IBM z/OS Connect (OpenAPI 2.0)

Developing RESTful APIs for a CICS COMMAREA program

Wildfire
Technical Hands-On Workshops

IBM

IBM Z
Wildfire Team - WSC

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 CICS to perform this exercise. Even if CICS is not relevant to your current plans performing this exercise will give additional experience using the Toolkit to develop services and APIs.

The objective of these exercises is to gain experience with working with z/OS Connect and the API Toolkit. These two products allow the exposure of z/OS resources to JSON clients. For information about scheduling this workshop in your area contact your IBM representative.

If you have completed another of the developing APIs exercises for this workshop you can start with section *z/OS Connect APIs and a CICS COMMAREA program* on page 7.

General Exercise Information and Guidelines
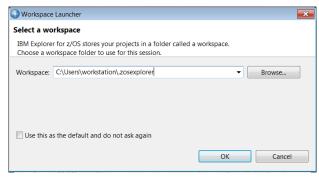
✓ This exercise requires using z/OS user identity *USER1*. The RACF password for this user is USER1.

✓ 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.

✓ For information regarding the use of the Personal Communication 3270 emulator, see the *Personal Communications Tips* PDF in the exercise folder.

Connect the IBM z/OS Explorer to the z/OS Connect 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 mouse button clicking and then selecting the *Open* option.

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

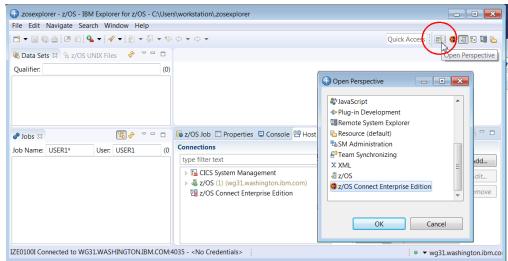___2. You will be prompted for a workspace:

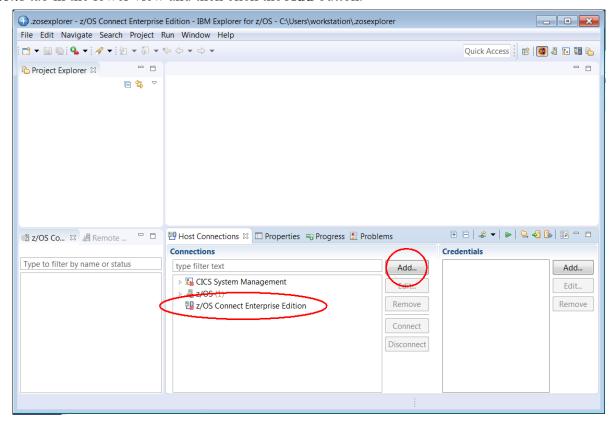Take the default value by clicking **OK**.

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

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

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

___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.
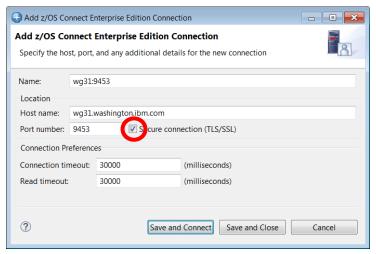

Tech-Tip: Eclipse based development tools like z/OS Explorer; provide a graphical interface consisting of multiple views within a single window.

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).
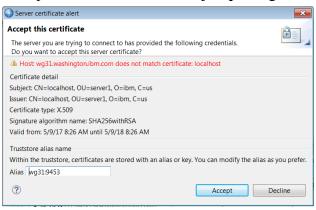
At any time, a specific view can be enlarged to fill the entire window by double clicking in the view's title bar. Double clicking in the view's title bar will be restored the original arrangement. If a z/OS Explorer view is closed or otherwise disappears, the original arrangement can be restored by 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.

Mitch Johnson (mitchj@us.ibm.com)

___6. In the pop-up list displayed select *z/OS Connect Enterprise Edition* and on the *Add z/OS Connect Enterprise Edition Connection* screen enter **wg31.washington.ibm.com** for the *Host name*, **9453** for the *Port Number*, check the box for *Secure connection (TLS/SSL)* and then click the **Save and Connect** button.

___7. On the *z/OS Connect Enterprise Edition – User ID* required screen create new credentials for a *User ID* of **Fred** and a *Password or Passphrase* of **fredpwd** (case matters). Remember the server is configured to use basic security. If SAF security had been enabled, then a valid RACF User ID and password will have to be used instead. Click **OK** to continue.

___8. Click the **Accept** button on the *Server certificate alert – Accept this certificate* screen. You may be presented with another prompt for a userid and password, enter **Fred** and **fredpwd** again.

___9. The status icon beside *wg31:9453* should now be a green circle with a lock. This shows that a secure connection has been established between the z/OS Explorer and the z/OS Connect server. A red box indicates that no connection exists.
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
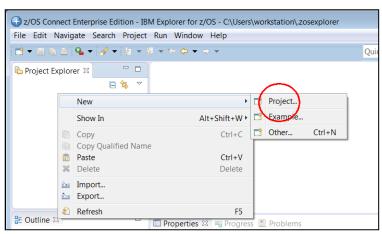
Mitch Johnson (mitchj@us.ibm.com)

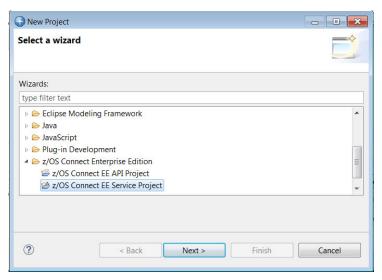# z/OS Connect APIs and a CICS COMMAREA program

Create the services

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

The next step is to create 3 services which correspond to these three functions provided by the sample program DFH0XCMN. These service names must match the service names specified in the server.xml file (see catalog.xml on page 31).
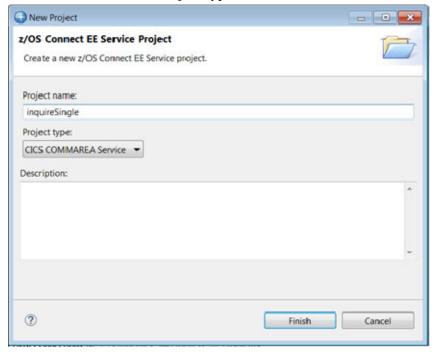
___1. In the z/OS Explorer session switch the perspective back to the *z/OS Connect Enterprise Edition* perspective.

