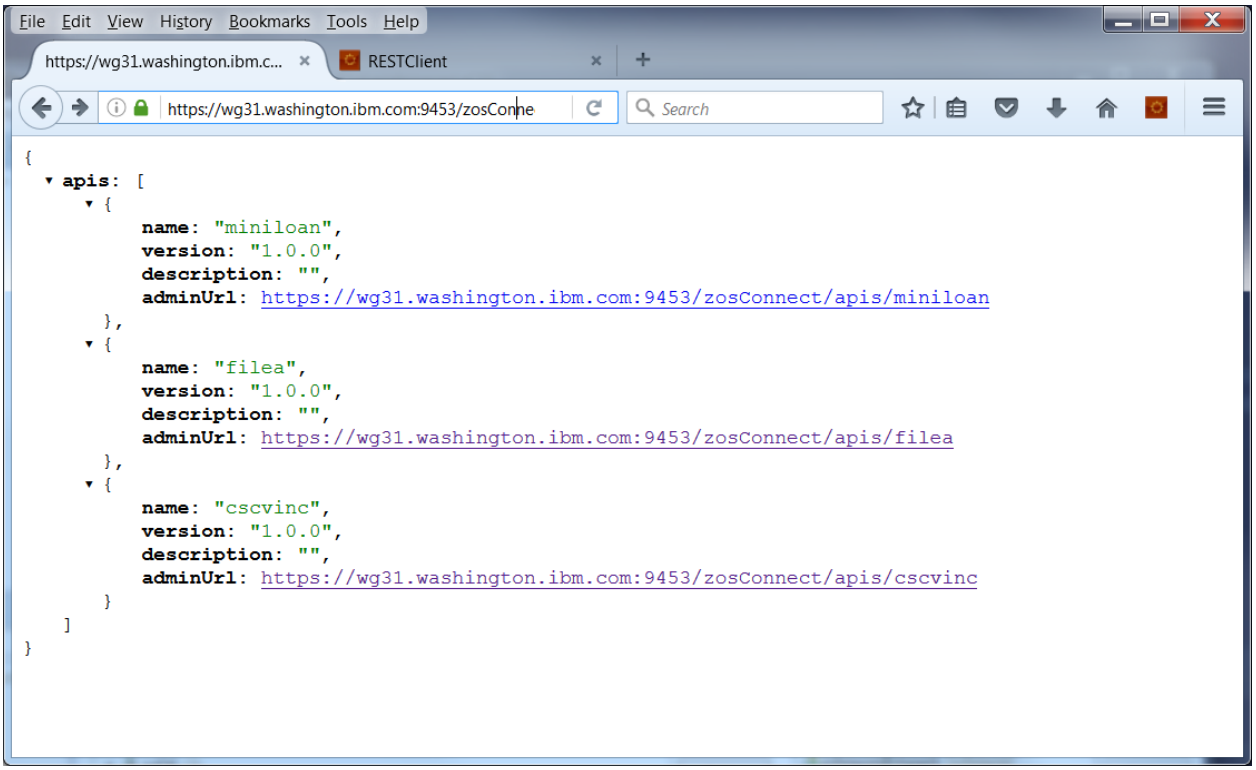
The rules for rejecting a loan can be for any one of the following:

- 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.

.

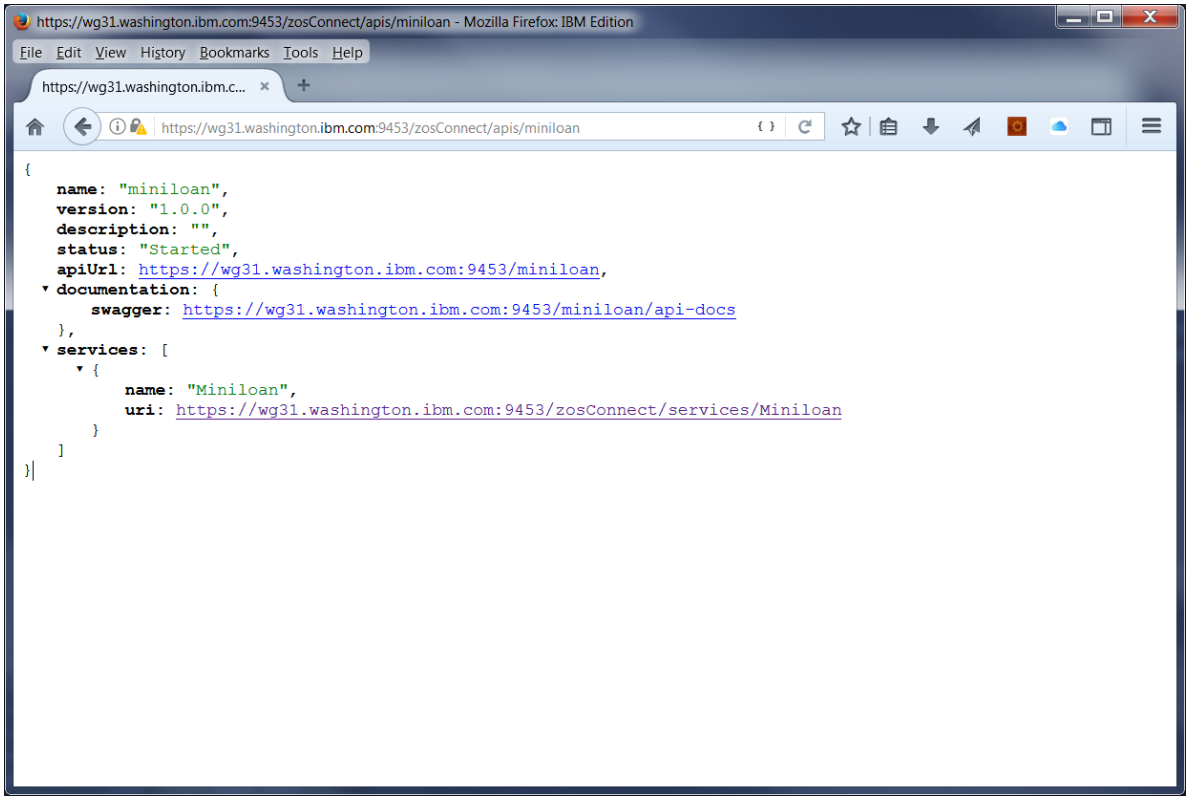
. **Tech Tip:** You may be challenged by Firefox because the digital certificate used by the Liberty z/OS server is self-signed Click the **Add Exception** button to continue. If the **Add Exception** button is not displayed click the **Advanced** button. Then click on the **Confirm Security Exception** 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.

___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.

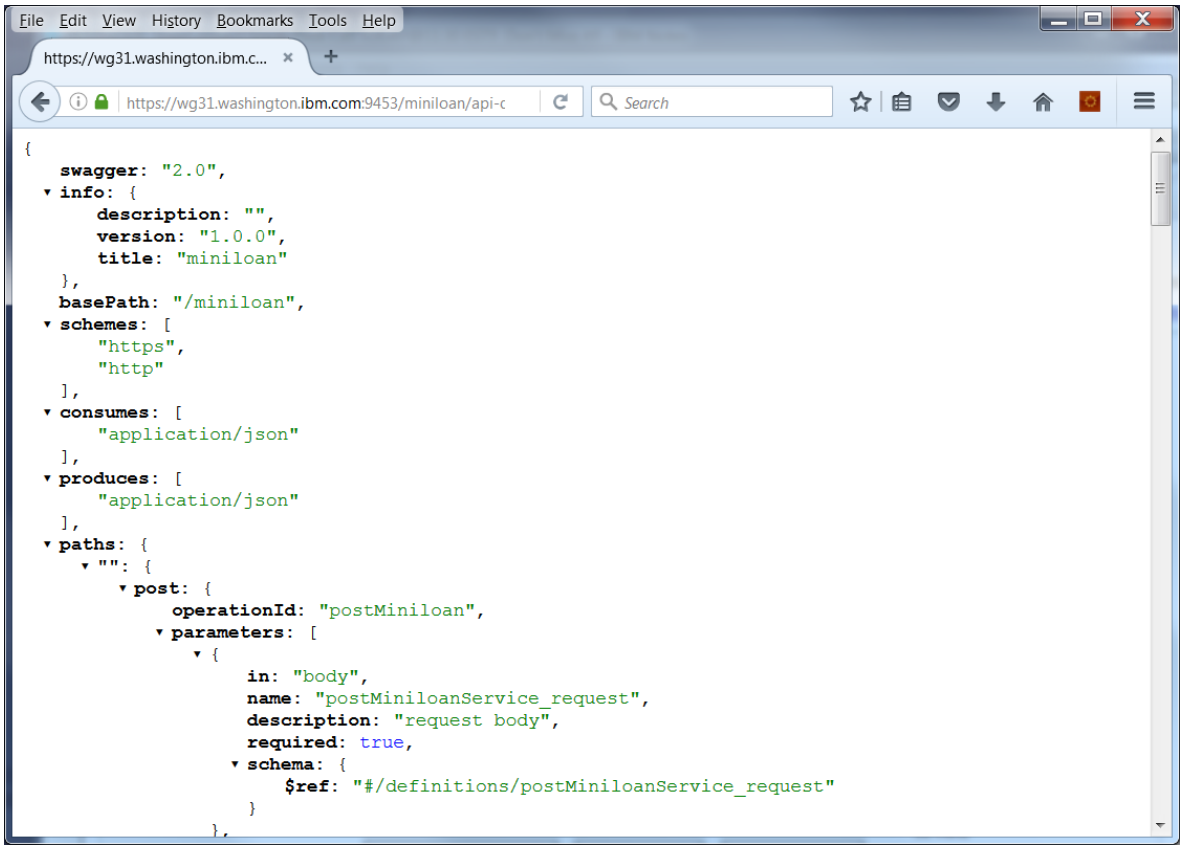


. **Tech Tip:** It is very important to access the z/OS Connect server from a browser prior to any testing using the Swagger UI. Accessing a z/OS Connect URL from a browser starts an SSL handshake between the browser and the server. If this handshake has not performed prior to performing any test the 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.

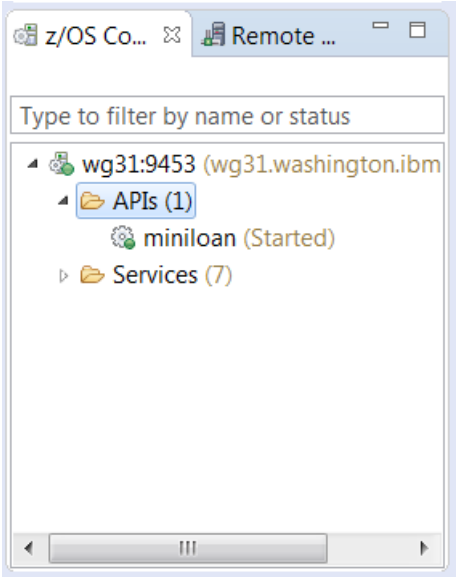


___3. Finally click on the *swagger* URL and you should see the Swagger document associated with this API.

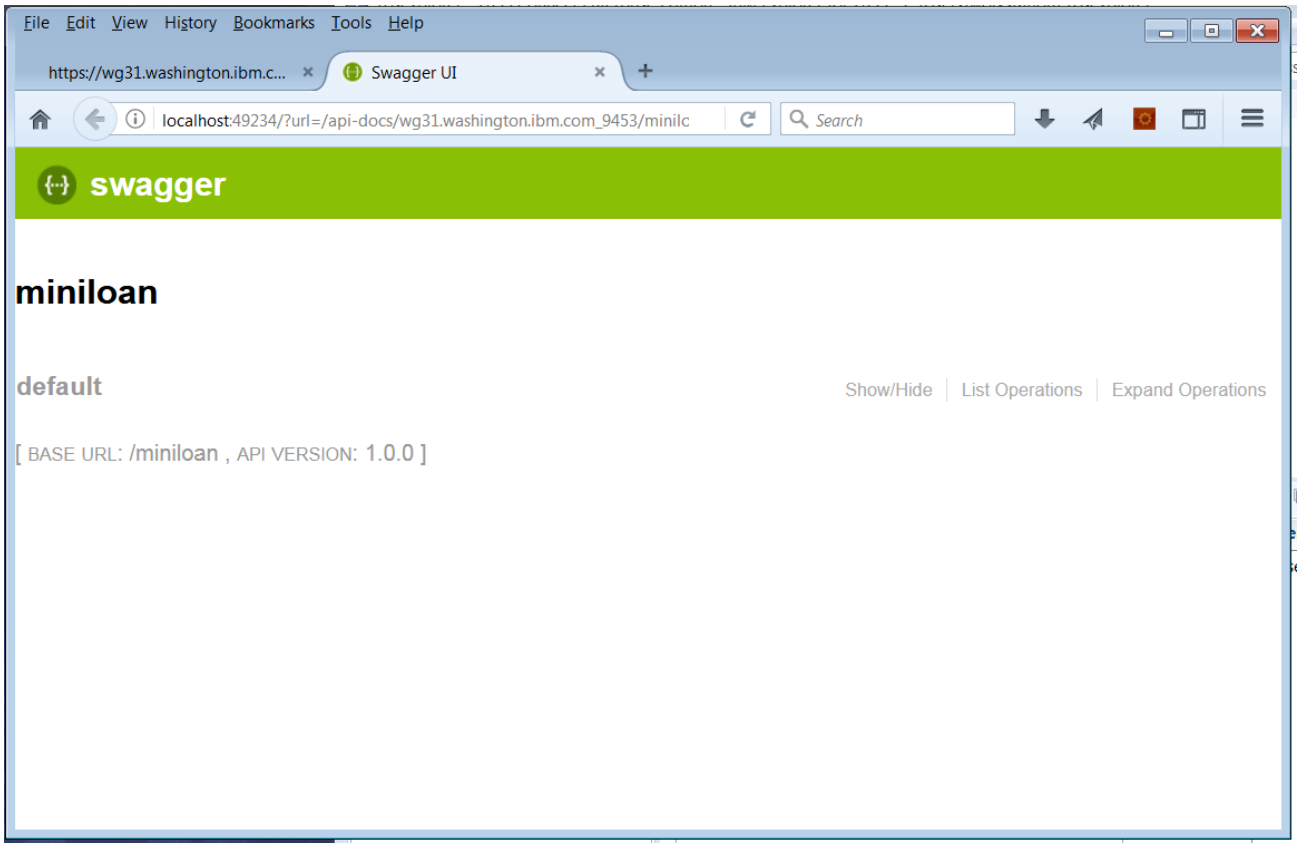


___4. Explore this Swagger document and you will see the results of the request and response mapping performed earlier. This document can be used by a developer or other tooling to develop REST clients for this specific API.

