



L97– zOSConnect EE – CICS Provider API

Lab Version V30.05.zVA

July 17, 2023

Please send any comments on this lab exercise to:
Steve Fowlkes, Leigh Compton, Kenishia Callaway
fowlkes@us.ibm.com
lcompton@us.ibm.com
kenishia@us.ibm.com

Overview

This lab exercise has CICS acknowledging a RESTful JSON web service using z/OS Connect Enterprise Edition (zCEE). The JSON request is delivered to an existing CICS application and uses a bottom-up approach.

Cloud and mobile applications reshape the way enterprises and systems interact. RESTful APIs that use JSON message formats are the predominant standards for new application development. z/OS Connect EE provides a framework that enables z/OS-based programs and data to participate fully in the new API economy for mobile and cloud applications.

z/OS Connect EE can help deliver benefits for an enterprise in the following ways:

- Mobile and cloud application developers, inside or outside the enterprise, can incorporate z/OS data and transactions into their applications without the need to understand z/OS subsystems. The z/OS resources appear as any other RESTful API.
- z/OS application programmers can take advantage of published RESTful APIs to incorporate service and data into their business applications.

The CICS Explorer provides a facility called the z/OS Connect EE API toolkit which can be used to configure JSON services for z/OS Connect. You can expose an existing CICS COMMAREA or channel-based program (the bottom-up technique) written in COBOL, PL/I, C, and C++.

The z/OS Connect EE API toolkit helps you create a service, define the request and response service interfaces, and specify how the service can access and interact with CICS.

A *service interface* defines how the underlying program, COBOL copybook, or PL/I include can be accessed, which fields can be exposed, and how they are exposed. A service interface can be reused by multiple services. Service interfaces are stored as service interface files (.si files).

After a service is created, export the service as a service archive (.sar) file. The service archive file is required for REST API creation and service deployment.

This lab exercise utilizes a bottom-up technique. It starts with a copybook and generates a JSON schema (along with other artifacts). We wanted to expose the existing COBOL application using a RESTful approach (i.e. GET, PUT, POST, DELETE).

Scenario

In our scenario, you have an existing COBOL application that you will expose as a RESTful API. You want to expose this application with a RESTful interface where the data is sent in JSON format, and a HTTP GET method does a read, and POST does a create, a PUT does an update, and a DELETE does a delete.

You will use the z/OS Connect EE API toolkit to create a Service Archive (SAR) file, Request/Response schemas, and other artifacts. You will then build an API project that will interface with the existing COBOL application.

After deploying the new resources, you will test your new RESTful JSON service.

Lab Requirements

Please note that there are often several ways to perform functions. This lab exercise will present one of the ways. If you are familiar with z/OS Connect, you will notice that some of the statements are general, and not necessarily true for every situation.

This lab uses the CICS Explorer, ISPF under TSO, and CICS. If you are not familiar with these, please contact one of the lab instructors for assistance.

The following are other assumptions made in this lab exercise.

- **CICS TS V6.1:** This is where the backend application is hosted.
- **Login:** TSO userids are available with appropriate passwords.
- **CICS Explorer V5.5:** This lab exercise uses the CICS Explorer V5.5, with the z/OS Connect EE plug-in.

Lab Step Overview

Part 1: Configuring the CICS Explorer connection to CICS

In this part of the lab, you will configure the CICS Explorer.

Part 2: Understand the Existing COBOL copybook and resulting Program Interface

In this section we discuss the required parts to use the z/OS Connect EE API toolkit. We show you a COBOL copybook, a generated JSON schema, and SAR file.

Part 3: Create a service project in the z/OS Connect Enterprise Edition perspective

This part of the lab exercise we are starting with a COBOL copybook and creating the JSON schema files and the service XML file in the service project.

Part 4: Build the API Application

In this part of the lab exercise we will use the CICS Explorer (z/OS Connect Enterprise Edition perspective) to build a z/OS Connect EE API Project.

Part 5: Deploy API to z/OS Connect EE Server (running in CICS)

We will use the deploy function of the Explorer to deploy the API to z/OS Connect EE and deploy the SAR file to z/OS Connect EE.

Part 6: Test your new RESTful JSON Service

We will use the RESTClient plug-in that is installed into Firefox to test your RESTful JSON service.

Part 7: Summary

This is a recap of the steps performed in this lab exercise.

Part 1: Configure the CICS Explorer Connection to CICS

In this part of the lab exercise you will configure the connection between the CICS Explorer running on your workstation to CICS running on z/OS.

Start the CICS Explorer

- ___1. From the **desktop**, **double-click** the **CICS Explorer** icon to start it, if it is not already running.



- ___2. When you start the CICS Explorer, **If** you are prompted for a workspace, click the **OK** button to use the default.
- ___3. If the CICS Explorer shows you a **Welcome page** click the **Workbench icon** in the upper-right corner to go to the workbench. Then **maximize** the Explorer window.



Verify that you have connections to z/OS and CICS in your CICS Explorer

- ___4. **If** you have connections defined from your CICS Explorer to z/OS, **start** them and then skip to **Part 2**.
- ___5. If you have not already created connections to the z/OS host system, follow the instructions in the **Connection Document** and then return here.

Part 2: Understand the Copybook and COBOL program interface

You can expose CICS-based programs to use JSON communications. We will use the tooling in the CICS Explorer to build the required artifacts. The CICS Explorer provides a facility called the z/OS Connect EE API toolkit which can be used to configure JSON services for CICS. You can expose an existing CICS COMMAREA or channel-based program (the bottom-up technique) written in COBOL, PL/I, C, and C++.