___2. In the upper left, position your mouse anywhere in the *Project Explorer* view and right-mouse click, then select *New* → *Project*:
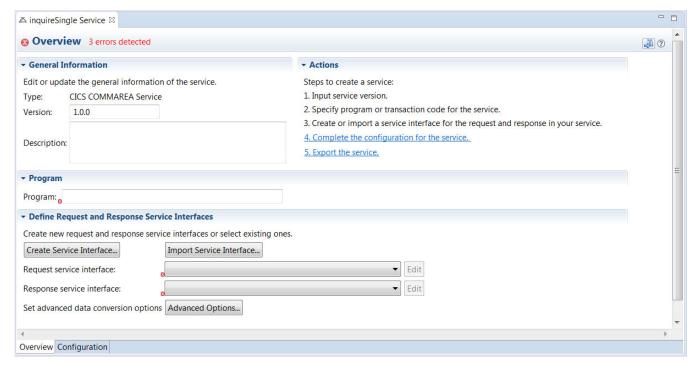
___3. In the *New Project* window, scroll down and open the *z/OS Connect Enterprise Edition* folder and select *z/OS Connect Service Project* and then click the **Next** button.

___4. On the new *New Project* window enter ***inquireSingle*** as the *Project name* and use the pull-down arrow to select *CICS COMMAREA Service* as the *Project type*. Click **Finish** to continue

Mitch Johnson (mitchj@us.ibm.com)
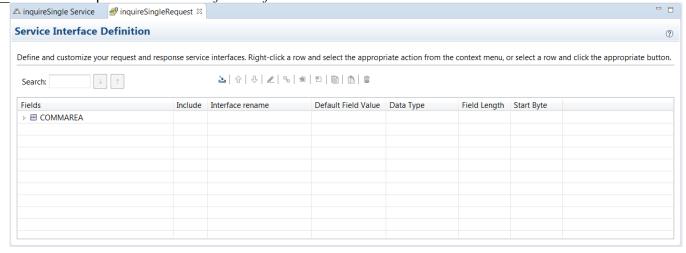
___5. This will open the *Overview* window for the *inquireSingle Service*. For now disregard the message about the 3 errors detected, they will be addressed shortly.

> **Tech-Tip:** If this view is closed it can be reopened by double clicking the *service.properties* file in the service project.
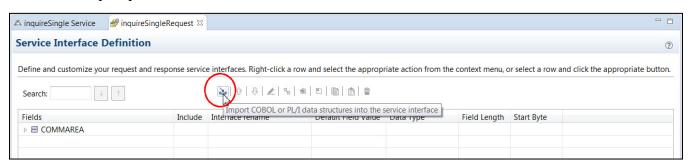
___6. Next enter the target CICS program name ***DFH0XCMN*** in the area beside *Program* (case is important).

___7. Click the **Create Service Interface** button to create the first service required by this API and enter a *Service interface name* of ***inquireSingleRequest***. Click **OK** to continue.

___8. This will open a *Service Interface Definition* window.
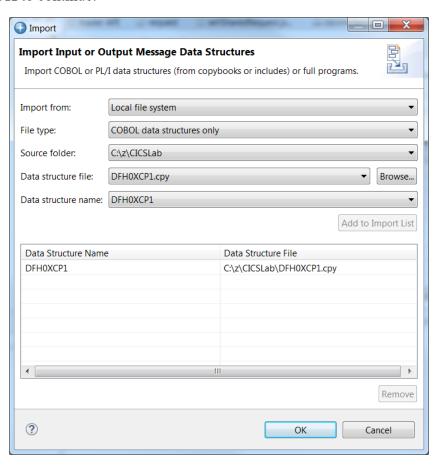
___9. The first step is to import the COBOL copy book that represents the COMMAREA when inquiring on a single item in the Catalog application. This copy book was downloaded from member DFH0XCP1 in the CICS SDFHSAMP target data set.

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

___10. On the *Service Interface Definition* window there is a tool bar near the top. If you hover over an icon its function will be display as below. Click the *Import COBOL or PL/I data structure into the service interface* icon to start the import process.
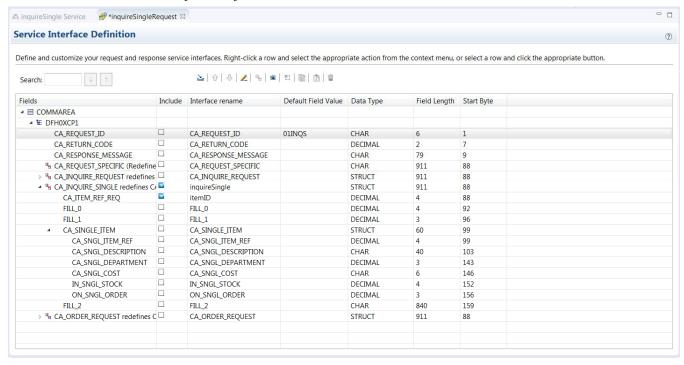
___11. This will open the *Import* window. On this window select *Local file system* as source of the import and *COBOL data structure only* as the *File type*. Press the **Browse** button and **Open** directory *C:\z\CICSLAB* and then select file *DFH0XCP1*.cpy and click **Open** to import this file into this project. Click the **Add to Import List** button and the click **OK** to continue.
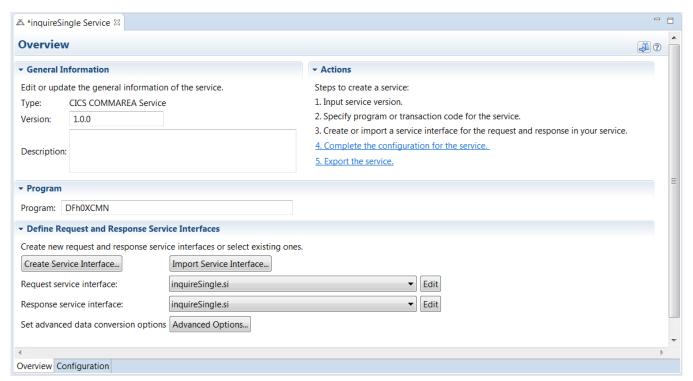
___12. When you expand *COMMAREA* you will see the COBOL 'variables' that have been imported into the service project.   In this service we want to return a single item and the only fields that need to be exposed in the request message is the item number.