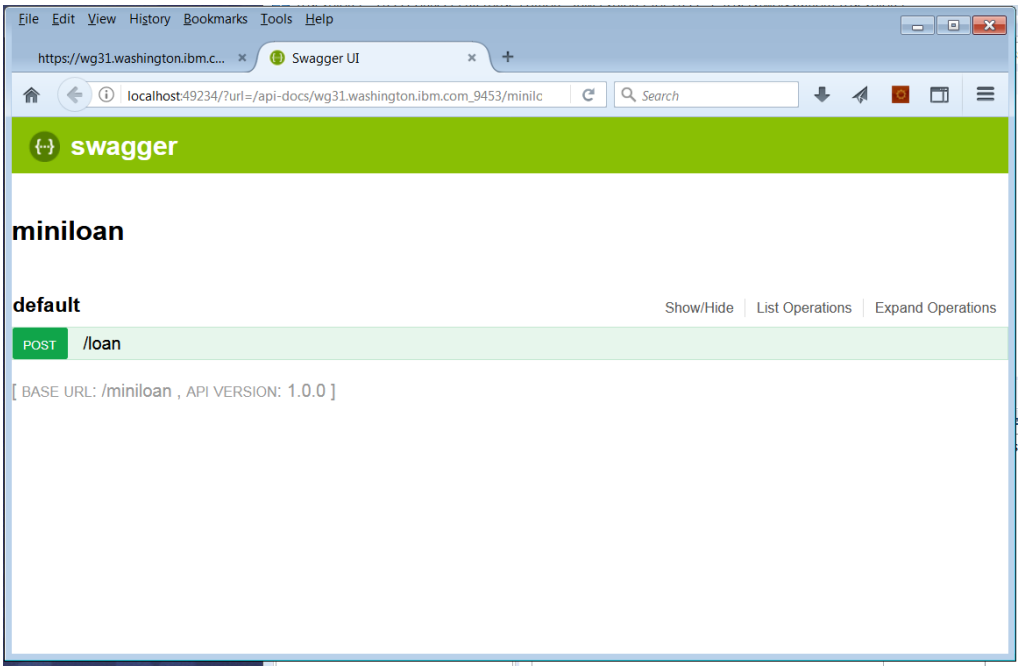
___5. In the lower left-hand side of the *z/OS Connect Explorer* perspective there is view entitled *z/OS Connect EE Servers*. Expand *wg31:9453* and the expand the *APIs* folder. You should see a list of the APIs installed in the server.



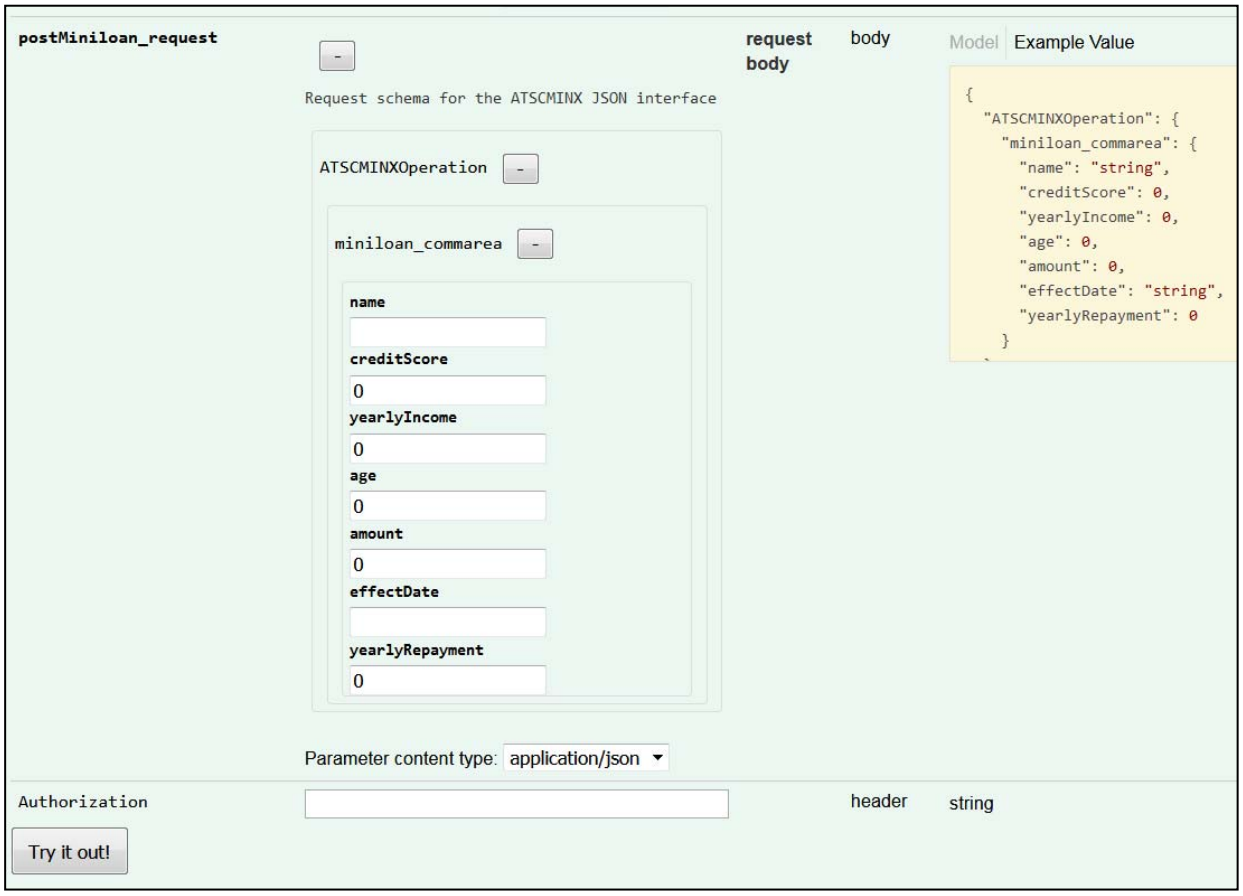
___6. Right click the mouse button on *miniloan* and select *Open in Swagger UI*. Click OK if an informational prompt appears. This will open a Firefox window showing a *Swagger* test client (see below).



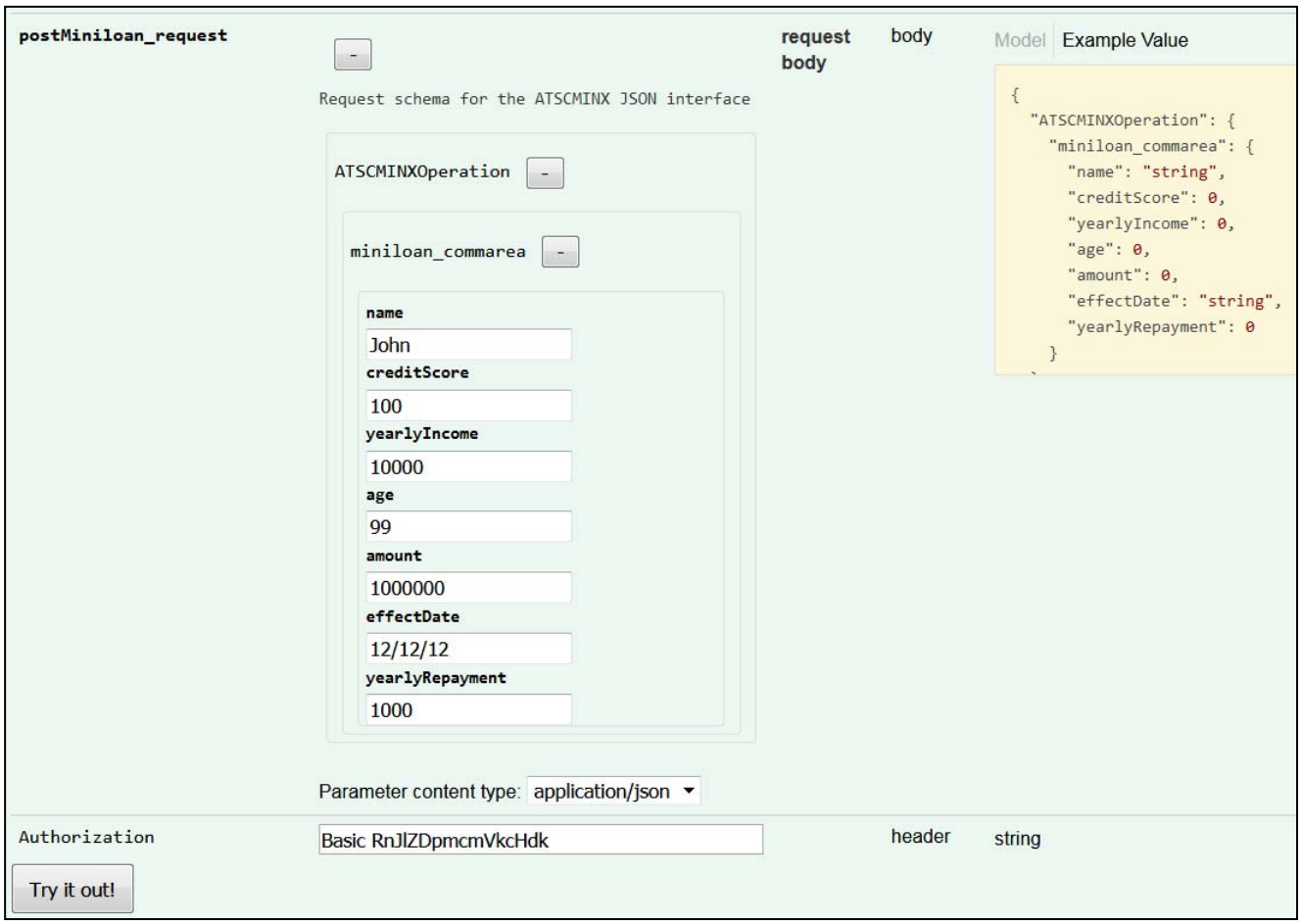
___7. Click the *List Operations* and the browser should show a list of the available HTTP methods like this:



___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:



___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*, ***12/12/12*** in the area under *effectDate* and ***1000*** in the area under *yearlyRepayment*. Finally enter ***Basic RnJlZDpmcmVkcHdk*** in the area beside *Authorization*.



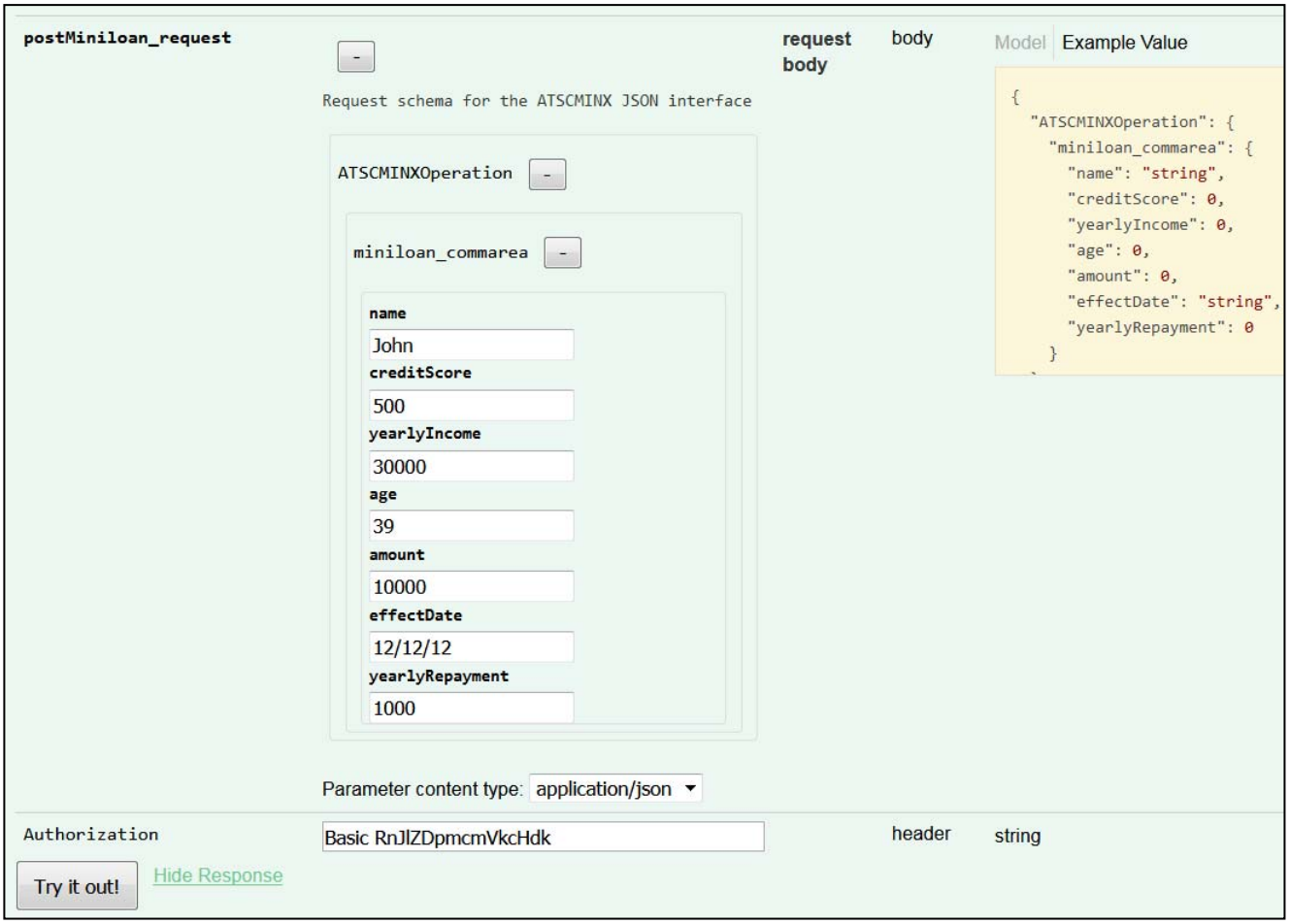
___10. Click **Try it out!**. You should see a *200* in the *Response Code* area and a display of the response from the loan application in the *Response Body.* This is the message from the application returned in the reply queue.