We want our CICS-based program to respond to REST requests (GET, PUT, POST, and DELETE). Since we have an existing CICS application that uses a COMMAREA, we will build a service project (SAR) and a REST API application to interface with the main program of the application. The API will be deployed to z/OS Connect EE. z/OS Connect EE will compose the COMMAREA and LINK to the existing program.

Below is the COBOL copybook:

```
*      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 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) .
```

Interestingly, the target URL of the JSON service is not in the JSON Schema. Note also that WSDL and a SOAP request (for XML-based web services) can contain additional information about security and transactionality, which are not available when using 'JSON Services'. The way to secure JSON Services is using HTTP Basic Authentication, TLS, JWT's, etc. The other WS- extensions like WS-Security, WS-Coordination, etc, and, SAML and Kerberos, are not available.

Below is the JSON schema definition generated by the z/OS Connect EE API toolkit (there is also a response schema)

```
{
  "type" : "object",
  "properties" : {
    "DFH0XCP1" : {
      "type" : "object",
      "properties" : {
        "CA_REQUEST_ID" : {
          "maxLength" : 6,
          "type" : "string"
        },
        "CA_RETURN_CODE" : {
          "minimum" : 0,
          "maximum" : 99,
          "type" : "integer"
        },
        "CA_RESPONSE_MESSAGE" : {
          "maxLength" : 79,
          "type" : "string"
        },
        "CA_INQUIRE_SINGLE" : {
          "type" : "object",
          "properties" : {
            "CA_ITEM_REF_REQ" : {
              "minimum" : 0,
              "maximum" : 9999,
              "type" : "integer"
            }
          }
        }
      }
    }
  }
}
```

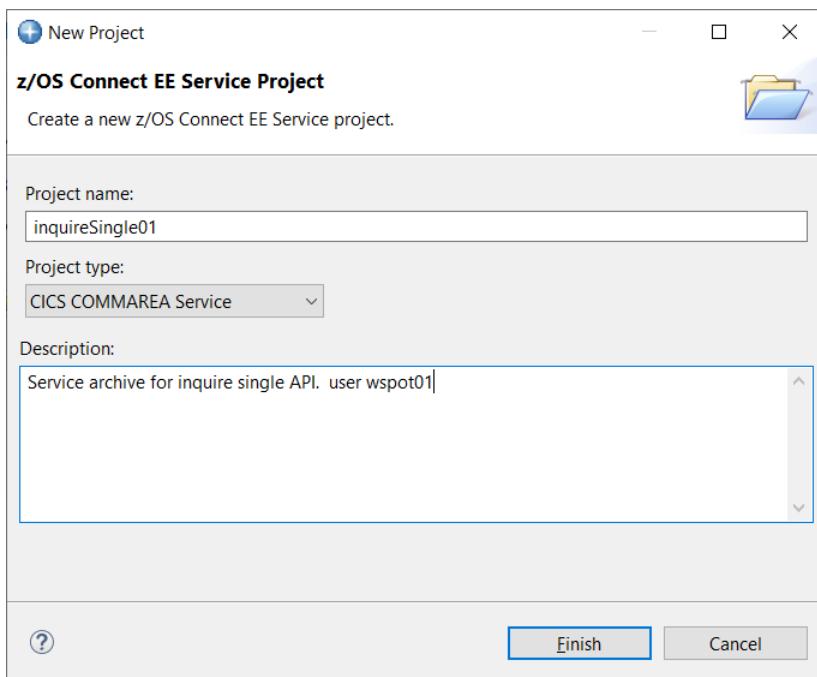
Part 3: Create a service project in the z/OS Connect Enterprise Edition perspective

We will start with a COBOL copybook and generate a Service Archive (SAR) file. The SAR file will be used by the API tooling in the CICS Explorer.

Create a service project using the API toolkit

Create a new z/OS Connect EE service project in the z/OS Connect Enterprise Edition perspective and define the request and response service interfaces.

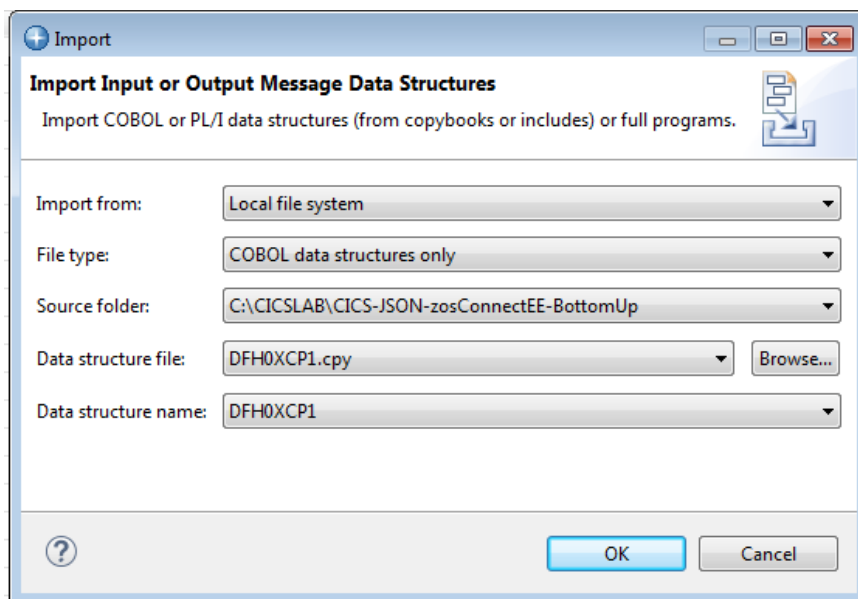
- ___6. From the **CICS Explorer**, the **z/OS Connect Enterprise Edition** perspective, the **Project Explorer** view, select **File > New > z/OS Connect EE Service Project**.
- ___7. From the **New Project** dialog, enter a **Project name** of **inquireSingle**.
- ___8. Enter a **Description** (see example below). Then click **Finish**.