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

of this API.

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

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

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

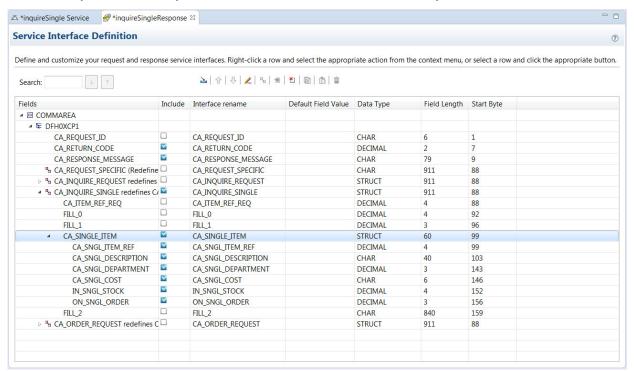N.B. the original interface names were derived from the COBOL source shown earlier.

___13. Close the *Service Interface Definition* window by clicking on the white X in the tab being sure to save the changes. Note now that the *Request service interface* and the *Response service interface* areas have now been populated with *inquireSingleRequest.si*. Also note that you can use their respective **Edit** buttons to return to the *Service Interface Definition* window for each interface.

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

___14. Repeat steps 6 through 10 to create a service interface for the *inquireSingle* service response message. Enter a service interface name of ***inquireSingleResponse*** rather than ***inquireSingleRequest***.

___15. A different set of fields will be returned in the response.   Uncheck the boxes under *Include* column so that only *CA_RETURN_CODE, CA_RESPONSE_MESSAGE, CA_INQUIRE_SINGLE* and *CA_SINGLE_ITEM* checked. These are fields that will be exposed to the requestor of this API in the response. You can rename these fields if you choose. If you do some of the later screen shots may be different.
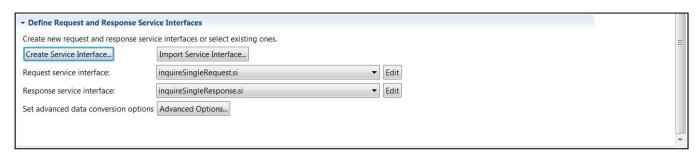
___16. Close the *Service Interface Definition* window.

___17. Set the *Response service interface* to *inquireSingleResponse.si*.

___18. Next, we need to identify a connection reference for which CICS region will be used. Click on the *Configuration* tab at the bottom of the *Overview* window to display the *Configuration* window. Enter **catalog** in the area beside *Connection reference*.
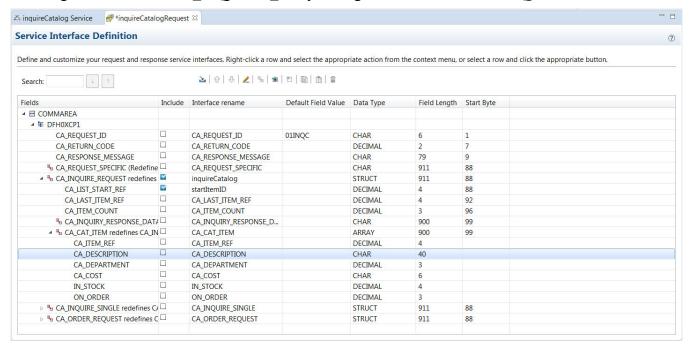
> **Tech-Tip:** The *Connection reference* identifies the *cicsIpicConnection* element to be used to access a CICS region in the z/OS Connect configuration, see catalog.xml on page 31.

___19. Verify that the request and response service interfaces are set correctly. If not, set them as below.

Define Request and Response Service Interfaces

Create new request and response service interfaces or select existing ones.

Create Service Interface...		Import Service Interface...

Request service interface:	inquireSingleRequest.si ▼	Edit
Response service interface:	inquireSingleResponse.si ▼	Edit

Set advanced data conversion options | Advanced Options... |

___20. Save the *inquireSingle* service by closing all open tabs.

___21. Repeat steps 1 through 5 to create the *inquireCatalog* service.

___22. Repeat steps 6 through 10 to create a service interface for ***inquireCatalog*** service request message**.** Enter a name of ***inquireCatalogRequest***.

___23. Expand the COMMAREA and perform the following steps for the *inquireCatalog* request:

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

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

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

inquireCatalog Service | ***inquireCatalogRequest***

Service Interface Definition

Define and customize your request and response service interfaces. Right-click a row and select the appropriate action from the context menu, or select a row and click the appropriate button.

Search: [ ]

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

___24. Close the *Service Interface Definition* window.

___25. Repeat steps 6 through 10 to create a service interface for the *inquireCatalog* response message with the *service interface name* of ***inquireCatalogResponse***.