Response Body

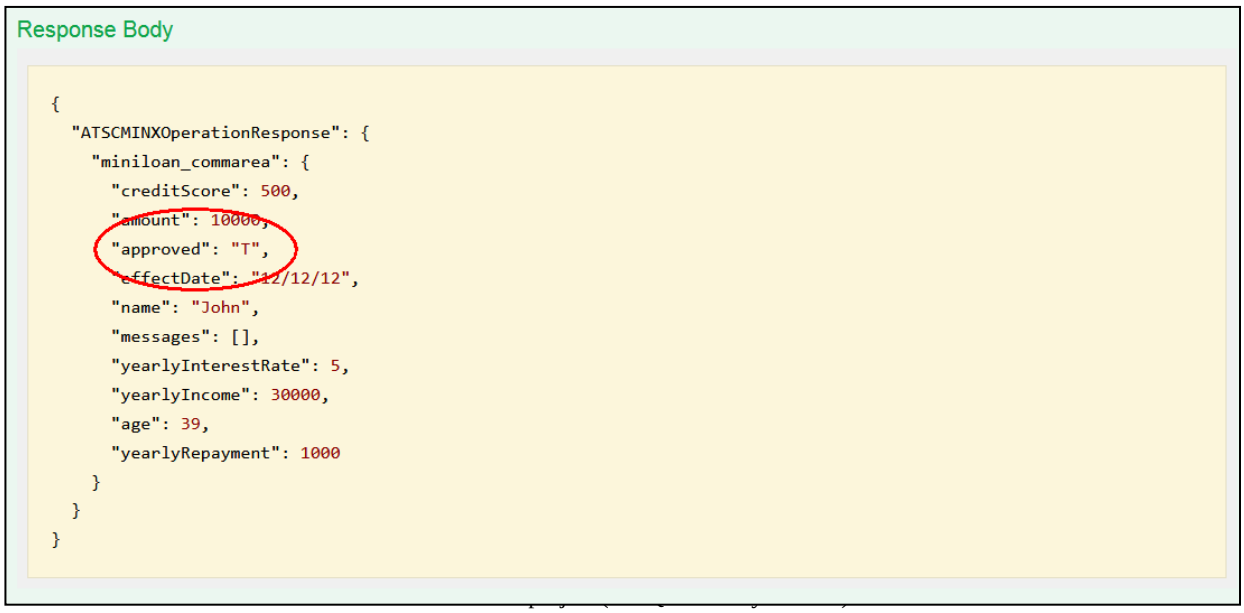

```
    "amount": 1000000,
    "approved": "F",
    "effectDate": "12/12/12",
    "name": "John",
    "messages": [
      {
        "messages": "The age exceeds the maximum"
      },
      {
        "messages": "Credit score below 300"
      },
      {
        "messages": "The yearly income is lower than the basic request"
      }
    ],
    "yearlyInterestRate": 5,
    "yearlyIncome": 10000,
    "age": 99,
    "yearlyRepayment": 1000
  }
```


The loan was not approved (*F* in the *approved* field) and an explanation in the *messages* array.

___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.



___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:

{

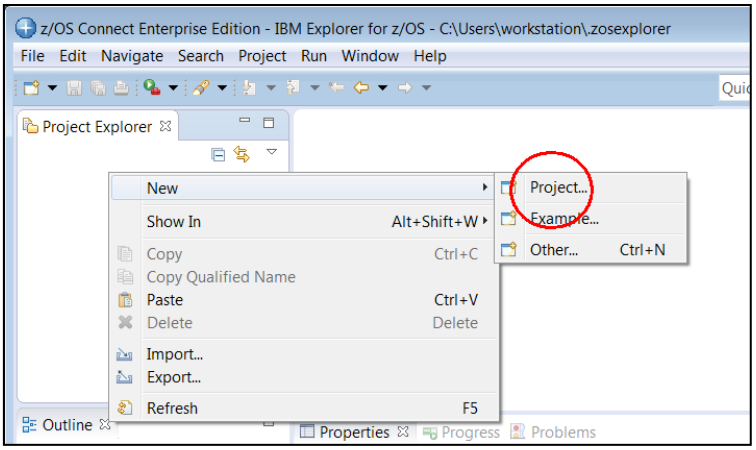
"errorMessage": "BAQR0405I: The asynchronous request under URL https://wg31.washington.ibm.com:9453/miniloan/loan has timed out after 30,000 milliseconds."

}

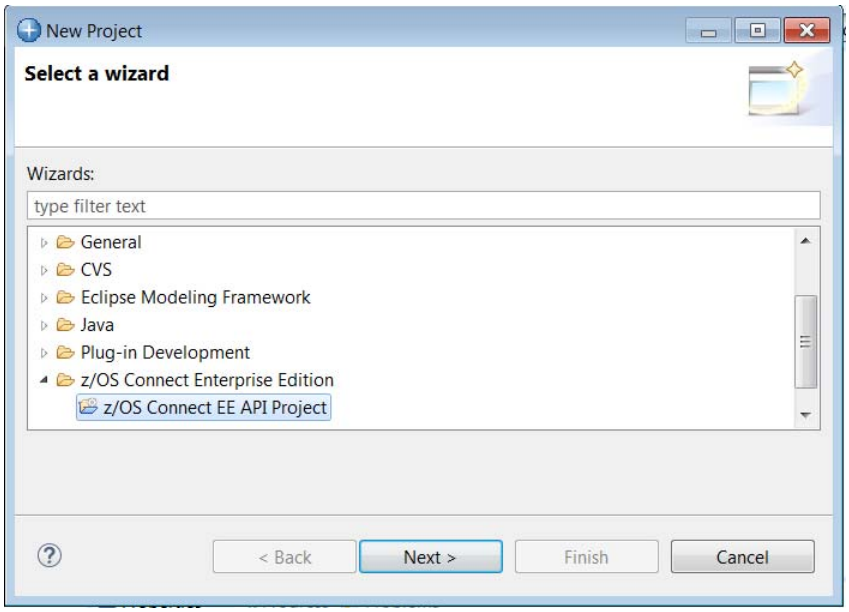
Create the FileaQueue API project (a One-Way Service)

In this section an API that uses a MQ one-way service will be developed. A MQ one-service only supports the **POST**, **GET** and **DELETE** methods. The POST method puts JSON request message on a queue after converting the JSON request message to a non-JSON format. The **GET** (non-destructive) and **DELETE** (destructive) methods are used get messages from a queue and convert the contents to a JSON response message.

___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*:

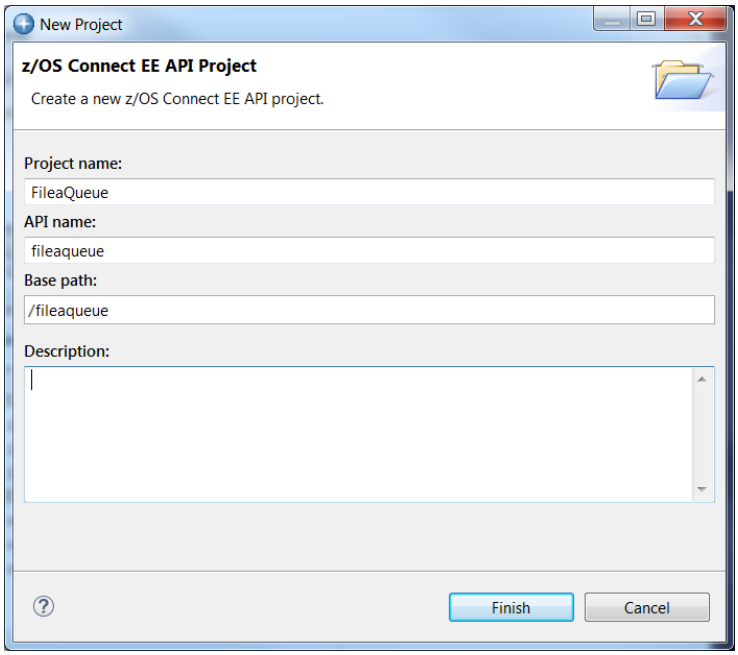


___2. In the *New Project* screen, scroll down and open the *z/OS Connect Enterprise Edition* folder and select *z/OS Connect EE API Project* and then click the **Next** button.



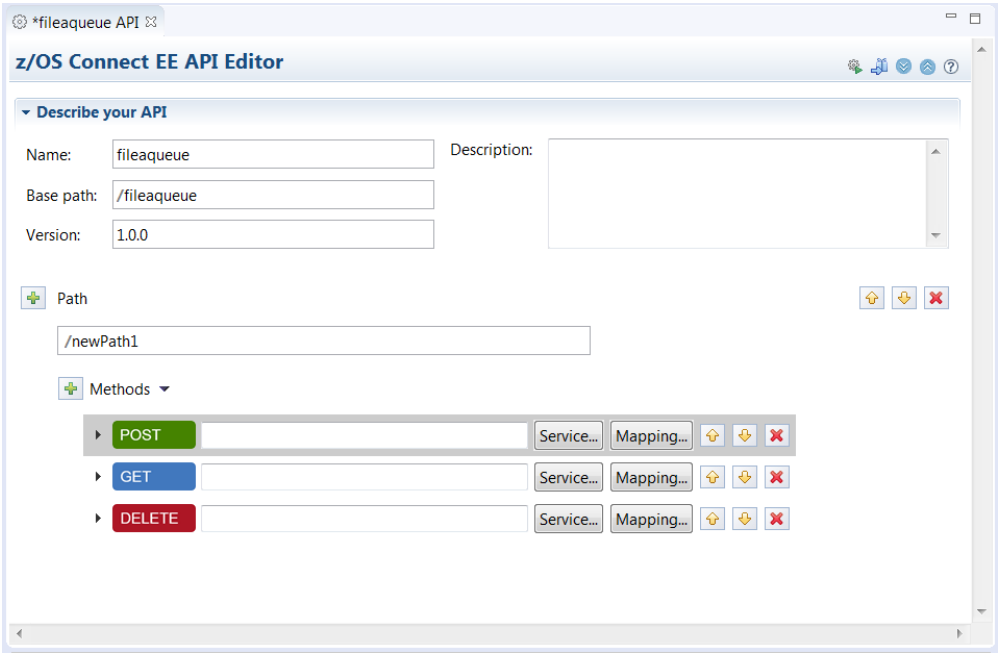
Mitch Johnson (mitchj@us.ibm.com)

___3. Enter ***FileaQueue*** for the *Project name*. Be sure the *API name* is set to ***fileaqueue*** and the *Base path is* set to ***/fileaqueue***. 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.

___4. 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), **GET** (a non-destructive 'get' of a message from a queue), and **DELETE** (a destructive 'get' of a message methods. The **PUT** method is not supported by the *MQ for z/OS Service Provider for z/OS Connect* so it should be deleted**.** The view may need to be adjusted by dragging the view boundary lines.



Summary

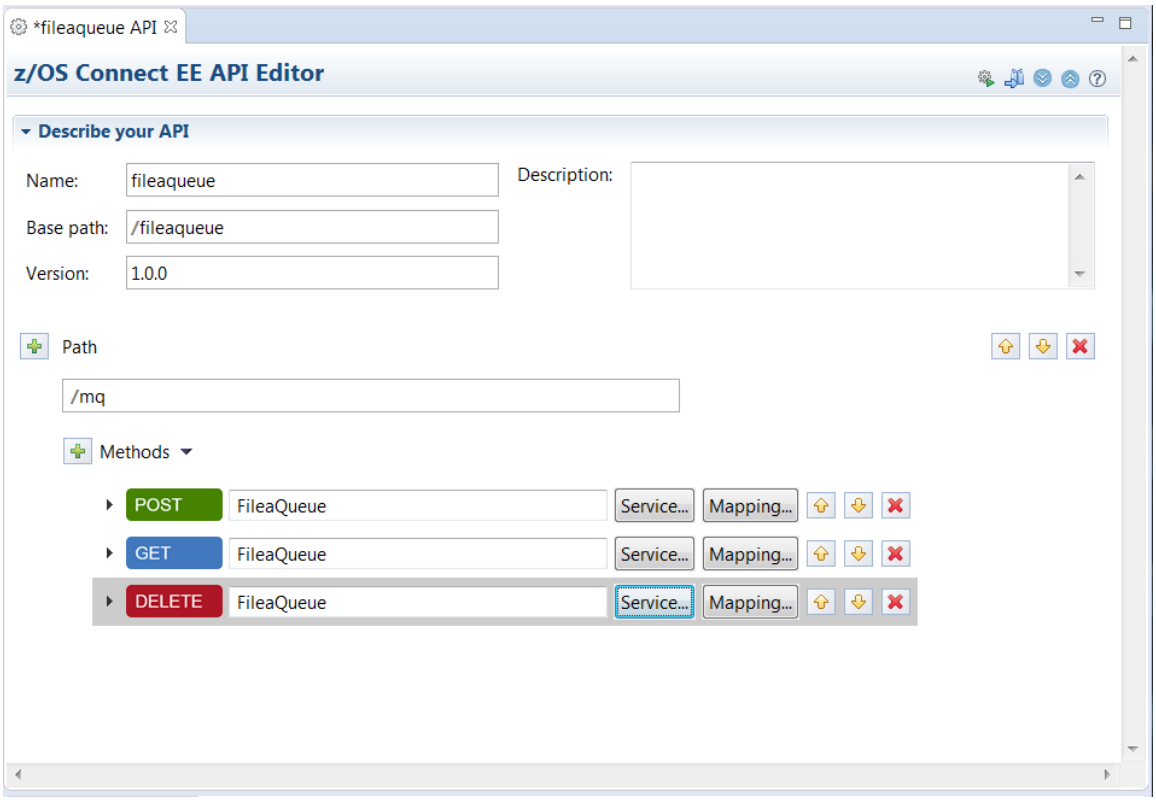
This created the basic framework for the API project in the API editor.

Compose the FileaQueue API

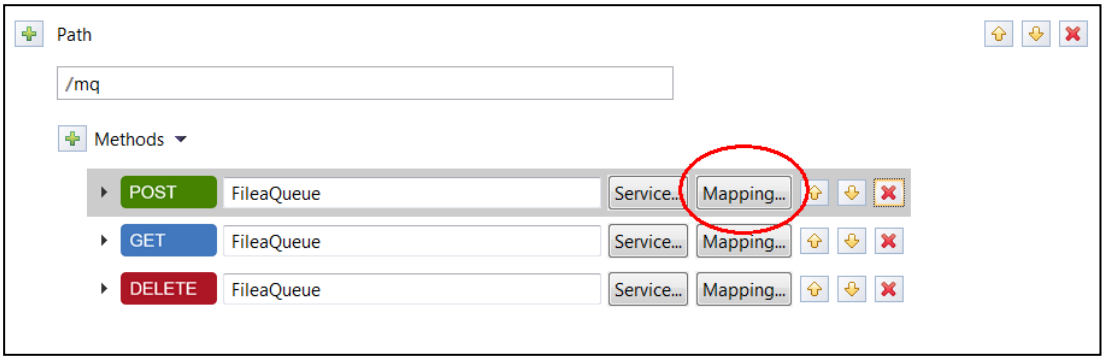
.