- ___9. The inquireSingle Service opens in the **service.properties** editor.
- ___10. From the inquireSingle Service Project Editor, the Program tab, enter a Program of **DFH0XCMN**.

Create the request service interface

- ___11. From the Define Request and Response Service Interfaces section, click the **Create Service Interface** button and enter a Service interface name of **inquireSingle_request**. Then click **OK**.
- ___12. From the Service Interface Editor dialog, **right-click COMMAREA**. Then click **Import data structure**.
- ___13. From the Import dialog, click the **Browse** button and select **C:\CICSLAB\CICS-JSON-zosConnectEE-BottomUp\DFH0XCP1.cpy**. Click **Open**. This will parse the supplied copybook.



- ___14. Click **OK**.
- ___15. Expand **COMMAREA > DFH0XCP1**. This will allow you to edit the Request service interface (input). Notice that there are three redefines visible (there are four in the structure). Fully expand the **second** one (**CA_INQUIRE_SINGLE**). This is used for an inquire single item, which we are building.

Using the Service Interface Definition dialog, you can define and customize request and response interfaces. You can copy a data structure, specify the order of the data structures, or add multiple containers or segments. You can also specify the fields to include or exclude, rename fields or data structures, change field value type, specify default field values, or add business descriptions.

We will do some editing in the Service Interface and additional editing when creating the API. The ability to do additional editing in the API is helpful, since you can use a Service project in multiple APIs.

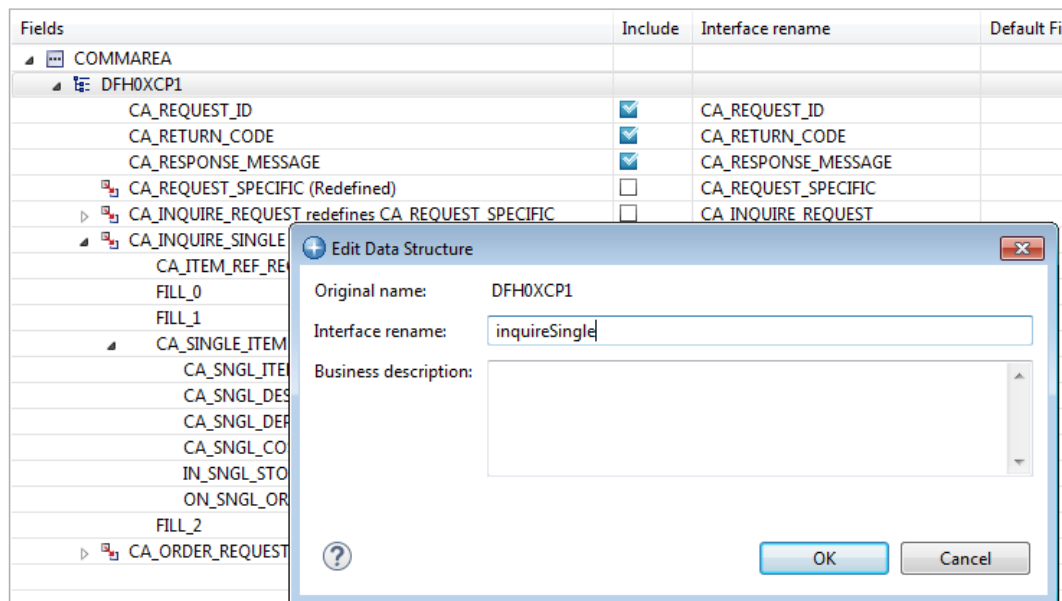
___16. Uncheck the CA_REQUEST_SPECIFIC (Redefined) and check the CA_INQUIRE_SINGLE redefines CA_REQUEST_SPECIFIC.