___26. A different set of fields will be returned in the response.   Uncheck the boxes under *Include* column so that only *CA_RETURN_CODE, CA_RESPONSE_MESSAGE, CA_INQUIRE_REQUEST, CA_LIST_START_REF, CA_LAST_ITEM_REF, CA_ITEM_COUNT* and *CA_CAT_ITEM* are checked. These are fields that will be exposed to the requestor of this API in the response. Rename these fields if you choose.
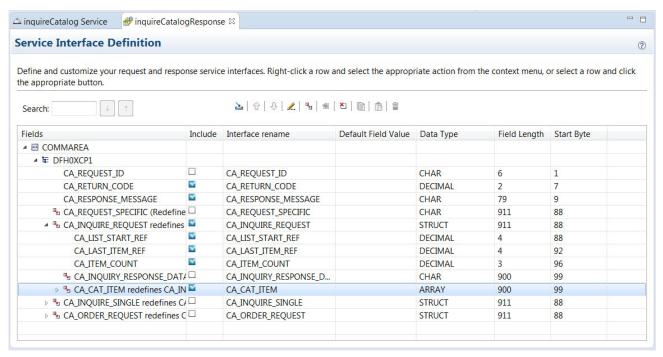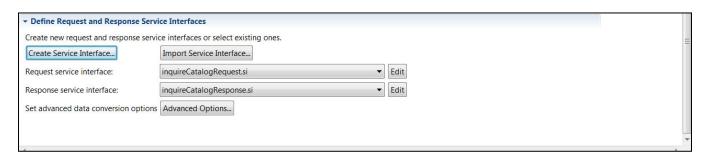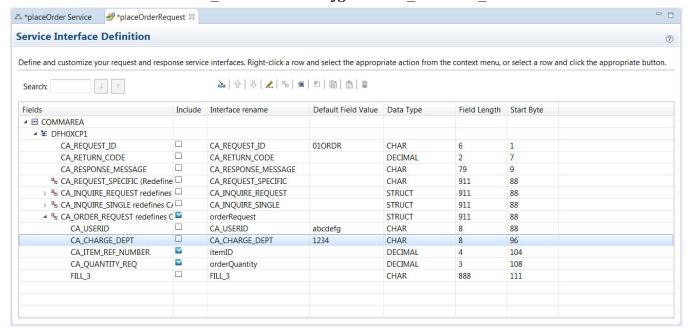

___27. Close the *Service Interface Definition* window.

___28. Set the *Response service interface* to *inquireCatalogResponse.si*.

___29. Identify the connection reference for which CICS region will be used for this service. Click on the Configuration tab at the bottom of the *Overview* window to display the *Configuration* window. Enter **catalog** in the area beside *Connection reference*.

___30. Verify that the request and response service interfaces are set correctly, if not set them as below.
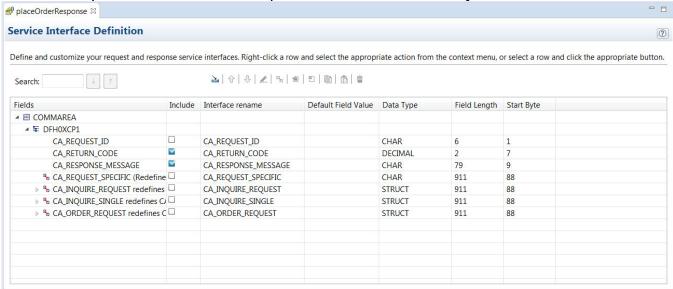
▾ Define Request and Response Service Interfaces
Create new request and response service interfaces or select existing ones.
[Create Service Interface...]   [Import Service Interface...]
Request service interface: [inquireCatalogRequest.si ▼] [Edit]
Response service interface: [inquireCatalogResponse.si ▼] [Edit]
Set advanced data conversion options [Advanced Options...]

___31. Save the *inquireCatalog* service by closing all open tabs.

___32. Create a *placeOrder* service by repeating steps 1 through 5.

___33. Repeat steps 6 through 10 to create *service interface name **placeOrderRequest***

___34. Expand the COMMAREA and perform the following steps for the placeOrderRequest:

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

b. Double click an entry in the *Interface rename* column to open it for editing. Rename interface fields *CA_ORDER_REQUEST* to ***orderRequest***, ***CA_ITEM_REF_NUMBER*** to ***itemID*** and CQ_QUANTITY_REQ to ***orderQuantity***.

c. Change the default value of field *CA_REQUEST_ID* to a value of ***01ORDR***.

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

△ *placeOrder Service   🗗 *placeOrderRequest ⊠

Service Interface Definition

Define and customize your request and response service interfaces. Right-click a row and select the appropriate action from the context menu, or select a row and click the appropriate button.

Search: [_____]

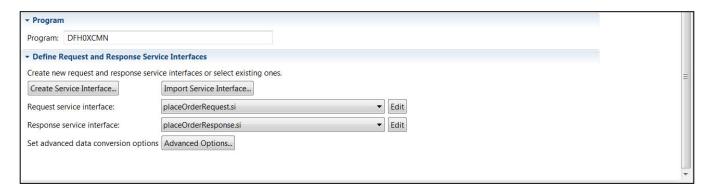| Fields	Include	Interface rename	Default Field Value	Data Type	Field Length	Start Byte
▲ ▦ COMMAREA						
▲ 🝆 DFH0XCP1						
CA_REQUEST_ID	☐	CA_REQUEST_ID	01ORDR	CHAR	6	1
CA_RETURN_CODE	☐	CA_RETURN_CODE		DECIMAL	2	7
CA_RESPONSE_MESSAGE	☐	CA_RESPONSE_MESSAGE		CHAR	79	9
🝆 CA_REQUEST_SPECIFIC (Redefine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

___35. Close the *Service Interface Definition* window.

___36. Repeat steps 6 through 10 to a create service interface ***placeOrderResponse*** response message for the *placeOrder* service.

Mitch Johnson (mitchj@us.ibm.com)

___37. A different set of fields will be returned in the response.   Uncheck the boxes under *Include* column so that only *CA_RETURN_CODE* and *CA_RESPONSE_MESSAGE,* are checked. These are fields that will be exposed to the requestor of this API in the response. Rename these fields if you choose.

___38. Set the *Response service interface* to *placeOrderResponse.si*.

___39. Close the *Service Interface Definition* window.

___40. Identify the connection reference for which CICS region will be used for this service. Click on the Configuration tab at the bottom of the *Overview* window to display the *Configuration* window. Enter **catalog** in the area beside *Connection reference*.

___41. Verify the request and response services interfaces are set correctly, if not set them as below.
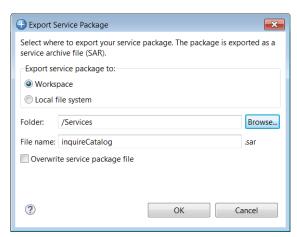
___42. Save the *placeOrder* service either by closing all open tabs.