___1. Begin by importing the SAR file, *FileaQueue.sar*, into the workspace for API project *Miniloan*. Click on the **Service** button to the right of the **POST** method. Then on the *Select a z/OS Connect EE Service* window click on the **Workspace** button. Expand the *services* folder and select *FileaQueue.sar*. Click **OK** three times. This will populate the field to the right of the method. This has imported this SAR file into the *Fileaqueue* project (expand the new *services* folder in *Miniloan).*

___2. Start by entering ***/mq*** 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 click on the **Workspace** button. Expand the *services* folder and select *FileaQueue.sar*. Click **OK** three times. This will populate the field to the right of the method. This has imported this SAR file into the *FileaQueue* project (expand the new *services* folder in *FileaQueue* Repeat this process for the **GET** and **POST** methods.



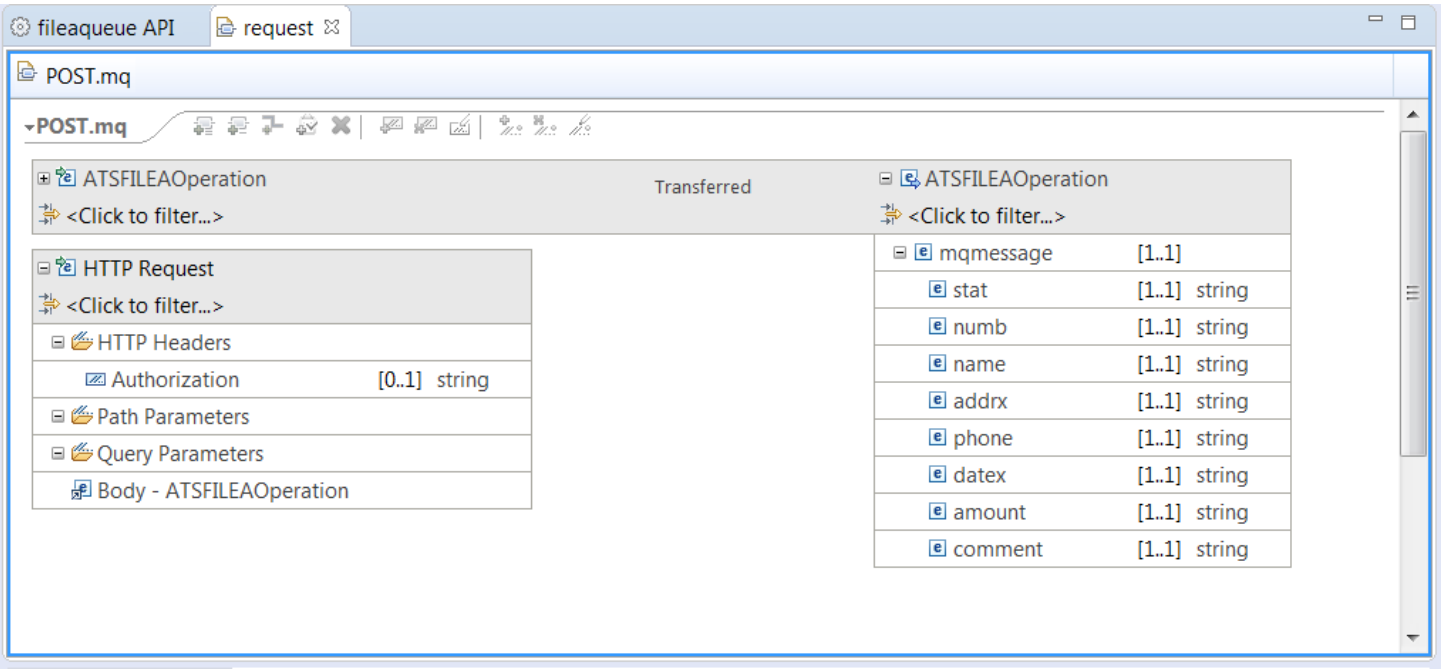
___3. Next, click on the **Mapping** button beside the **POST** method and then select *Open Request Mapping*:



___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 *mqmessage*. You should see fields that correspond to the fields defined in the original COBOL copy book *MINILOAN* in *USER1.ZCEE.CNTL*.

```
       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).
```

___6. 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.

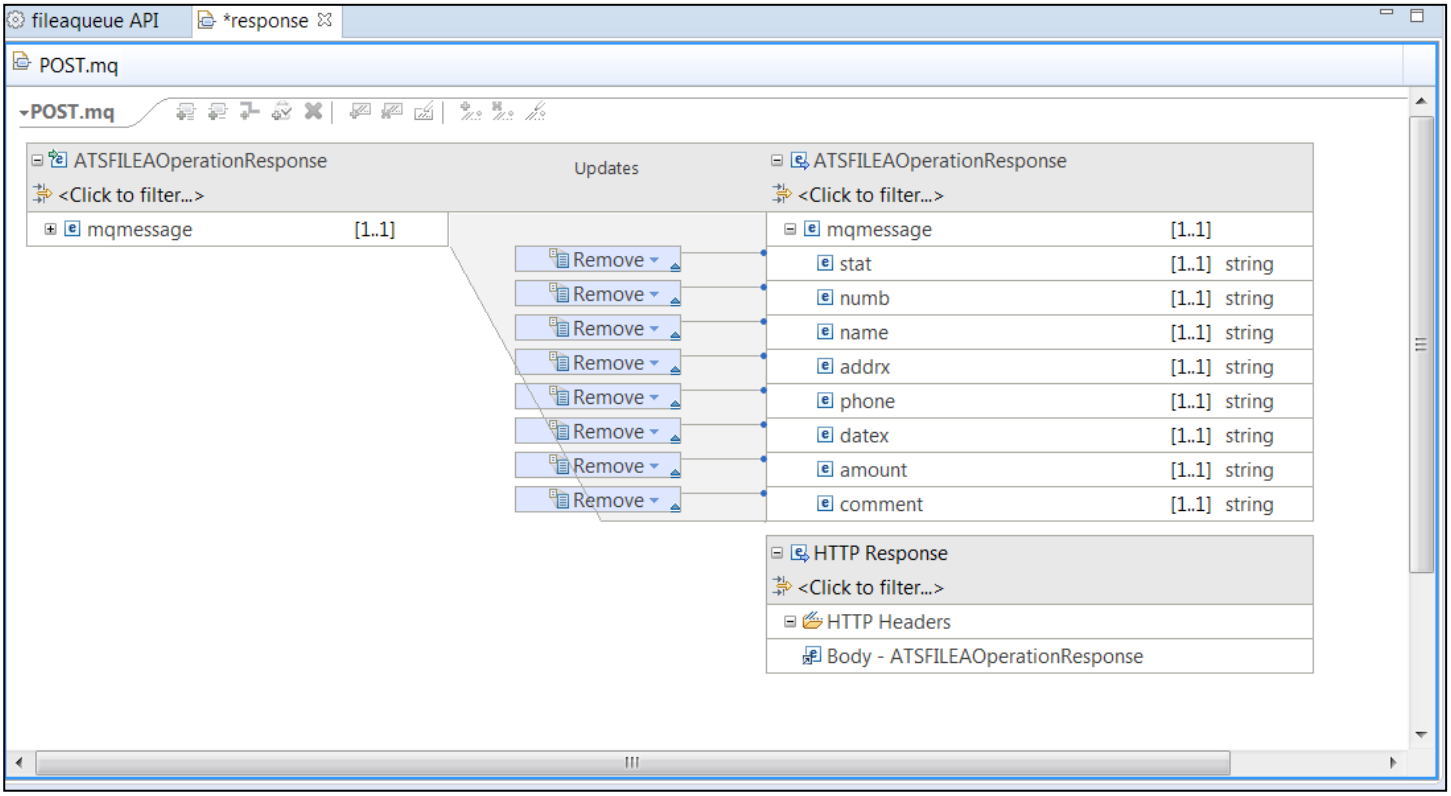


___7. Use the **Ctrl-S** key sequence to close this view.

___8. Next, click on the **Mapping** button beside the **POST** method and then select *Open Response Mapping*.

___9. Use the slider bar to fully expose the *mqmessage* structure. Use the left mouse button and draw a dotted line box that fully includes the *stat*, *numb*, *name*, *addres*, *phone*, *datex*, *amount* and *comment* fields When you release the button all of these fields should be selected (the background should be blue).

___10. Right click the mouse button on one of the selected fields and select the *Add Remove transform* option to remove all of these fields from the response. The MQ Service Provider does not return a response message from a **POST** request for a one-way service.

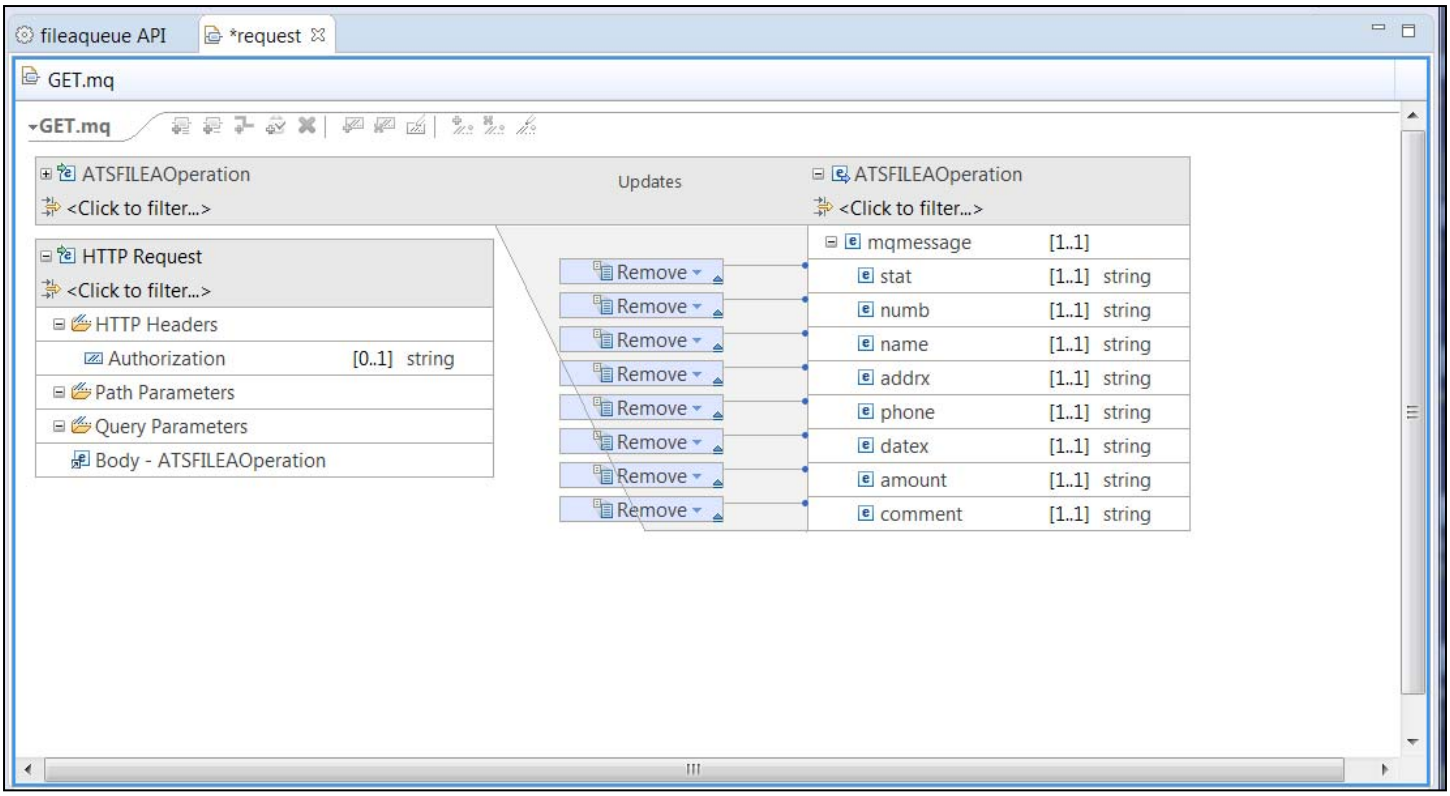


___11. Use the **Ctrl-S** key sequence to close this view.

___12. Next, click on the **Mapping** button beside the **GET** method and then select *Open Request Mapping*.

___13. Use the slider bar to fully expose the *mqmessage* structure. Use the left mouse button and draw a dotted line box that fully includes the *stat*, *numb*, *name*, *addres*, *phone*, *datex*, *amount* and *comment* fields. When you release the button all of these fields should be selected (the background should be blue).

___14. Right click the mouse button on one of the selected fields and select the *Add Remove transform* option to remove all of these fields from the response. The MQ Service Provider does not require a request message from a **GET** request.



___15. Fields can be removed from the response message 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.

___16. Repeat the steps performed for the request message of the **GET** method for the request message of the **DELETE** method. Like the **GET** method the **DELETE** method does not require a request message.

Summary

You created the API, which just a base path and with the **POST GET** and **DELETE** 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 ***Miniloan*** service is added to the z/OS Connect server's *server.xml* file by the including the *miniloan.xml* file (see below).