___17. In the CA_INQUIRE_SINGLE structure, uncheck FILL_0, FILL_1, FILL_2 and CA_SINGLE_ITEM.

Fields	Include	Interface rename
COMMAREA		
DFH0XCP1		
CA_REQUEST_ID	<input checked="" type="checkbox"/>	CA_REQUEST_ID
CA_RETURN_CODE	<input checked="" type="checkbox"/>	CA_RETURN_CODE
CA_RESPONSE_MESSAGE	<input checked="" type="checkbox"/>	CA_RESPONSE_MESSAGE
CA_REQUEST_SPECIFIC (Redefined)	<input type="checkbox"/>	CA_REQUEST_SPECIFIC
CA_INQUIRE_REQUEST redefines CA_REQUEST_SPECIFIC	<input type="checkbox"/>	CA_INQUIRE_REQUEST
CA_INQUIRE_SINGLE redefines CA_REQUEST_SPECIFIC	<input checked="" type="checkbox"/>	CA_INQUIRE_SINGLE
CA_ITEM_REF_REQ	<input checked="" type="checkbox"/>	CA_ITEM_REF_REQ
FILL_0	<input type="checkbox"/>	FILL_0
FILL_1	<input type="checkbox"/>	FILL_1
CA_SINGLE_ITEM	<input type="checkbox"/>	CA_SINGLE_ITEM
CA_SNGI_ITEM_REF	<input type="checkbox"/>	CA_SNGI_ITEM_REF
CA_SNGI_DESCRIPTION	<input type="checkbox"/>	CA_SNGI_DESCRIPTION
CA_SNGI_DEPARTMENT	<input type="checkbox"/>	CA_SNGI_DEPARTMENT
CA_SNGI_COST	<input type="checkbox"/>	CA_SNGI_COST
IN_SNGI_STOCK	<input type="checkbox"/>	IN_SNGI_STOCK
ON_SNGI_ORDER	<input type="checkbox"/>	ON_SNGI_ORDER
FILL_2	<input type="checkbox"/>	FILL_2
CA_ORDER_REQUEST redefines CA_REQUEST_SPECIFIC	<input type="checkbox"/>	CA_ORDER_REQUEST

___18. Right-click on DFH0XCP1 and select **Edit data structure**.

___19. Enter an **Interface rename** of **inquireSingle** and click **OK**.



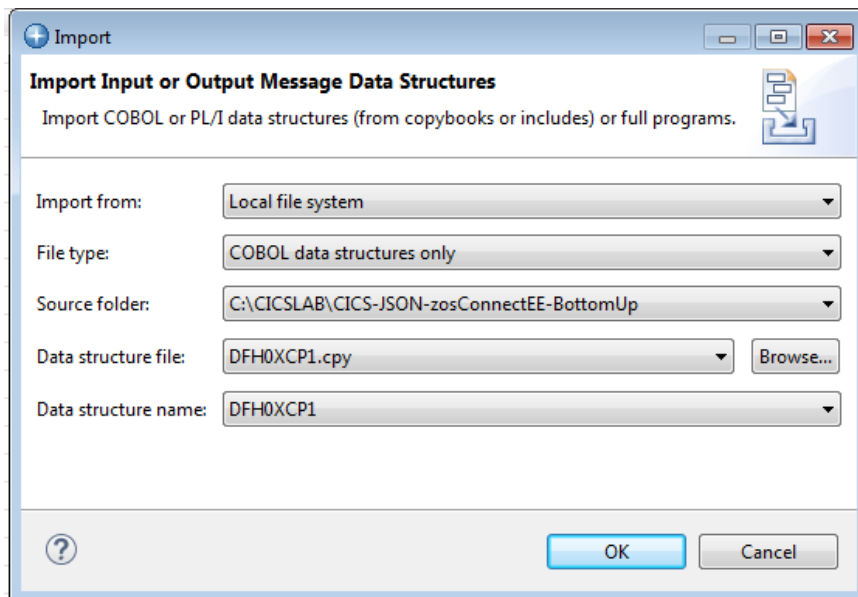
___20. Save and close the **inquireSingle_request** editor.

Create the response service interface

___21. From the Define Request and Response Service Interfaces section, click the **Create Service Interface** button and enter a Service interface name of **inquireSingle_response**. Then click **OK**.

___22. From the Service Interface Definition dialog, **right-click** **COMMAREA**. Then click **Import data structure**.

___23. From the Import dialog, click the **Browse** button and select **C:\CICSLAB\CICS-JSON-zosConnectEE-BottomUp\DFH0XCP1.cpy**. Click **Open**. This will parse the supplied copybook.



___24. Click **OK**.

___25. Expand **COMMAREA > DFH0XCP1**. This will allow you to edit the Response service interface (output). Notice that there are three redefines visible (there are four in the structure). Expand the second one (**CA_INQUIRE_SINGLE**) fully. This is used for an inquire single item.

___26. Uncheck the **CA_REQUEST_SPECIFIC** (Redefined) and check the **CA_INQUIRE_SINGLE** redefines **CA_REQUEST_SPECIFIC**.

___27. In the **CA_INQUIRE_SINGLE** structure, uncheck **FILL_0**, **FILL_1**, **FILL_2**.

Fields	Include	Interface rename	D
COMMAREA			
DFH0XCP1			
CA_REQUEST_ID	<input checked="" type="checkbox"/>	CA_REQUEST_ID	
CA_RETURN_CODE	<input checked="" type="checkbox"/>	CA_RETURN_CODE	
CA_RESPONSE_MESSAGE	<input checked="" type="checkbox"/>	CA_RESPONSE_MESSAGE	
CA_REQUEST_SPECIFIC (Redefined)	<input type="checkbox"/>	CA_REQUEST_SPECIFIC	
CA_INQUIRE_REQUEST redefines CA_REQUEST_SPECIFIC	<input type="checkbox"/>	CA_INQUIRE_REQUEST	
CA_INQUIRE_SINGLE redefines CA_REQUEST_SPECIFIC	<input checked="" type="checkbox"/>	CA_INQUIRE_SINGLE	
CA_ITEM_REF_REQ	<input checked="" type="checkbox"/>	CA_ITEM_REF_REQ	
FILL_0	<input type="checkbox"/>	FILL_0	
FILL_1	<input type="checkbox"/>	FILL_1	
CA_SINGLE_ITEM	<input checked="" type="checkbox"/>	CA_SINGLE_ITEM	
CA_SNGI_ITEM_REF	<input checked="" type="checkbox"/>	CA_SNGI_ITEM_REF	
CA_SNGI_DESCRIPTION	<input checked="" type="checkbox"/>	CA_SNGI_DESCRIPTION	
CA_SNGI_DEPARTMENT	<input checked="" type="checkbox"/>	CA_SNGI_DEPARTMENT	
CA_SNGI_COST	<input checked="" type="checkbox"/>	CA_SNGI_COST	
IN_SNGI_STOCK	<input checked="" type="checkbox"/>	IN_SNGI_STOCK	
ON_SNGI_ORDER	<input checked="" type="checkbox"/>	ON_SNGI_ORDER	
FILL_2	<input type="checkbox"/>	FILL_2	
CA_ORDER_REQUEST redefines CA_REQUEST_SPECIFIC	<input type="checkbox"/>	CA_ORDER_REQUEST	