These three services now need to be made available for developing the API and for deployment to the z/OS Connect server.
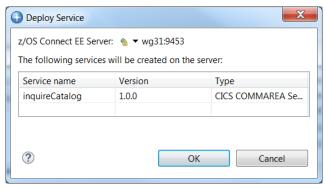
Export and deploy the Service Archive files

Before a service interface can be used it must be exported to create Service Archive (SAR) file and deployed as a SAR file to the server. The exported SAR is used in developing an API in the z/OS Connect Toolkit. This section describes the process for exporting and deploying SAR files.

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

___2. On the *New Project* window enter **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 *inquireCatalog* service project and right mouse button click. On the pop-up selection select **z/OS Connect** ➔ **Export z/OS Connect Service Archive.** On the *Export Services Package* window select the radio button beside *Workspace* and use the **Browse** button to select the *Services* folder. Click **OK** to continue.
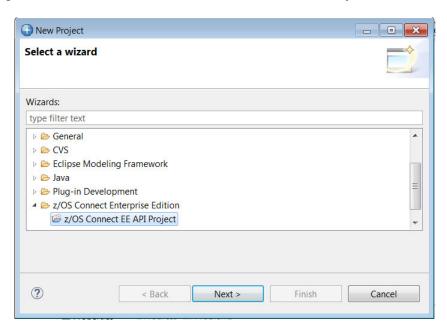
___4. Select the *inquireCatalog* service project again and right mouse button click again and on the pop-up selection select **z/OS Connect → Deploy Service to z/OS Connect Server**. On the *Deploy Service* window select the target server (*wg31:9453*) and click **OK** twice to continue.
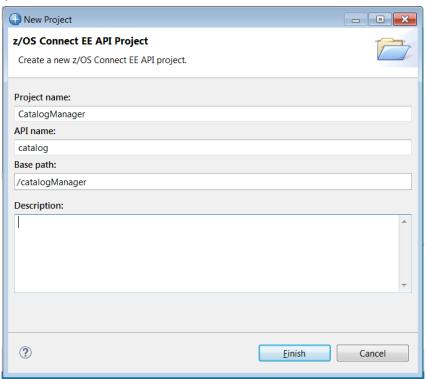
___5. Repeat these two steps to export the *inquireSingle* and *placeOrder* service project to the *Services* project folder and to deploy the services to the z/OS Connect server.
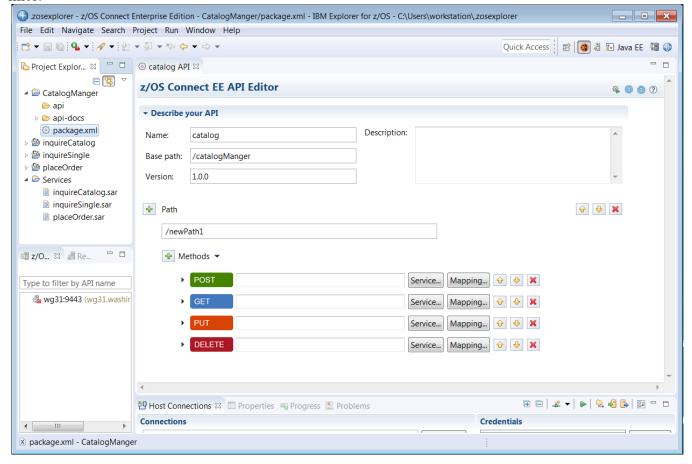
Mitch Johnson (mitchj@us.ibm.com)

Create the API Project

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

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

Mitch Johnson (mitchj@us.ibm.com)

___3. You should now see something like below. The views may need to be adjusted by dragging the view boundary lines.
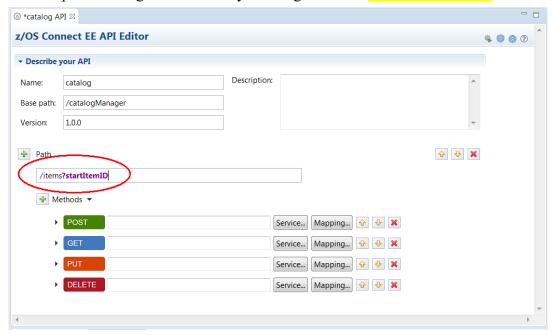
> **Tech-Tip:** If the API Editor view is closed, it can be reopened by double clicking the *package.xml* file in the API project.

Summary

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

Compose the API with a CICS COMMAREA application
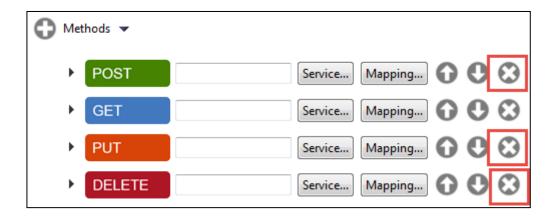
___1. To compose the *inquireCatalo*g service start by entering a *Path* of <mark>*/items?startItemID*</mark>.

> **Note:** The */items* path element is somewhat arbitrary but use this value so tasks later in the exercise will work unchanged.
>
> The *?startItemID* element tells the editor that value will be supplied as a path parameter in the URL.
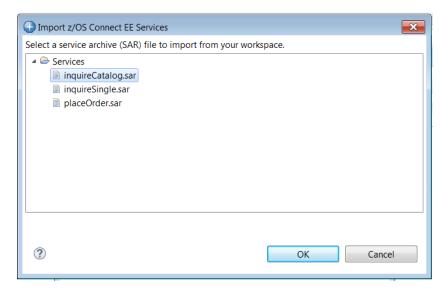
___2. For the *inquireCatalog* service the HTTP method will be **GET**, so we do not need the **POST**, **PUT** and **DELETE** methods. Remove them by clicking the *X* icon to the right of each:
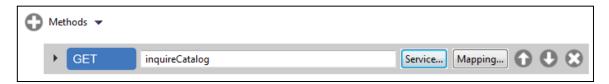
That should leave you with only the **GET** method.