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

 <featureManager>
       <feature>jms-2.0</feature>
       <feature>mqzosconnect:zosConnectMQ-2.0</feature>
       <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"/>

 <zosconnect_zosConnectService id="fileaqueue"
         invokeURI="/FileaQueue"
         dataXformRef="xformJSON2Byte"
         serviceName="FileaQueue"
         serviceDescription="MQ Oneway Service"
         serviceRef="FileaQueue" />

 <zosconnect_zosConnectService id="miniloan"
         dataXformRef="xformJSON2Byte"
         serviceName="Miniloan"
         serviceDescription="MQ Reply/Response Service"
         serviceRef="Miniloan" />

  <mqzosconnect_mqzOSConnectService id="FileaQueue"
         connectionFactory="jms/qmgrCf"
         destination="jms/default" />

  <mqzosconnect_mqzOSConnectService id="Miniloan"
         connectionFactory="jms/qmgrCf"
         waitInterval="30000"
         destination="jms/request"
         replyDestination="jms/response"/>

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

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

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

1 2 3 4 5

Mitch Johnson (mitchj@us.ibm.com)

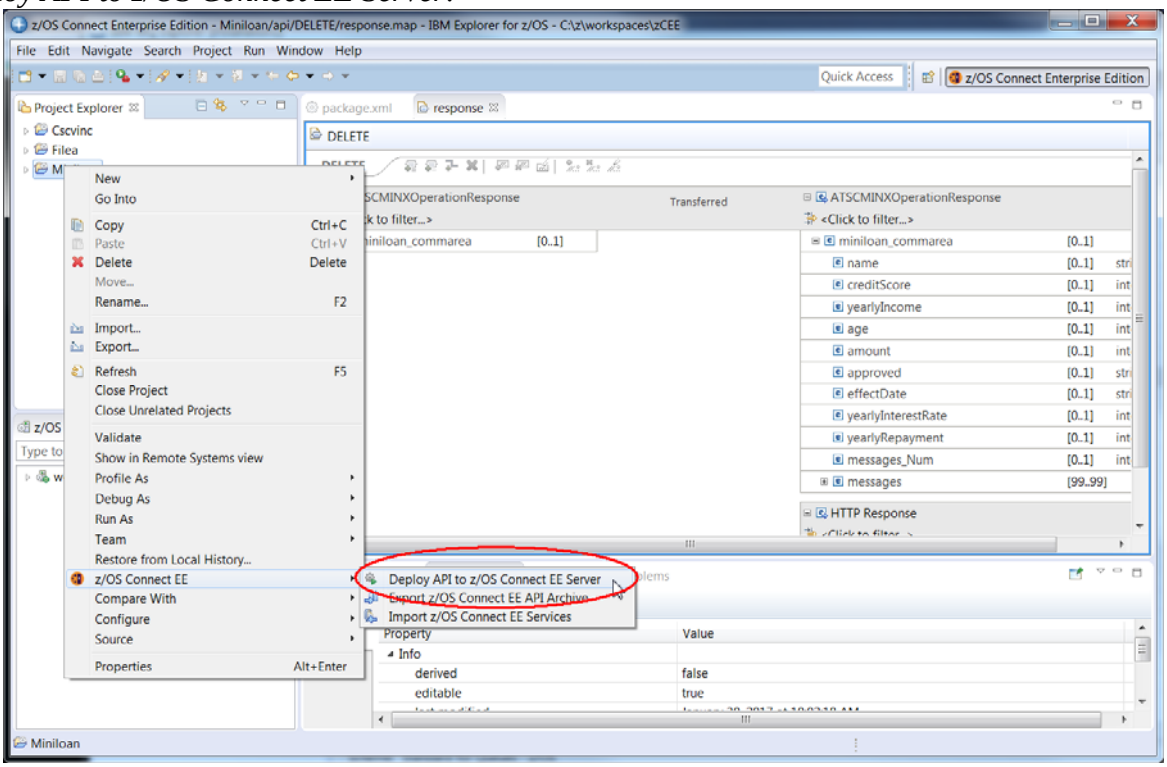
```
<jmsQueue id="request" jndiName="jms/request">
     <properties.wmqJms
         baseQueueName="ZCONN2.TRIGGER.REQUEST"
         targetClient="MQ"
         CCSID="37"/>
 </jmsQueue>

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

Figure 2 - miniloan.xml

Tech-Tip: The *mqzosconnect_mqzOSConnectService* elements identify which queue is being access by each service using the JNDI names in the *destination* and/or the *replyDestination* attributes.

6. The *featureManager* element identifies the Liberty features required by the *MQ for z/OS Service Provider for z/OS Connect.*
7. The *zosConnectService* element provides security constraints (none in this case) and the identity of the *dataXformRef* element that provides the location of API artifacts.
8. The *mqzOSConnectServices* element identifies the JMS connection factories, the JMS destinations (queues) by JNDI name and other JMS characteristics required by each service.