28. Save and close the **inquireSingle_response** editor.

29. Using the drop-down, select **inquireSingle_response.si** for Response service interface.

▼ 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: inquireSingle_request.si Edit

Response service interface: inquireSingle_response.si Edit

Set advanced data conversion options Advanced Options...

From the **inquireSingle Service** editor, select the **Configuration** tab across the bottom. Enter a Connection reference of **catalog**.

*inquireSingle01 Service

Service Project Editor: Configuration

▼ Required Configuration

Enter the required configuration for this service.

Coded character set identifier (CCSID): 37

Connection reference: catalog

▼ Optional Configuration

Enter the optional configuration for this service.

Transaction ID:

Transaction ID usage: ▼

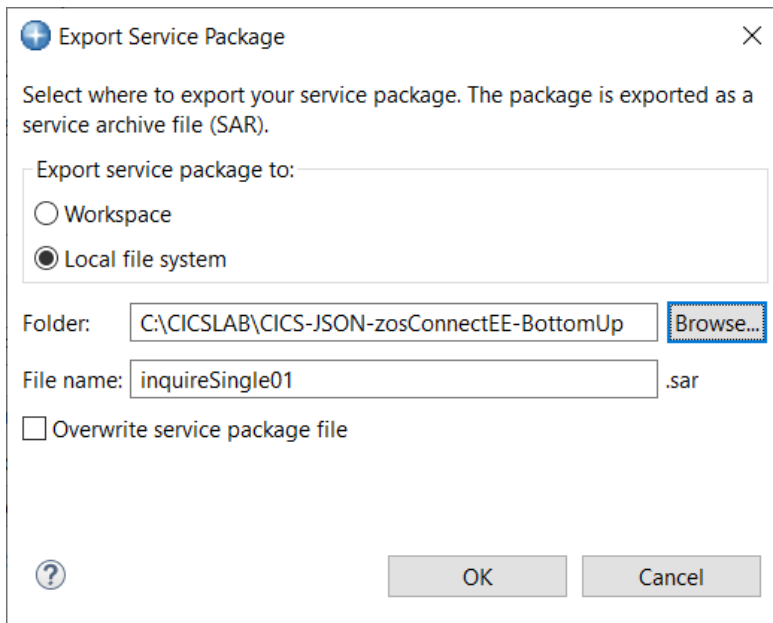
Bidi configuration reference:

___30. Save and close the inquireSingle Service editor.

Export the SAR file.

___31. **Right-click** on your **inquireSingle** project. Select **z/OS Connect EE > Export z/OS Connect EE Service Archive**.

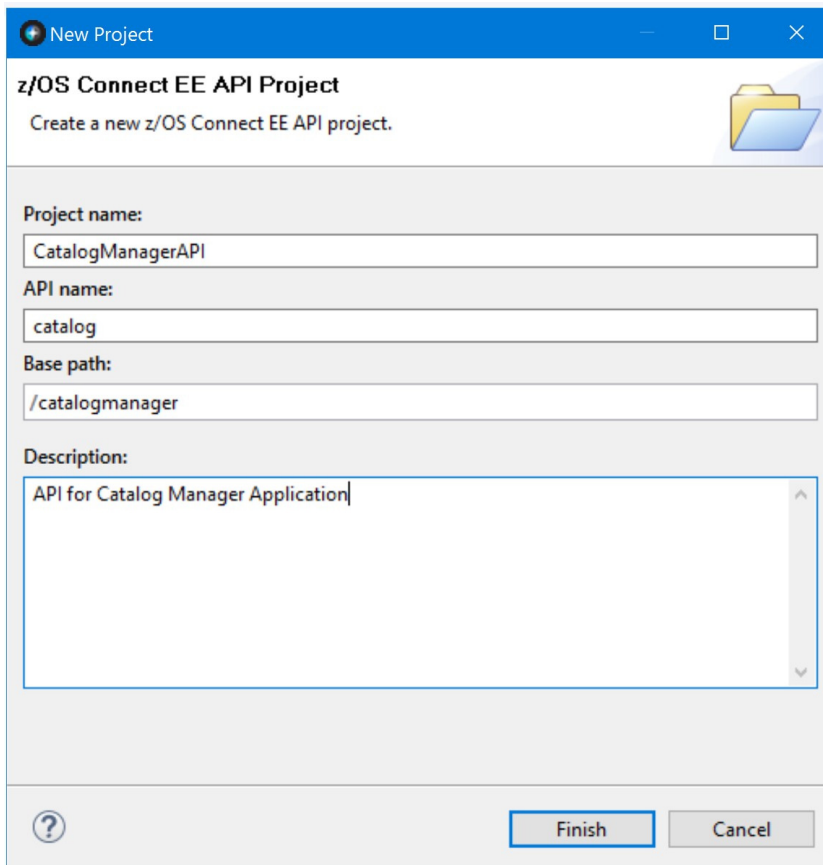
___32. From the Export Service Package dialog, select **Local file system**. Use the **Browse** button to select folder **C:\CICSLAB\CICS-JSON-zosConnectEE-BottomUp** and click **OK**.