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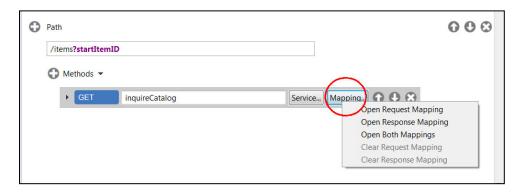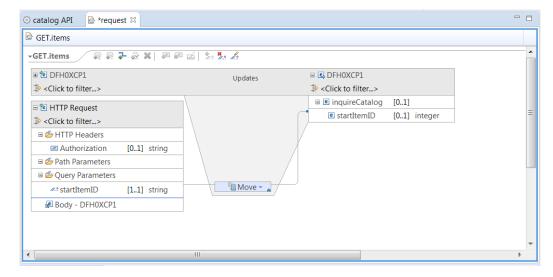
This will populate the field to the right of the method:

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

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

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

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

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

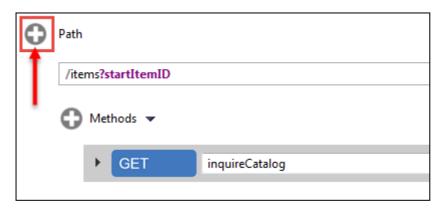http://<host>:<port>/catalogManager/items?startItemID=10

The query parameter provides the program the *starting* item number for the listing of items.
Those fields not needed on a request are hidden from the REST client's view. Those fields that have the same static value every time (*ca_request_id*) are assigned a value of 01INQC.
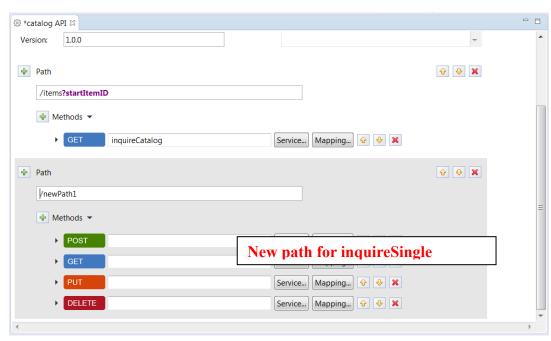
___7. Save the changes using key sequence **Ctrl-S**.

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

___9. No changes are required for the response mapping.

You're done with the *inquireCatalog* portion of the API. Next up is *inquireSingle*. The steps are very similar. You will find you get better at this the more you do it.
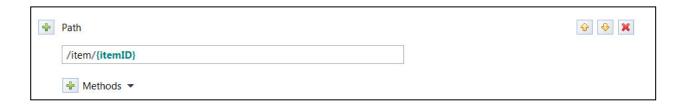
___10. Click the plus icon to add another path to the API.
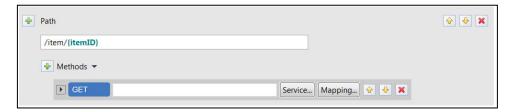
The result is:

New path for inquireSingle

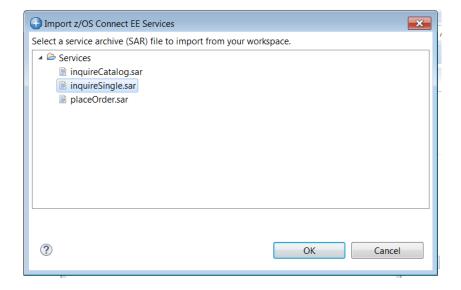___11. Enter a path value of of */item/{itemID}*:

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

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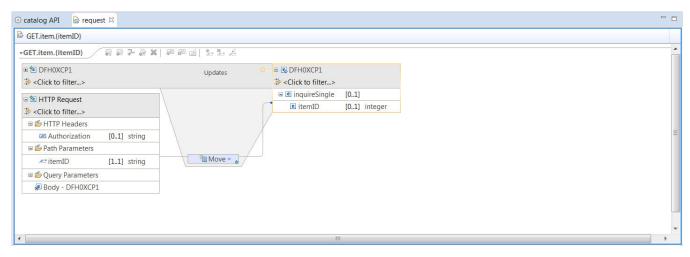

. **Tech-Tip:** Additional *Paths* can be added by clicking the + icon beside *Path and a*dditional *Methods* can be added by clicking the + icon beside *Methods*.

___13. Next import the service required for this method of this *Path*. Click on the **Service** button to the right of the **GET** me*thod. On the Select a z/OS Connect Service window click the* **Workspace** *button and expand the Services* folder on the next window. Select the *inquireSingle* service from the list of service archive files and click **OK** three times.

___14. Save the changes by using the key sequence **Ctrl-S**.

___15. Click on *Mapping → Open request mapping.*

___16. Expand *inquireSingle* by clicking on the little + sign to the left of the field. Select *itemID* on the left-hand side and drag it over to *itemID* on the right hand side to make a move connection so the value or contents of *itemID* are moved into *itemID* field.

___17. The picture below is the final result. Inspect your mapping and make sure:

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

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

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

___18. Save the the changes using the key sequence **Ctrl-S**.

___19. Close the *request* mapping tab.

___20. No changes are required to the *response* mapping.

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

___21. Add another path by clicking on the + circle:

___22. In your new path field, provide a path value of ***/orders***:

___23. The HTTP method for this will be **POST**, so we can get ride of **GET**, **PUT** and **DELETE.** Remove those methods by clicking the **X**. You should then see:

___24. Click on *Mapping* → *Open request mapping* and review. No mappings are required so close the view.

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

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

Summary

You created the API, which consists of three paths and the request and response mapping associated with each. That API will now be deployed into a z/OS Connect server. The REST interfaces are now simpler than the earlier coarse-grained services. The services are still used, but by z/OS Connect server as it processes the higher-level API requests.
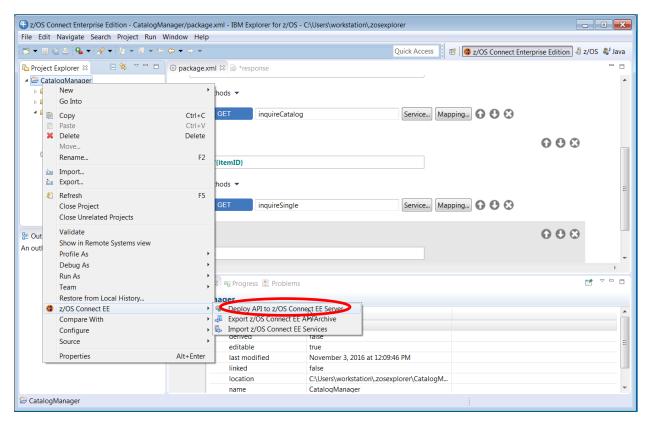
Mitch Johnson (mitchj@us.ibm.com)

Deploy the API to a z/OS Connect Server

___1. The *catalog* API *and* **inquireCatalog, inquireSingle and placeOrder** services were defined by the inclusion of the *catalog.xml* file in the *server.xml*.