9. The *jmsConnectionFactory* element associates the JMS connection factory(*jndiName*) with the target queue manager and details on how to connect to this queue manager.
10. The jmsQueue elements provide details that associate the JMS destination (*jndiName*) with the target queue (*baseQueueName*) and its JMS/MQ properties.

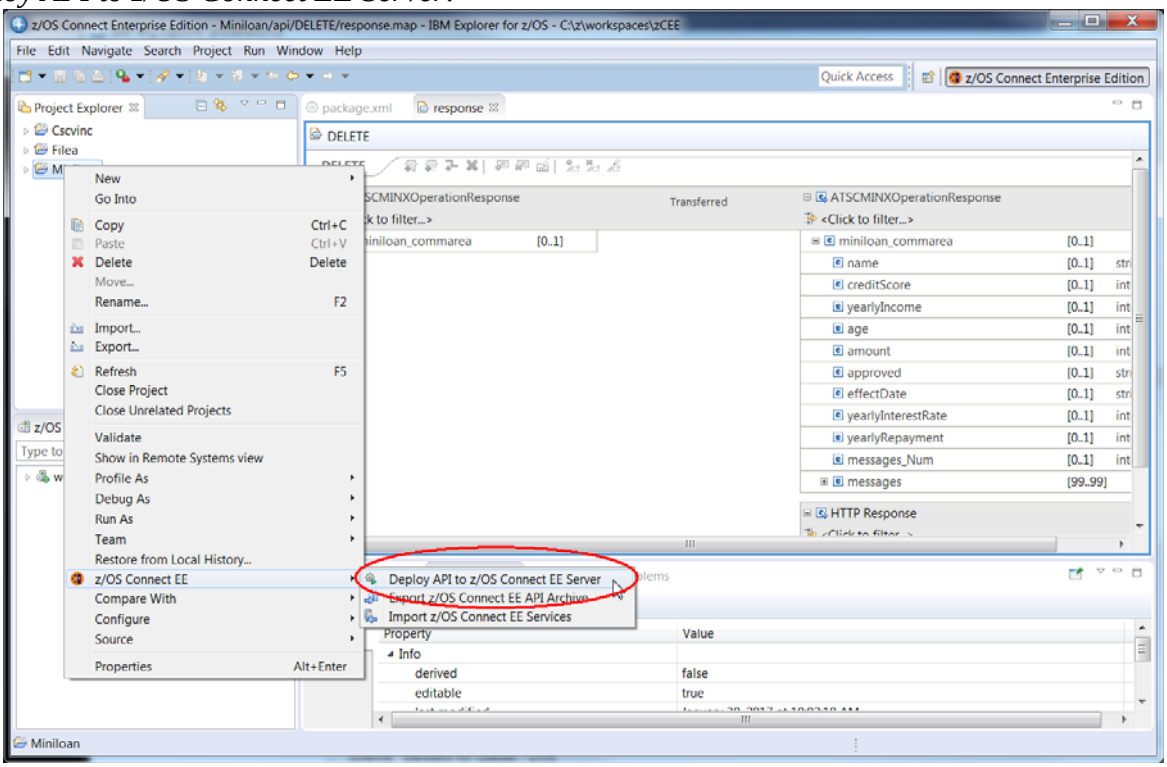
___6. In the *Project Explorer* view (upper left), right-mouse click on the *FileaQueuey* folder, then select *z/OS Connect EE→ Deploy API to z/OS Connect EE Server*.



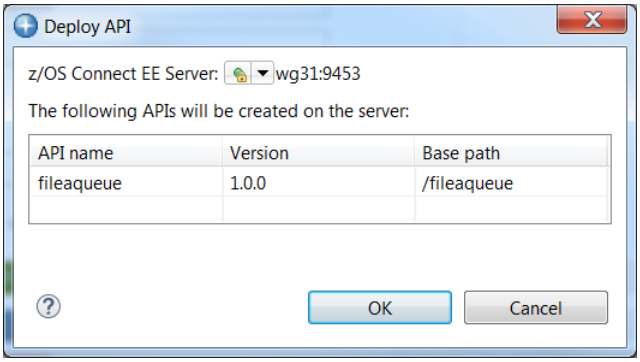
___7. Since z/OS Explorer is connected to only one z/OS Connect server there is only one choice (*wg31:9453*). If z/OS Explorer had multiple host connections to z/OS Connect servers then the pull-down arrow would allow a selection to which server to deploy. Click **OK** on this screen to continue.

Mitch Johnson (mitchj@us.ibm.com)

___8. In the *Project Explorer* view (upper left), right-mouse click on the *FileaQueue* folder, then select *z/OS Connect EE→ Deploy API to z/OS Connect EE Server*.

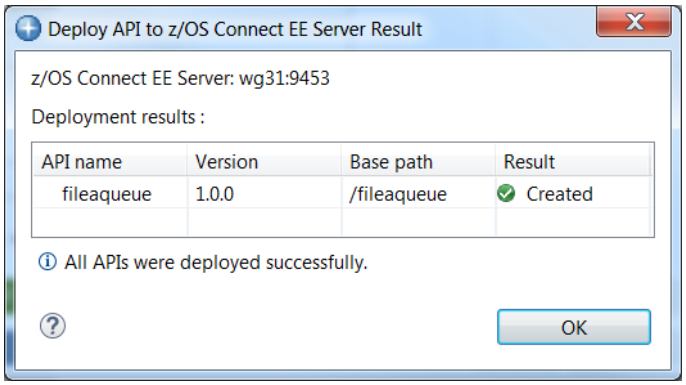


___9. Since z/OS Explorer is connected to only one z/OS Connect server there is only one choice (*wg31:9453*). If z/OS Explorer had multiple host connections to z/OS Connect servers then the pull-down arrow would allow a selection to which server to deploy. Click **OK** on this screen to continue.



Mitch Johnson (mitchj@us.ibm.com)

___10. The API artifacts will be transferred to z/OS and copied into the */var/ats//zosconnect/servers/zceesrvr/resources/zosconnect/apis* directory.

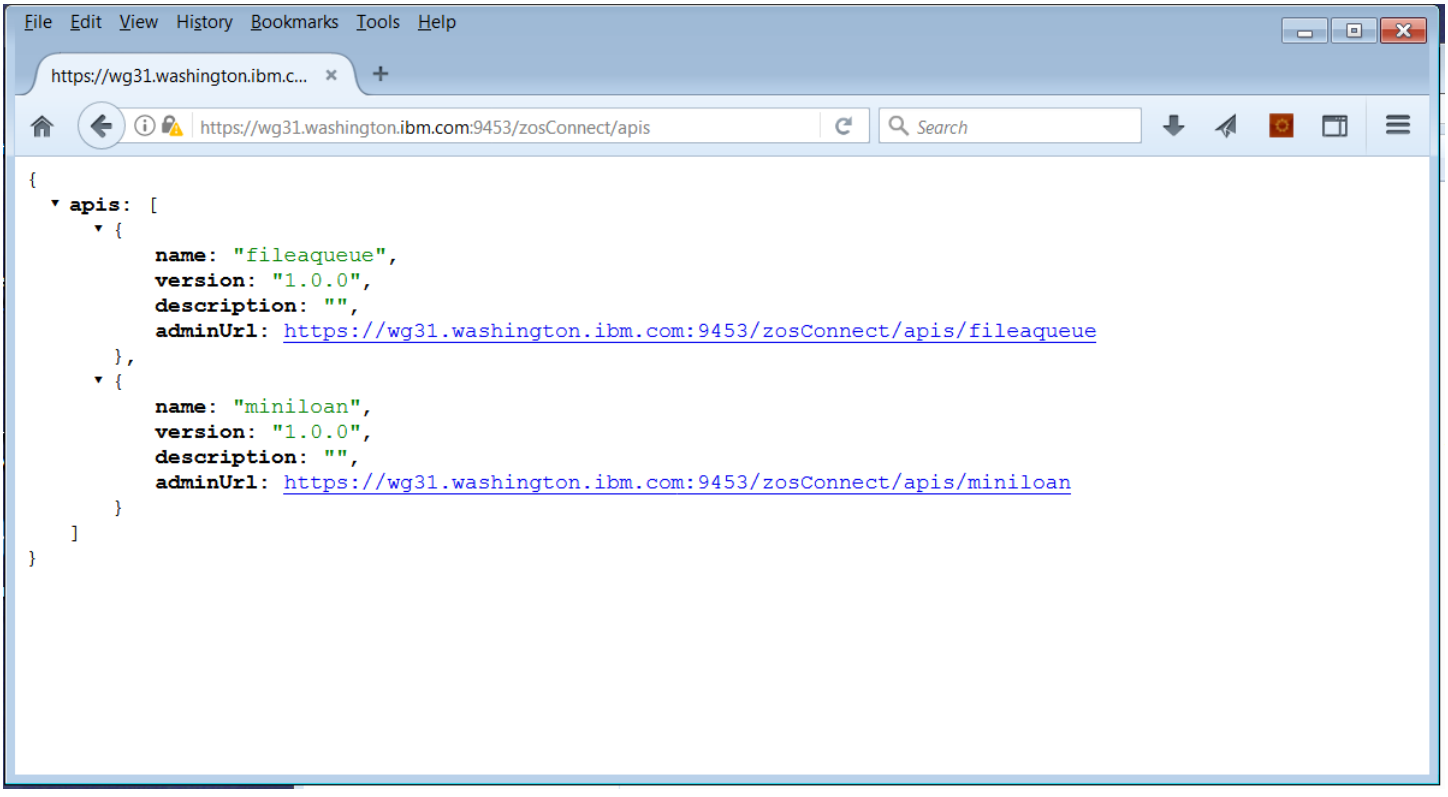


Test the FileaQueue MQ One Way Service

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** (nondestructive get of a message), and DELETE (destructive get of a message). Remember, the **PUT** method is not supported by the *MQ for z/OS Service provider for z/OS Connect* in a one-way service.

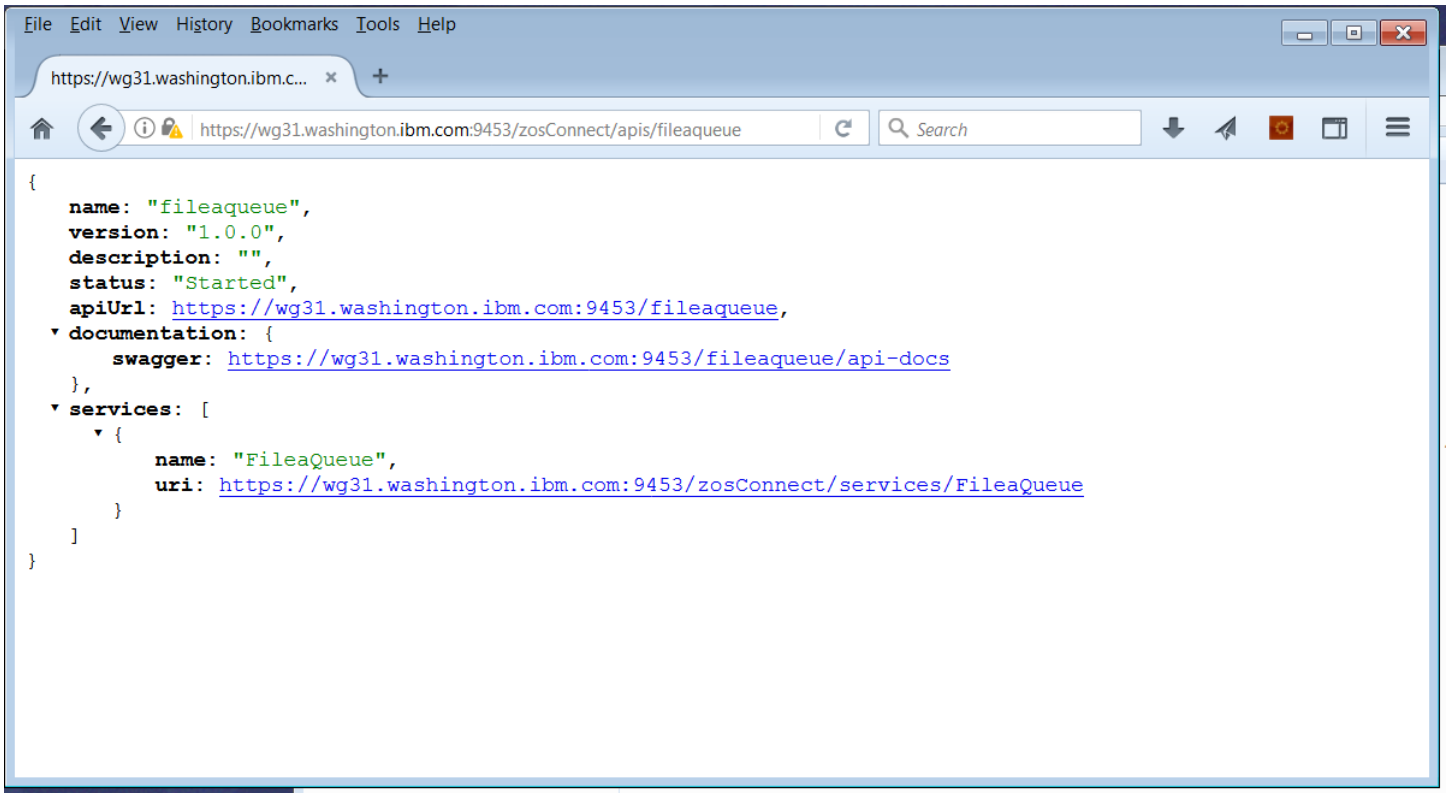