Part 4: Build the API Application

Build the API application

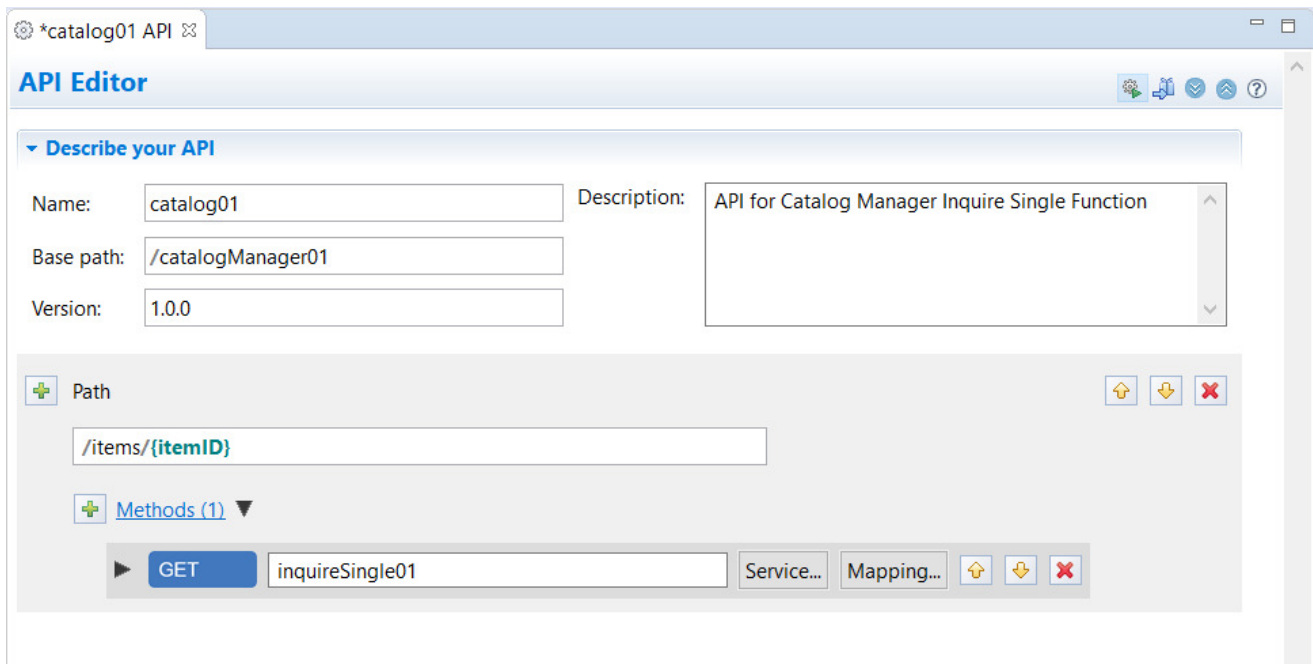
- ___1. From the **CICS Explorer**, the **z/OS Connect Enterprise Edition** perspective, the **Project Explorer** view, select **File > New > z/OS Connect EE API Project**.
- ___2. From the **New Project** dialog, supply the **Project name**, **API name**, **Base path**, and **Description**, based on the sample below.
- ___3. Then click **Finish**.



The screenshot shows the 'New Project' dialog box in CICS Explorer. The title bar says 'New Project'. The main title is 'z/OS Connect EE API Project'. Below it, it says 'Create a new z/OS Connect EE API project.' There is a folder icon on the right. The dialog has four input fields: 'Project name:' with the value 'CatalogManagerAPI', 'API name:' with the value 'catalog', 'Base path:' with the value '/catalogmanager', and 'Description:' with the value 'API for Catalog Manager Application'. At the bottom, there is a question mark icon, a 'Finish' button, and a 'Cancel' button.

- ___4. This will place you in the **z/OS Connect EE API Editor**.
- ___5. We will only configure the **GET** method for our API. This will invoke the **inquireSingle** service.
- ___6. Remove the **POST**, **PUT**, and **DELETE** methods by clicking the red “X” on the right.

Enter a **Path** of **/items/{itemID}**, replacing **/newPath1** (See screen capture below).



- ___7. Click **Service...** to the right of **GET**.
- ___8. In the **Select a z/OS Connect EE Service** dialog, click **File System**.
- ___9. Navigate to **C:\CICSLAB\CICS-JSON-zosConnectEE-BottomUp**.
- ___10. Click **C:\CICSLAB\CICS-JSON-zosConnectEE-BottomUp\inquireSingle.sar**, then click **Open**. Click **OK** and **OK** on the next two panels. This will import the inquireSingle SAR file into your project. This is the SAR file that you created and exported in the prior steps.
- ___11. Use **Ctrl-S** to **save** your API project.

Create request mapping

- ___12. For your **GET** method click **Mapping > Open Request Mapping**.
- ___13. **Right-click** in the open area and select **Expand All** from the request editor.