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

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

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

</server>                                                "
```
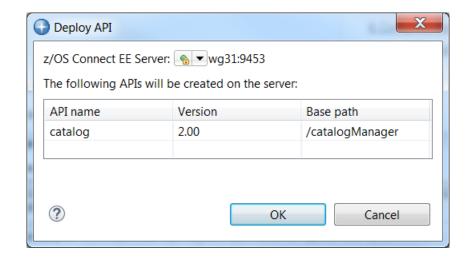
Figure 1 - catalog.xml

The *zosconnect_cicsIpicConnection* element provides the CICS IPIC information that will be used for communications with the CICS region,
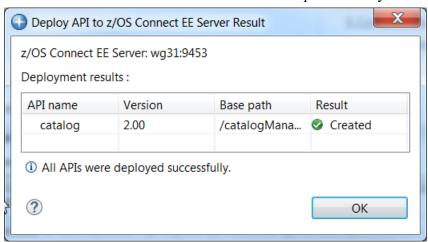
___2. In the *Project Explorer* view (upper left), right-mouse click on the *CatalogManager* folder, then select *z/OS Connect*→ *Deploy API to z/OS Connect Server*.

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/zosconnect/servers/zceesrv1/resources/zosconnect/apis directory.
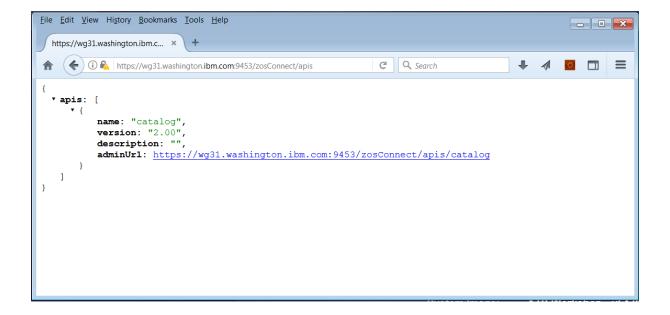

Summary

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

Test the API

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

___1. Next enter URL ***https://wg31.washington.ibm.com:9453/zosConnect/apis*** in the Firefox browser and you should see the window below. The *catalog* API now shows as being available.\

> **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.

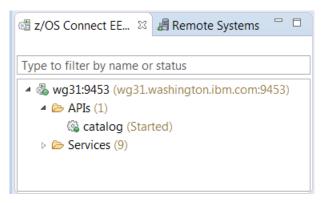___2. If you click on *adminUrl* URL the window below should be displayed:

```
{
    name: "catalog",
    version: "2.00",
    description: "",
    status: "Started",
    apiUrl: https://wg31.washington.ibm.com:9453/catalogManager,
  ▼ documentation: {
        swagger: https://wg31.washington.ibm.com:9453/catalogManager/api-docs
    },
  ▼ services: [
      ▼ {
            name: "placeOrder",
            uri: https://wg31.washington.ibm.com:9453/zosConnect/services/placeOrder
        },
      ▼ {
            name: "inquireSingle",
            uri: https://wg31.washington.ibm.com:9453/zosConnect/services/inquireSingle
        },
      ▼ {
            name: "inquireCatalog",
            uri: https://wg31.washington.ibm.com:9453/zosConnect/services/inquireCatalog
        }
    ]
}
```

___3. Next click on the *swagger* URL and you should see the Swagger document associated with this API.
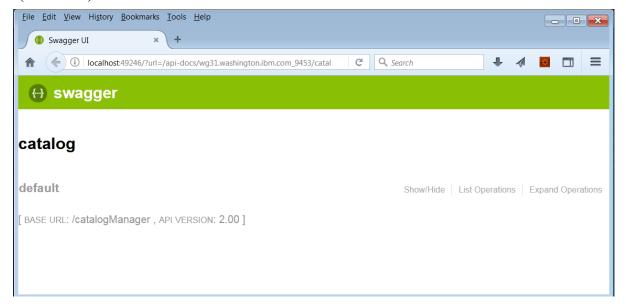
```
{
    swagger: "2.0",
  ▼ info: {
        description: "",
        version: "1.0.0",
        title: "catalog"
    },
    basePath: "/catalogManager",
  ▼ schemes: [
        "https",
        "http"
    ],
  ▼ consumes: [
        "application/json"
    ],
  ▼ produces: [
        "application/json"
    ],
  ▼ paths: {
      ▼ "/item/{itemID}": {
          ▼ get: {
                operationId: "getInquireSingle"
```

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

___4. In the lower left-hand side of the *z/OS Connect Explorer* perspective there is a view entitled *z/OS Connect Servers.* Expand *wg31:9453* and then expand the *APIs* folder. You should see a list of APIs installed in the server.
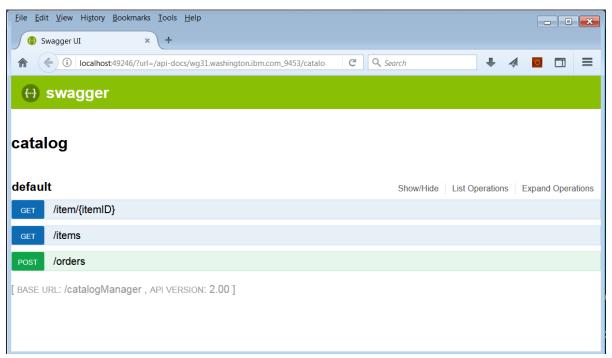
___5. Right mouse button click on *catalog* and select *Open in Swagger UI.* Click **OK** if an informational prompt about certificates appears (see Tech Tip for explanation). This will open a new view showing a *Swagger* test client (see below).

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

___6. Click on *List Operations* option in this view and this will display a list of available HTTP methods in this API.

___7. Select the method for selecting a single item from the catalog by item number table by clicking on the *GET* box. This action will expand this method in this view and provides a Swagger UI test client (you may have to use the slider bar and adjust the perspective to see the entire client).

Response Content Type application/json ▼				
Parameters				
Parameter	Value	Description	Parameter Type	Data Type
itemID	(required)		path	string
Authorization			header	string
Try it out!				

___8. Enter *10* in the box beside *itemID* and **==Basic RnJlZDpmcmVkcHdk==** in the area beside *Authorization* and press the **Try it out!** button. You may see a Security Alert pop-up warning about the self-signed certificate being used by the z/OS Connect server. Click **Yes** on this pop-up.

> **Tech Tip:** Since the z/OS Connect server require security the Swagger UI must include the authorization string in the Request Header. The format of this authorization string is the type of authorization (e.g. *Basic*) and the base 64 encoded string for *Fred:fredpwd ,e.g RnJlZDpmcmVkcHdk,* see URL https://www.base64encode.org. and/or the *z/OS Connect Getting Started Guide*, TechDoc WP102724. Be sure there are no extra spaces at the end of the string.

___9. Scroll down the view and you should see the *Response Body* which contains the results of the GET method (see below). Note that the request field removed from the interface in an earlier steps is not present.

Response Body