___2. Enter URL ***https://wg31.washington.ibm.com:9453/zosConnect/apis*** in the Firefox browser and you should see the window below. The API *fileaqueue* is now displayed. This is because this API was just deployed to this server.

. **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.

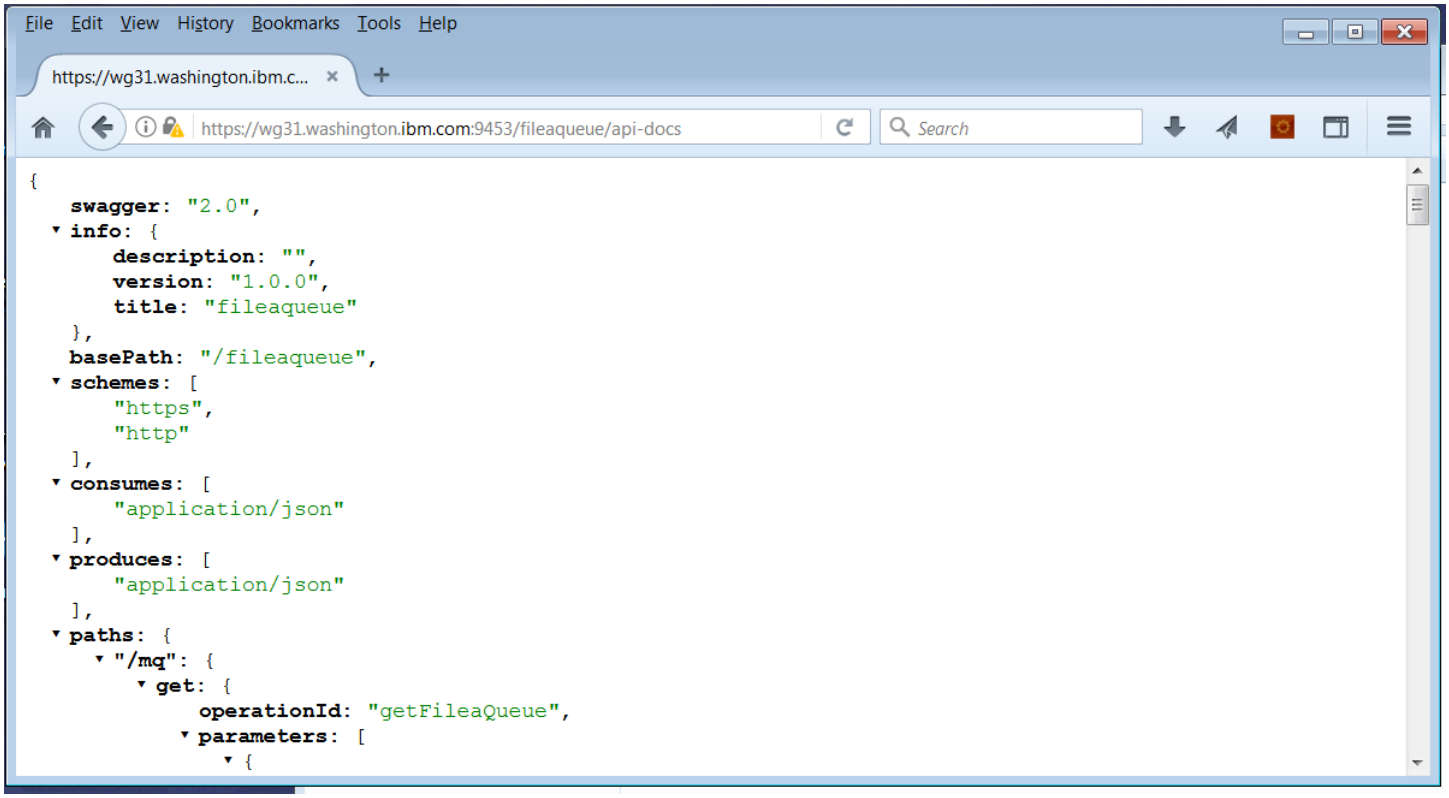


Tech Tip: It is very important to access the z/OS Connect server from a browser prior to any testing using the Swagger UI. Accessing a z/OS Connect URL from a browser starts an SSL handshake between the browser and the server. If this handshake has not performed prior to performing any test the test 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.

___3. If you click on *adminUrl* URL the window below should be displayed.

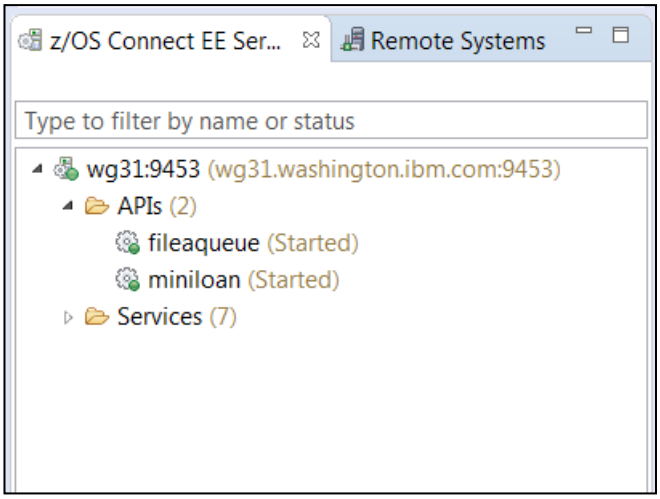


___4. Finally click on the *swagger* URL and you should see the Swagger document associated with this API.

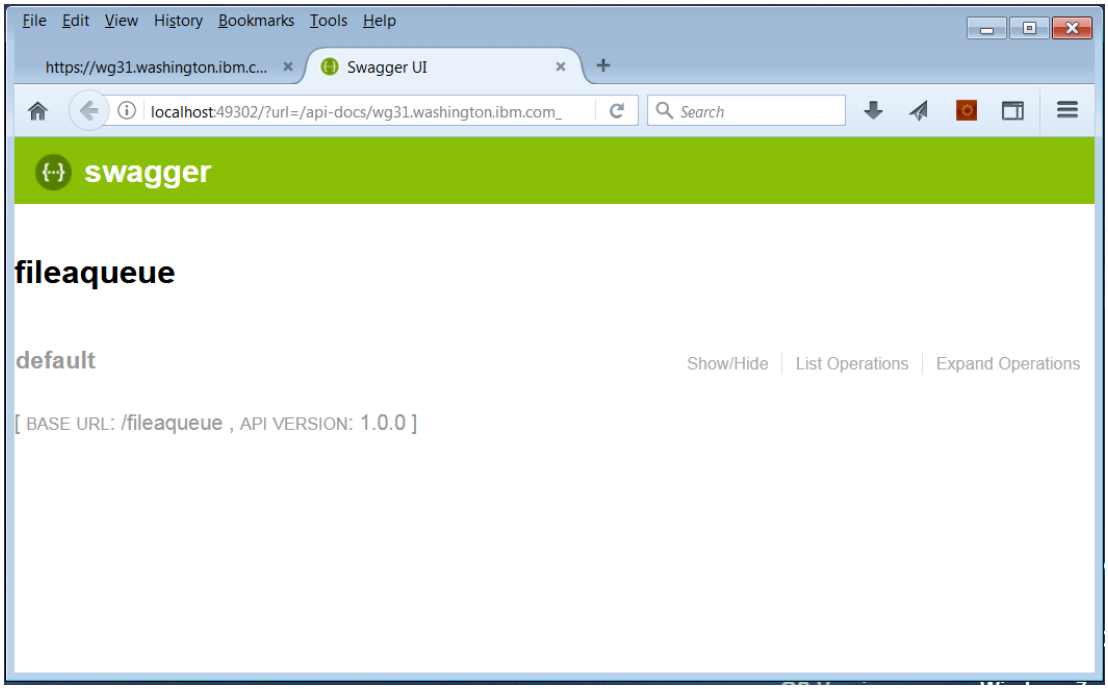


___5. Explore this Swagger document and you will see the results of the request and response mapping performed earlier. This document can be used by a developer or other tooling to develop REST clients for this specific API.

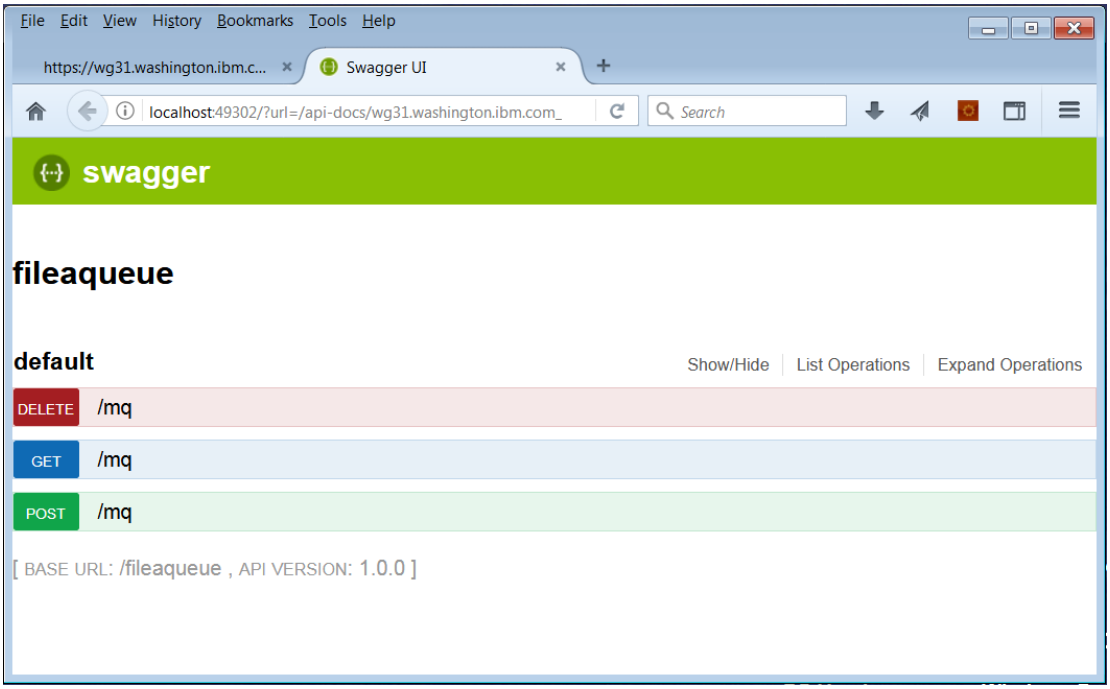
___6. In the lower left-hand side of the *z/OS Connect Explorer* perspective there is view entitled *z/OS Connect EE Servers*. Expand *wg31:9453* and the expand the *APIs* folder. You should see a list of the APIs installed in the server.



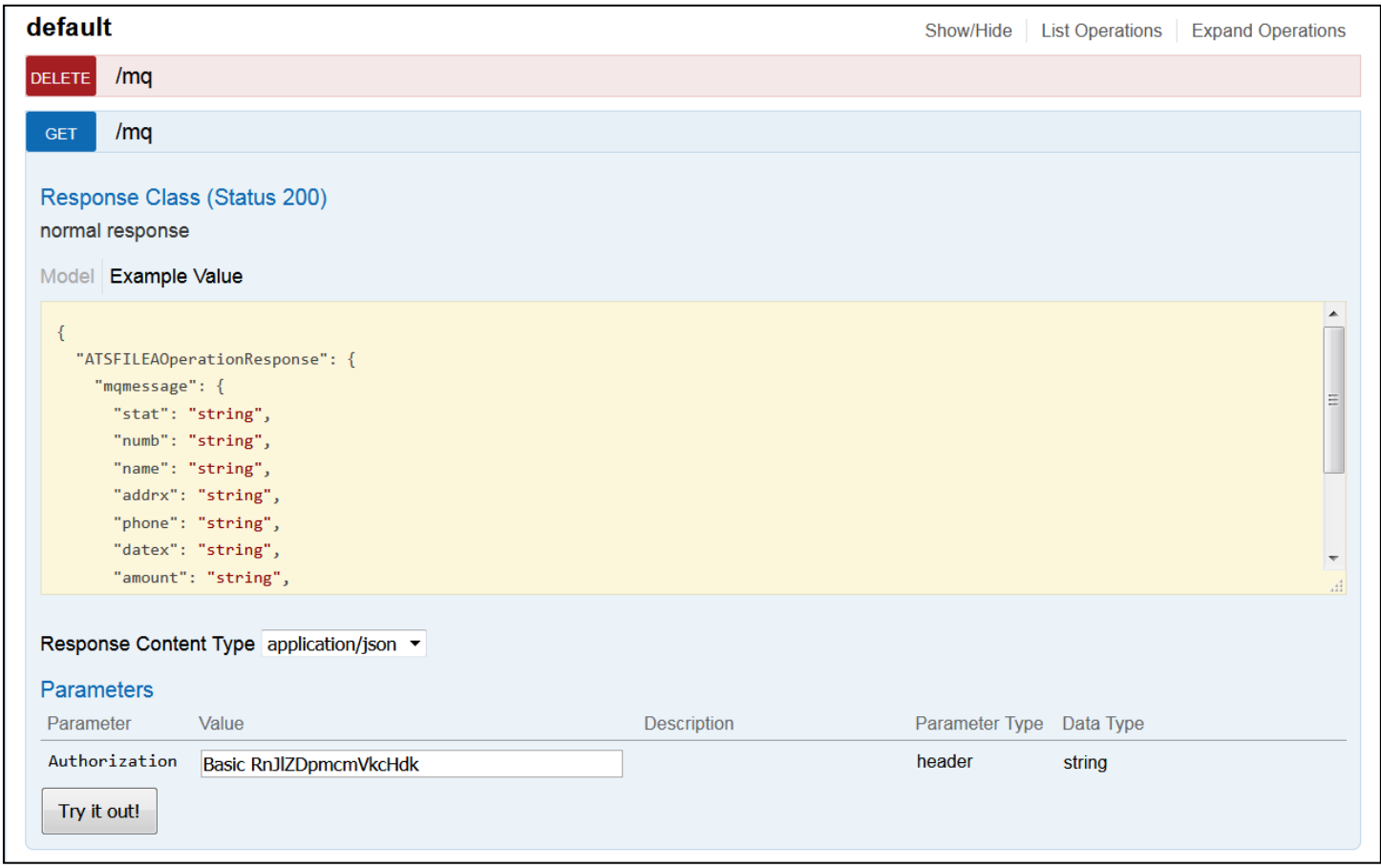
___7. Right click the mouse button on *fileaqueue* and select *Open in Swagger UI*. Click **OK** if an informational prompt appears. This will open a Firefox window showing a *Swagger* test client (see below).



___8. Click the *List Operations* and the browser should show a list of the available HTTP methods for this API.

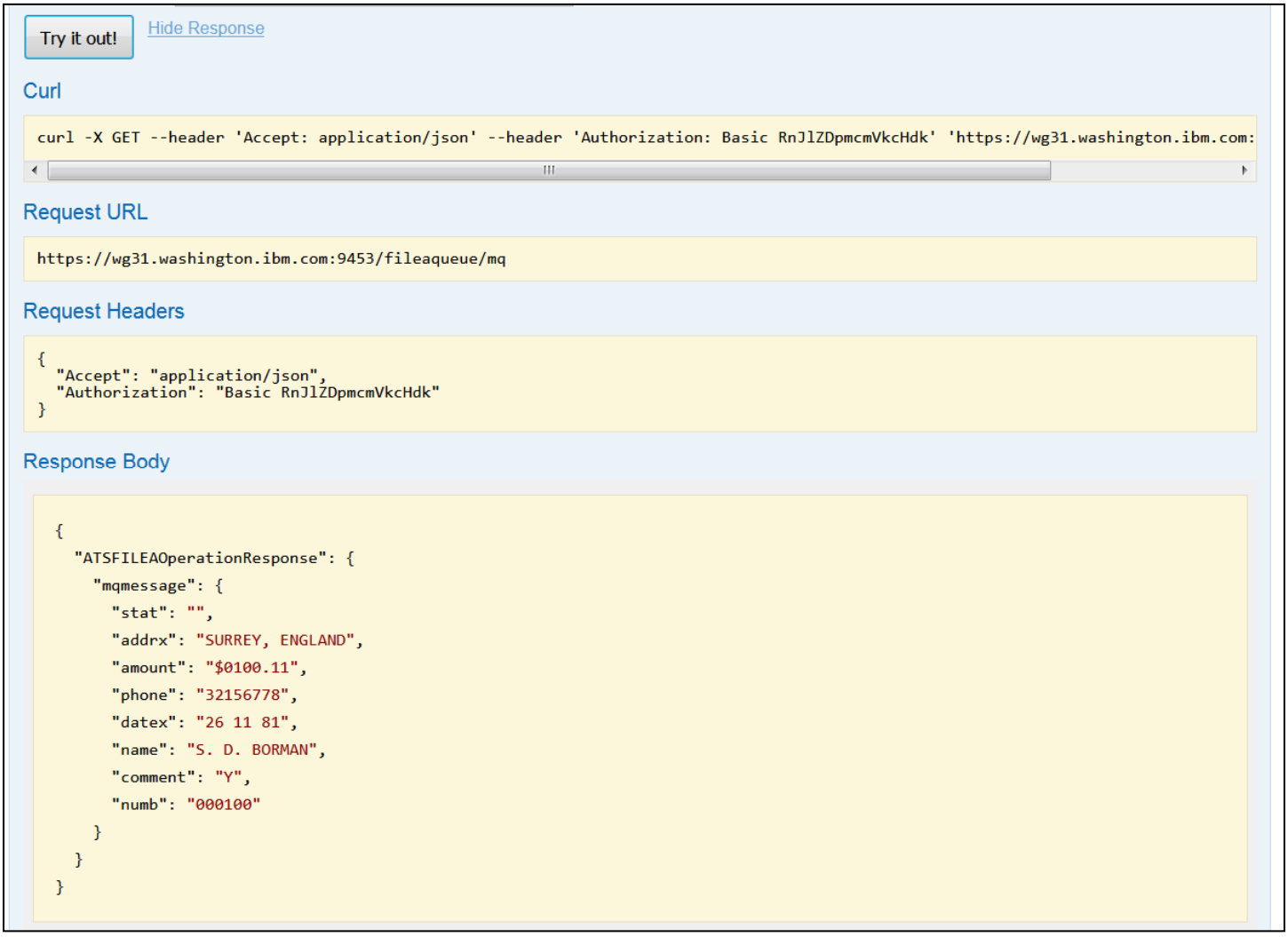


___9. Expand the **GET** method by clicking on the path beside it (e.g. */mq)* and scroll down until the method *Parameters* are displayed as shown below:

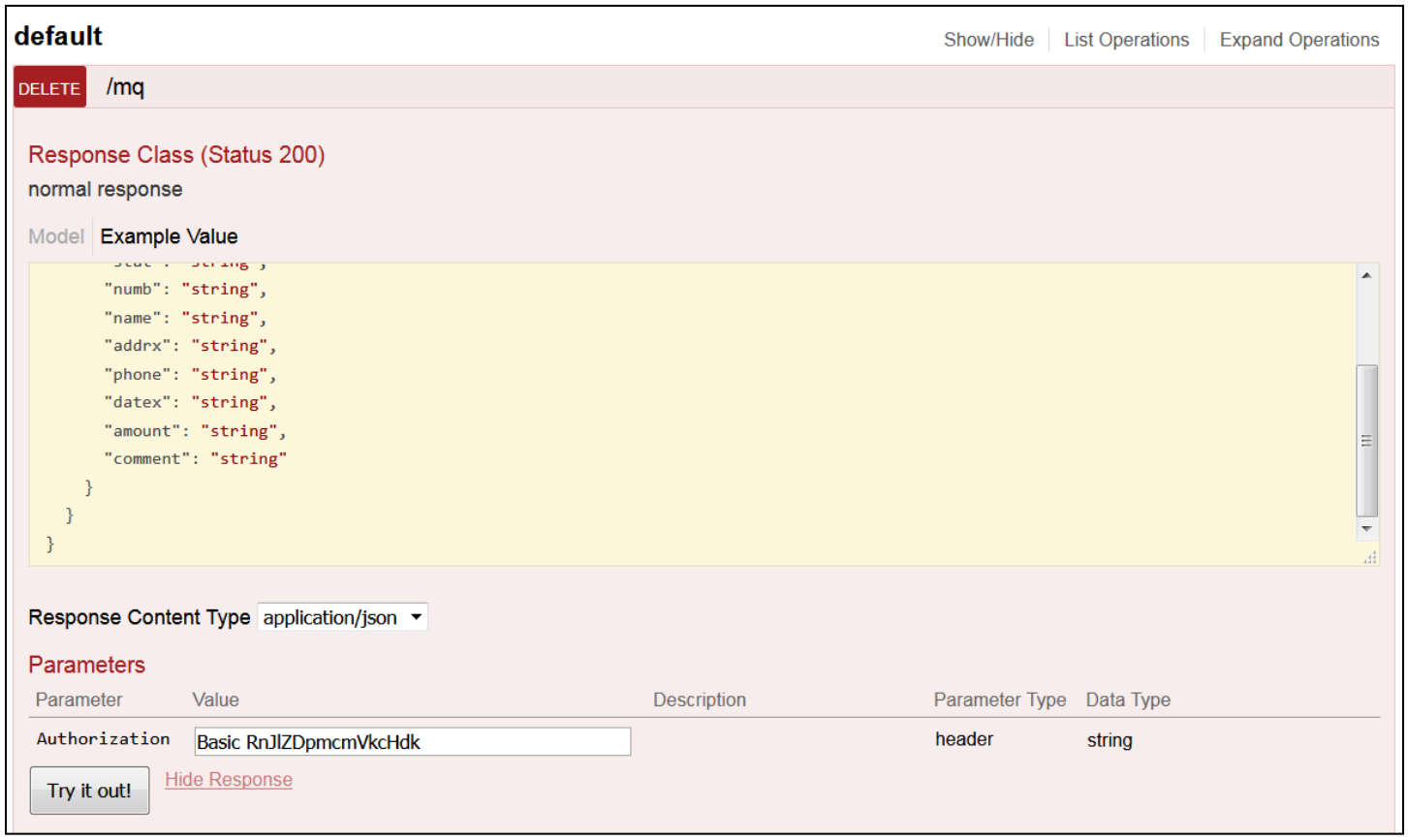


Mitch Johnson (mitchj@us.ibm.com)

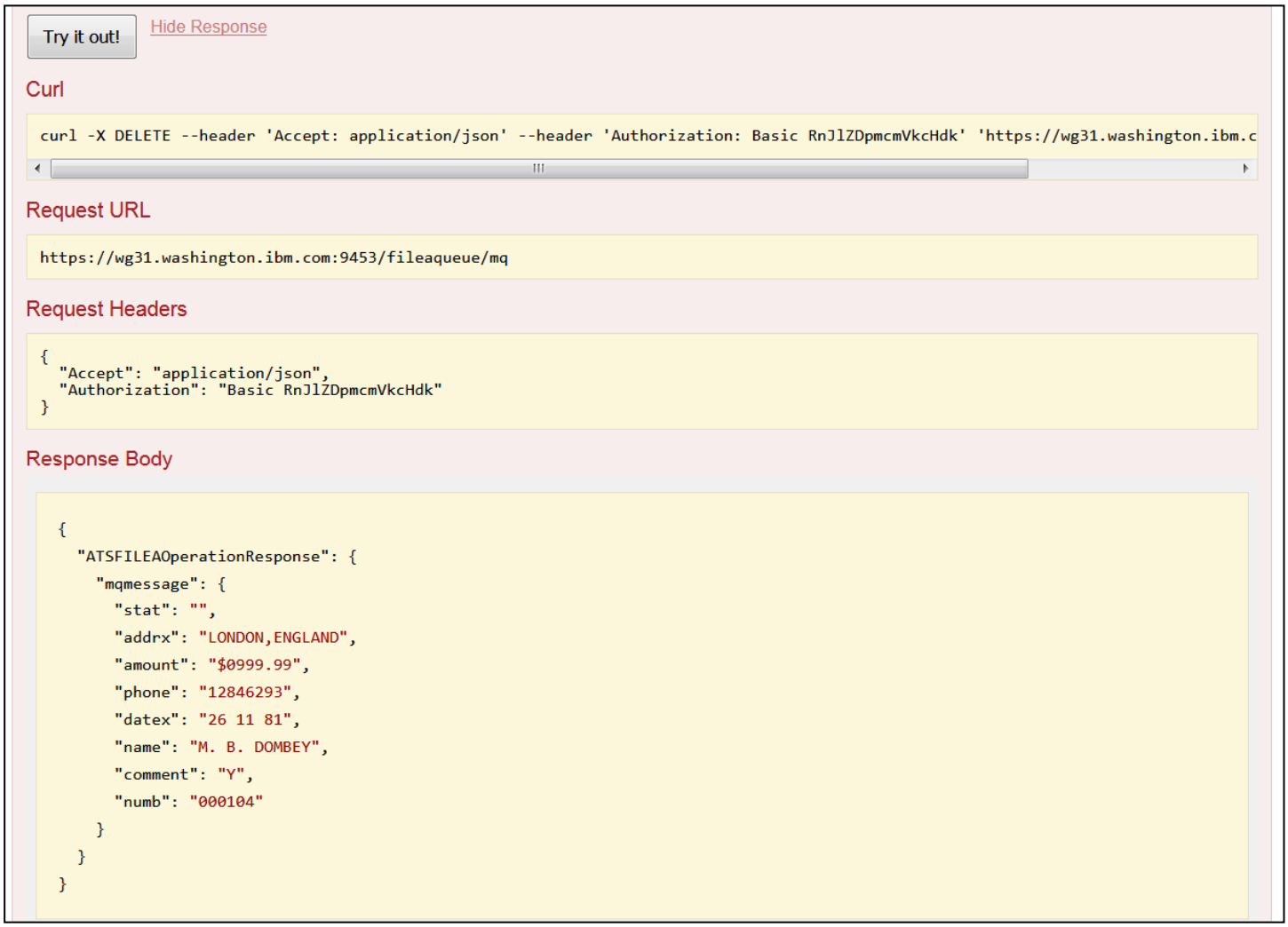