GET.items.{itemID}

Transferred

Field Name	Data Type	Value
CA_REQUEST_ID	[0..1] string	
CA_RETURN_CODE	[0..1] integer	
CA_RESPONSE_MESSAGE	[0..1] string	
CA_INQUIRE_SINGLE	[0..1]	
CA_ITEM_REF_REQ	[0..1] integer	

HTTP Request

HTTP Headers

Authorization [0..1] string

Path Parameters

itemID [1..1] string

Query Parameters

Body - DFH0XCP1

For a **request** map, three sections of fields are represented:

Right side: This section represents the COMMAREA fields seen in the COPYBOOK when the Service project was created to produce the SAR file.

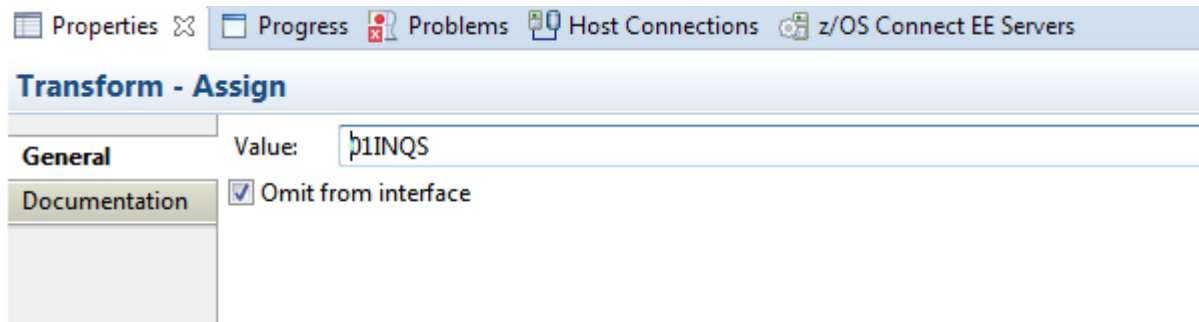
Upper-left side: This section represents the fields exposed to the REST client. Initially there is a one-to-one mapping; that is, initially all the COMMAREA fields are exposed to the REST client. But you control which fields are exposed and which are hidden by your actions in the API Editor.

Lower-left side: This section represents the information fields available on the HTTP request, such as authorization headers, path parameters, query parameters, or information in the body of a request.

- ___ **14.** On the right side of the screen, right click on **CA_REQUEST-ID** and select **Add Assign transform**.

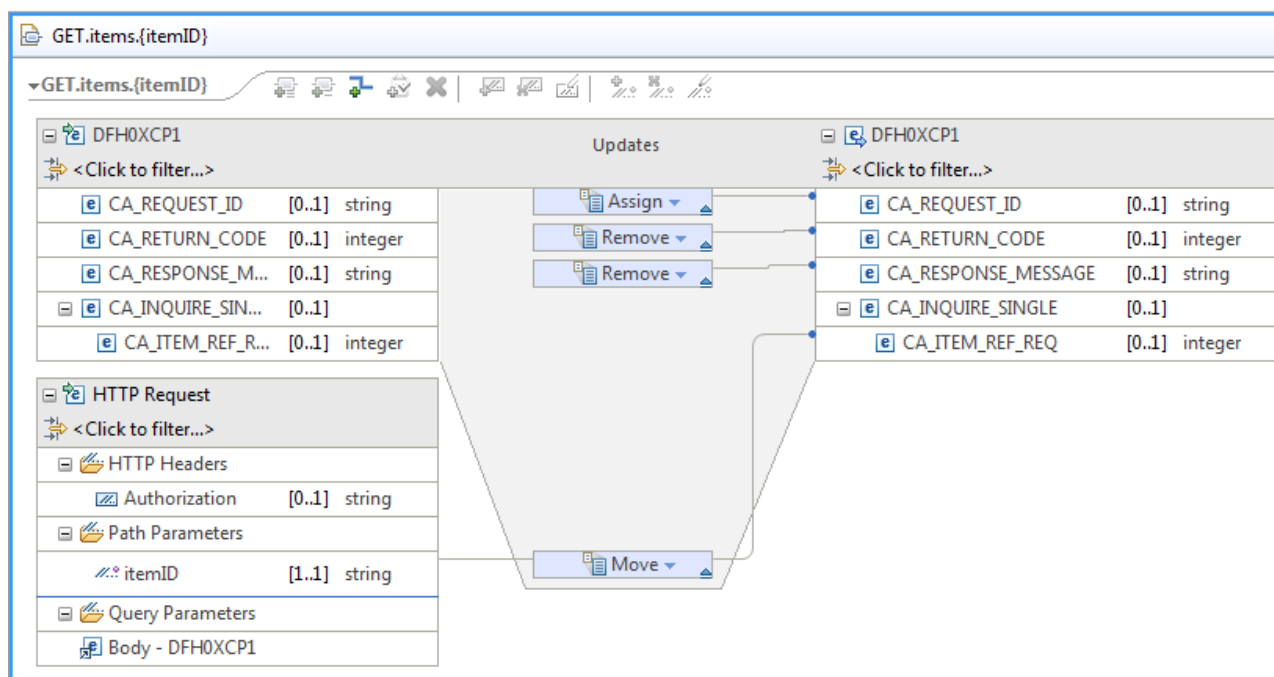
Note: based on our knowledge of the Catalog Manager application, we know the **ca_request_id** field for **inquireSingle** is **01INQS**. So we are going to assign that field the static value of **01INQS** in the API Editor. z/OS Connect EE will populate that field with **01INQS** whenever this API is called. (This can also be added when creating the service)

- ___ **15.** Click the **Assign** tag, then in the **Properties** view (near the bottom of the screen). Enter a value of **01INQS**.