```
{
  "DFH0XCP4": {
    "CA_RESPONSE_MESSAGE": "RETURNED ITEM: REF =0010",
    "CA_INQUIRE_SINGLE": {
      "CA_SINGLE_ITEM": {
        "CA_SNGL_ITEM_REF": 10,
        "CA_SNGL_DESCRIPTION": "Ball Pens Black 24pk",
        "CA_SNGL_DEPARTMENT": 10,
        "IN_SNGL_STOCK": 135,
        "CA_SNGL_COST": "002.90",
        "ON_SNGL_ORDER": 0
      }
    },
    "CA_RETURN_CODE": 0
  }
}
```

Response Code

```
200
```

___10. Repeat this process with other items and observe the results. For example using a value of *30* would return.

```
Response Body

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

The available catalog items are listed below.

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

___11. Collapse the Swagger UI test area for the **GET** single item method for the clicking on the blue *GET* box beside */item/{itemID}*.

___12. Click on the blue *GET* box beside */items* to access the **GET** method to open its Swagger Test user interface for the inquire catalog service (e.g. *inquireCatalog*) and scroll down to the *Response Content Type* area.
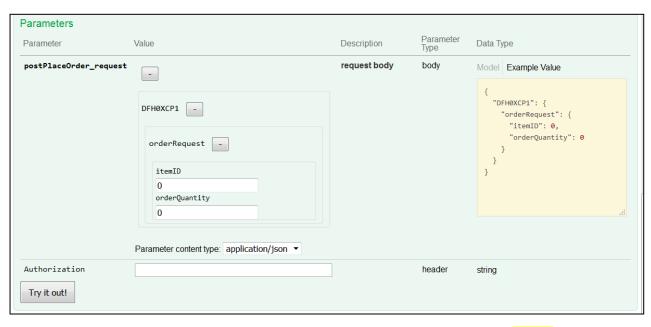
Response Content Type application/json ▾

Parameters

Parameter	Value	Description	Parameter Type	Data Type
startItemID	(required)		query	string
Authorization			header	string

Try it out!

___13. Enter **40** in the box beside *startItemID* and ***Basic RnJlZDpmcmVkcHdk*** in the area beside *Authorization* and press the **Try it out!** button.   Scroll down and you should see the following information in the *Response Body*.

Response Body

```
{
  "DFH0XCP3": {
    "CA_RESPONSE_MESSAGE": "+15 ITEMS RETURNED",
    "CA_INQUIRE_REQUEST": {
      "CA_LAST_ITEM_REF": 180,
      "CA_CAT_ITEM": [
        {
          "ON_ORDER": 0,
          "CA_ITEM_REF": 40,
          "CA_COST": "002.90",
          "IN_STOCK": 80,
          "CA_DESCRIPTION": "Ball Pens Green 24pk",
          "CA_DEPARTMENT": 10
        },
        {
          "ON_ORDER": 0,
          "CA_ITEM_REF": 50,
          "CA_COST": "001.78",
          "IN_STOCK": 83,
          "CA_DESCRIPTION": "Pencil with eraser 12pk"
```

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

___14. Click on the green POST box to access the **POST** method to open its Swagger Test user interface for the place order service (e.g. *placeOrder*) and scroll down to the *Response Content Type* area.

___15. Enter **40** in the area below *itemID* and **1** in the area under *orderQuantity* and **Basic RnJlZDpmcmVkcHdk** in the area beside *Authorization* and press the **Try it out!** button

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

Summary

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

Mitch Johnson (mitchj@us.ibm.com)

Optional

Start a 3270 session with CICS and enter CICS transaction **CEDX CSMI**. When you repeat of any of the above test you should be able to trace the flow of the request through CICS.

```
Session A

File  Edit  View  Communication  Actions  Window  Help

TRANSACTION: CSMI PROGRAM: DFHMIRS  TASK: 0000174 APPLID: CICS53Z  DISPLAY:  00
 STATUS:   ABOUT TO EXECUTE COMMAND
 EXEC CICS LINK PROGRAM
  PROGRAM ('DFH0XCMN')
  COMMAREA ('01INQS.....................................'...)
  LENGTH (998)
  SYNCONRETURN
  NOHANDLE


  OFFSET:X'001CD6'        LINE:                    EIBFN=X'0E02'


 ENTER:   CONTINUE
 PF1 : UNDEFINED          PF2 : SWITCH HEX/CHAR    PF3 : UNDEFINED
 PF4 : SUPPRESS DISPLAYS  PF5 : WORKING STORAGE    PF6 : USER DISPLAY
 PF7 : SCROLL BACK        PF8 : SCROLL FORWARD     PF9 : STOP CONDITIONS
 PF10: PREVIOUS DISPLAY   PF11: EIB DISPLAY        PF12: ABEND USER TASK
MA    A                                            ^                  09/031
Connected to remote server/host wg31q using lu/pool TCP00102 and port 23
```

Summary

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