___10. Enter ***Basic RnJlZDpmcmVkcHdk*** in the area beside *Authorization* and click **Try it out!**. Scroll down to display the response in the *Response Body.* This is the message from the application returned in the reply queue. This was the message retrieved by a non-destructive get. If you continue to click the **Try it out!** button you will see the same message over and over.



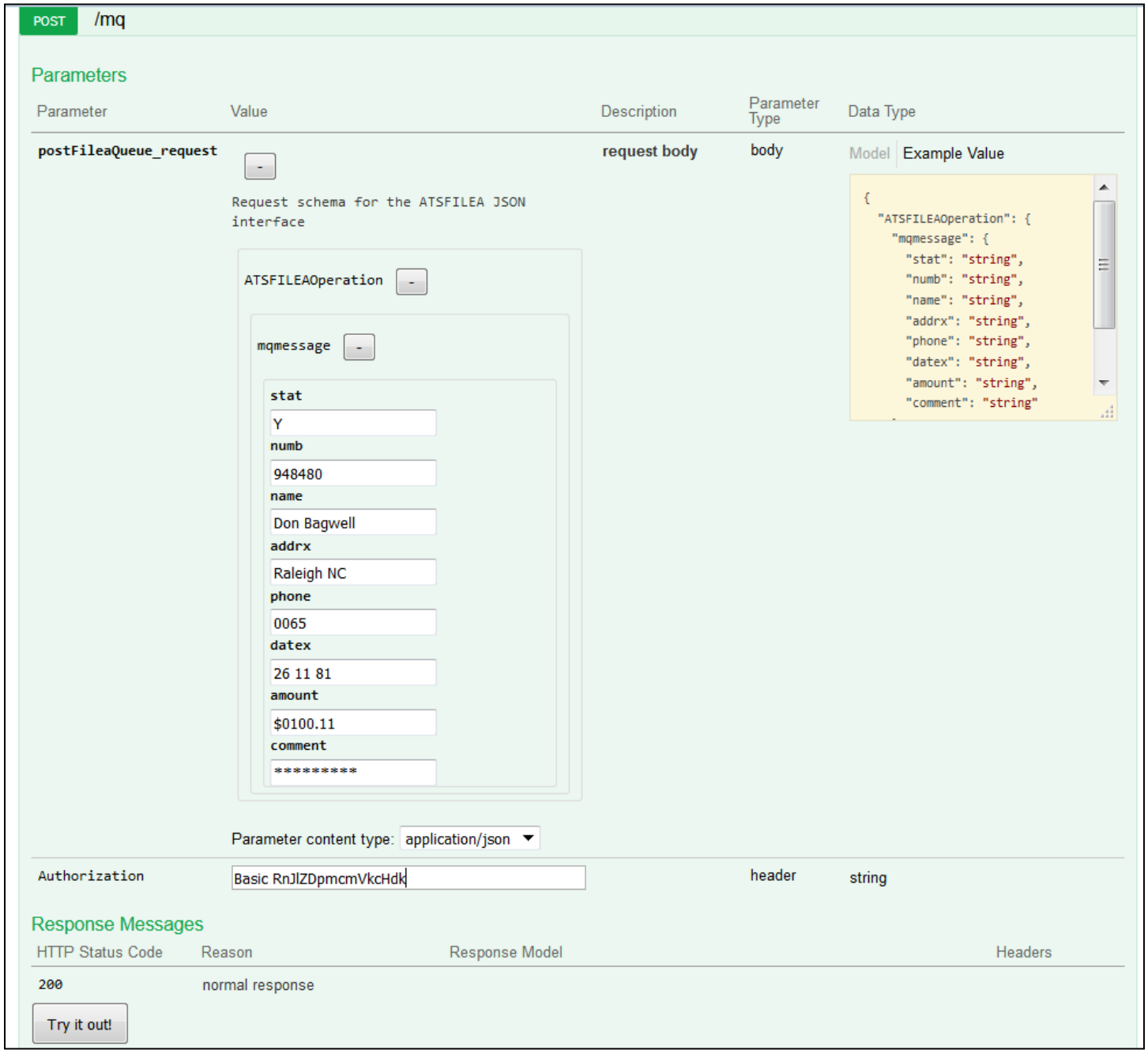
___11. Expand the **DELETE** method by clicking on the path beside it (e.g. */mq)* and scroll down until the method *Parameters* are displayed as shown below:



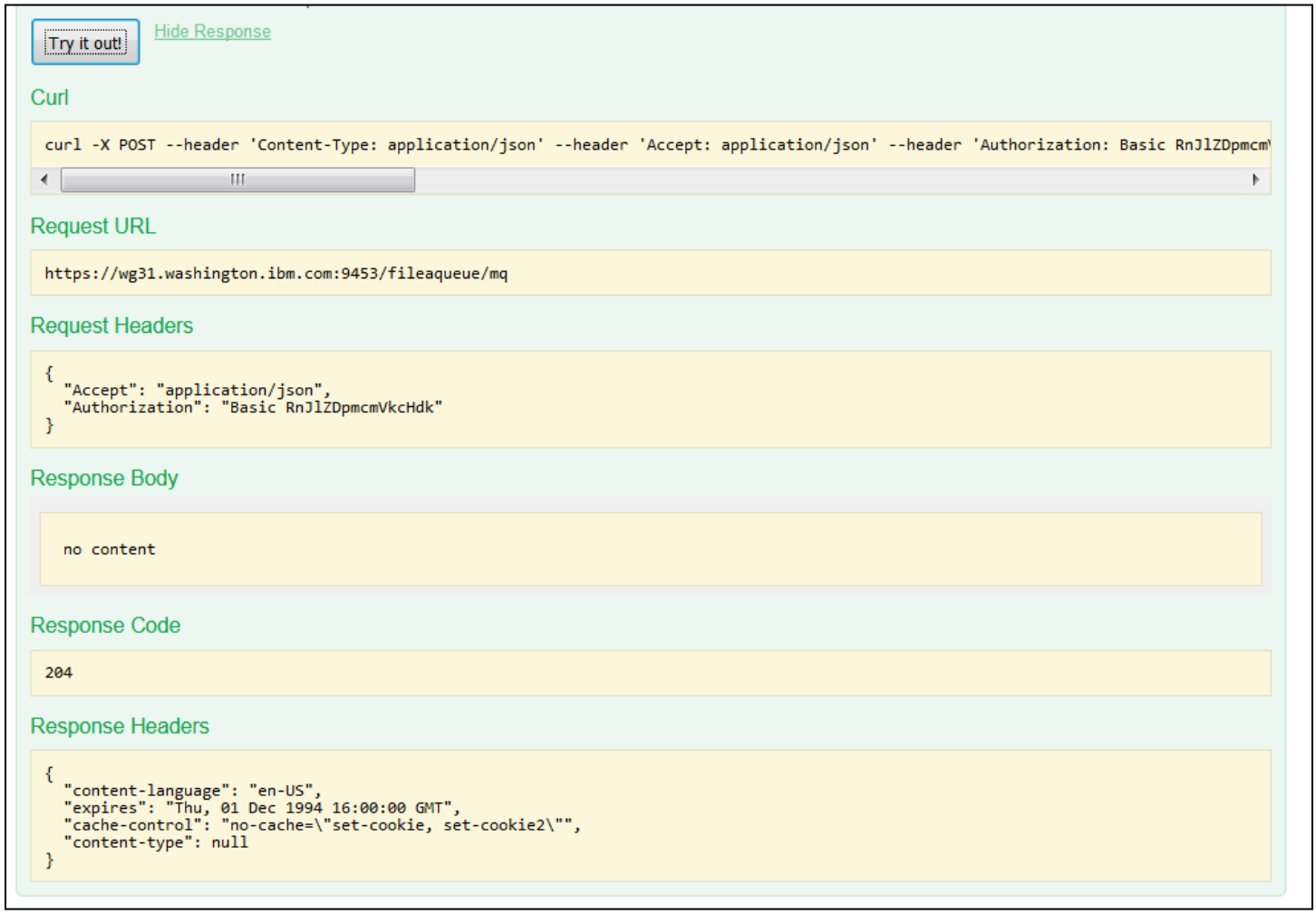
___12. Enter ***Basic RnJlZDpmcmVkcHdk*** in the area beside *Authorization* and click **Try it out!**. Scroll down to display the response in the *Response Body*. This is the message from the application returned in the reply queue. This was the message retrieved by a destructive get. If you continue to click the **Try it out!** button you will see a different message each time.



___13. Expand the **POST** method by clicking on the path beside it (e.g. */mq)* 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.



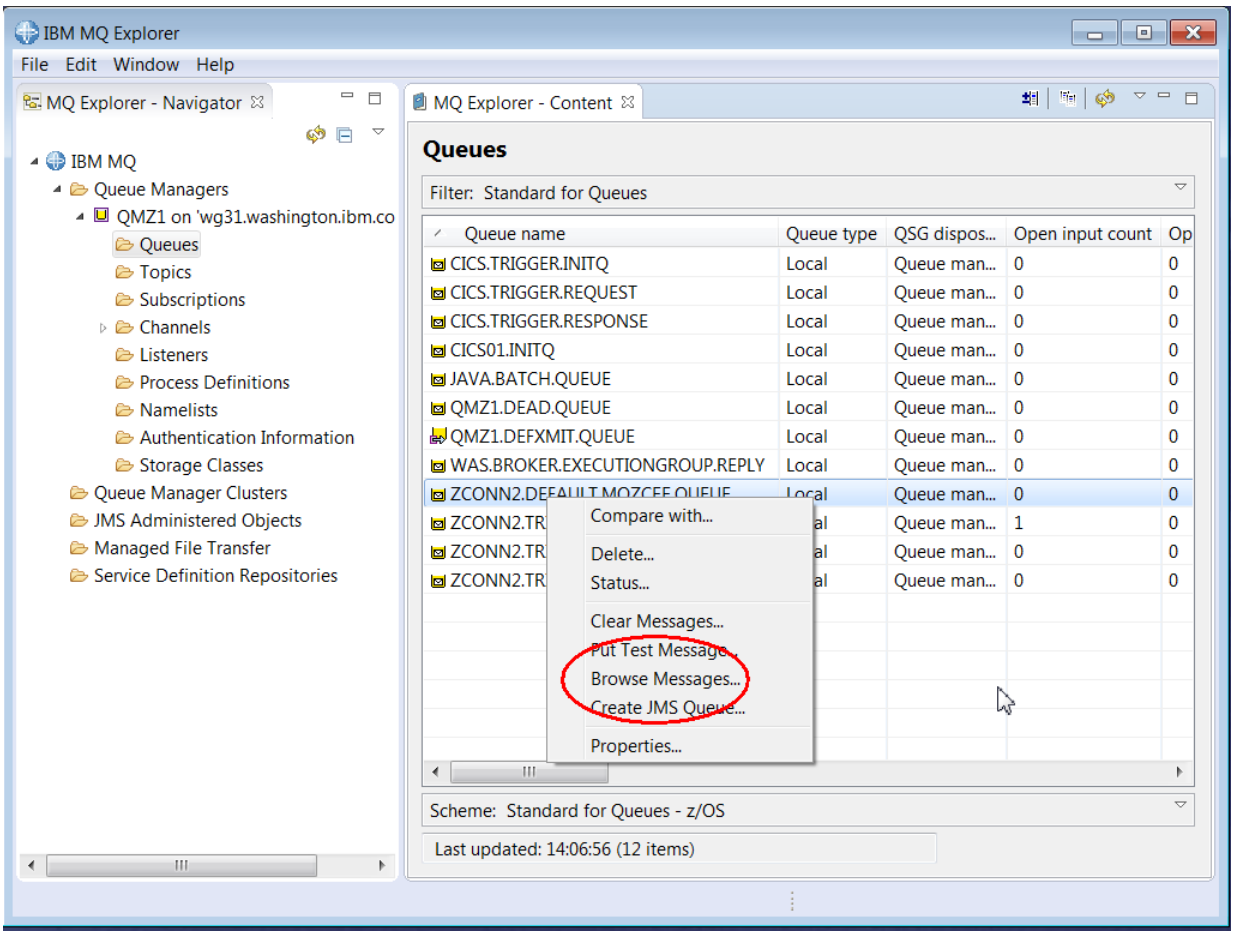
___14. Scroll down to display the response in the *Response Body.* Note that there is no response message. The MQ Service provider does not return a reply message for the *POST* method. If you continue to click the **Try it out!** button you will see the same results over and over.

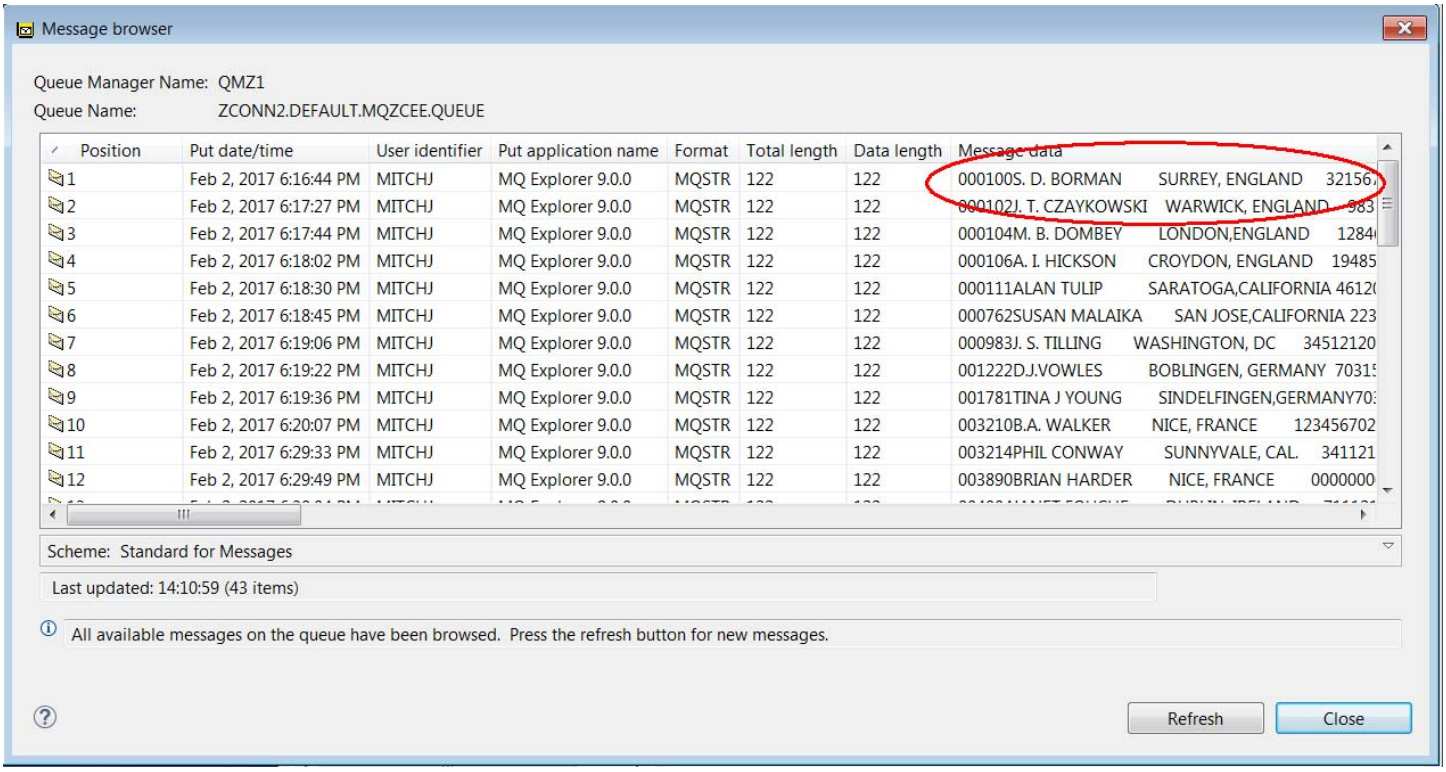


Tech-Tip: An HTTP code of 204 the server processed the request but returned no content. For an explanation of HTTP codes see URL
https://en.wikipedia.org/wiki/List_of_HTTP_status_codes

___15. 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.

___16. Do a few more **GET** and **DELETE** request and observes the changes in the queue. Also do a few **POST** requests but remember there is no **PUT** method available. Updating an existing message is not supported.

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 (FileaQueue) is a one-way service where the same queue is used for POSTs, PUTs and GETs.

Mitch Johnson (mitchj@us.ibm.com)