This assigns the static value to the field. The checkbox **Omit from interface** means the REST client does not see this field as part of the API.

- ___16. On the right side of the screen, right-click on **CA_RETURN_CODE** and select **Add Remove transform**. (This just show the flexibility of the tooling)
- ___17. On the right side of the screen, right-click on **CA_RESPONSE_MESSAGE** and select **Add Remove transform**.
- ___18. Build a **Move** connection between the path parameter **itemID** on the left **CA_ITEM_REF_REQ** on the right... Right-click on the **itemID** under **Path Parameters** on the left and select **Add Connection**. A connection line will appear and will follow your mouse. Move your mouse over to **CA_ITEM_REF_REQ** on the right side, then **click** your mouse. That will build the connection from **itemID** to **CA_ITEM_REF_REQ** and add a **Move** element. (You can also just drag a line between the two fields)



___ **19. Save** then **close** the request mapping dialog.

Create response mapping

___ **20. For your GET method click Mapping > Open Default Response Mapping.**

___ **21. Right-click** in the open area and select **Expand All** from the popup dialog.

This mapping allows us to determine what information is to be sent back to the REST client. The right side represents the COMMAREA fields, and the left side represents the fields that will be provided in the JSON response.

___ **22. On the right side of the screen, right-click on CA_REQUEST_ID and select Add Remove transform.**

___ **23. On the right side of the screen, right-click on CA_ITEM_REF_REQ and select Add Remove transform.**

___ **24. You have removed two fields and allowed the rest to map over to the JSON that will flow back to the client.**

The screenshot shows the 'Response Mapping' dialog in the IBM API Catalog. The dialog is titled 'GET.items.{itemID}' and 'DFH0XCP1'. It displays two side-by-side tables of fields. The left table lists fields from the GET method, and the right table lists fields from the HTTP Response. The 'CA_ITEM_REF_REQ' field is highlighted in green in the right table. A 'Remove' button is visible between the two tables, indicating that fields have been removed from the mapping.

Field	Type
CA_REQUEST_ID	[0..1] string
CA_RETURN_CODE	[0..1] integer
CA_RESPONSE_MESSAGE	[0..1] string
CA_INQUIRE_SINGLE	[0..1]
CA_ITEM_REF_REQ	[0..1] integer
CA_SINGLE_ITEM	[0..1]
CA_SNGI_ITEM_REF	[0..1] integer
CA_SNGI_DESCRIPTI...	[0..1] string
CA_SNGI_DEPARTM...	[0..1] integer
CA_SNGI_COST	[0..1] string
IN_SNGI_STOCK	[0..1] integer
ON_SNGI_ORDER	[0..1] integer

___ **25. Save** and **close** the **Response Mapping** dialog.

___ **26. Close** the catalog **API editor**.

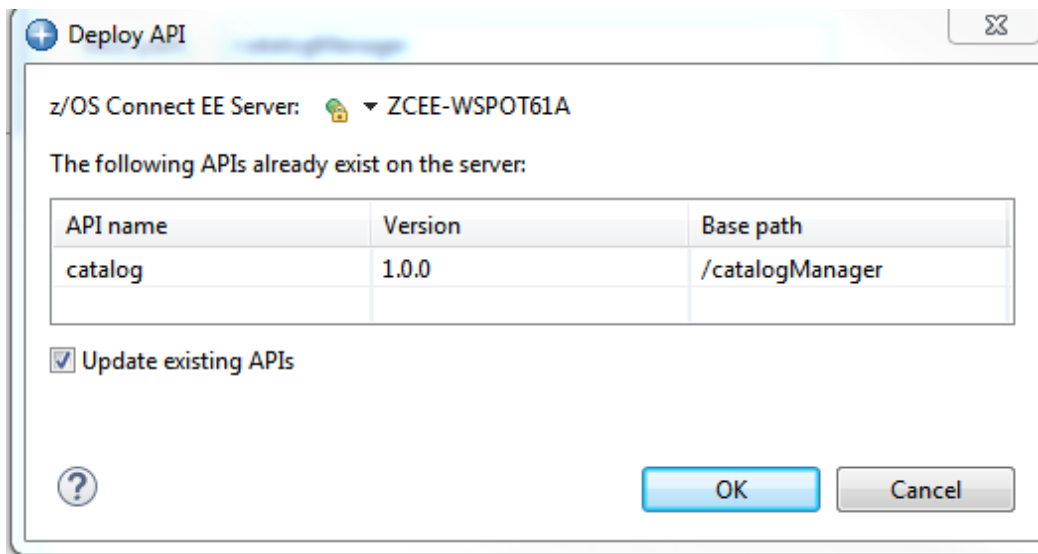
Part 5: Deploy API and Service to z/OS Connect EE Server

Deploy API to z/OS Connect EE Server

- ___ **27.** ~~We have defined the connection for you, so skip to Step 30.~~ If you do not have a connection, continue to next step.

Hint: If you do not have a connection, go to the **Host Connections** view to create a connection for **z/OS Connect Enterprise Edition**.

- ___ **28.** **Right-click** on your **new** connection and select **Set Credentials > USER1**.
- ___ **29.** From the **z/OS Connect Enterprise Edition** perspective, the **z/OS Connect EE Servers** view, **right-click** your z/OS Connect Server and select **Connect**.
- ___ **30.** From the **z/OS Connect Enterprise Edition** perspective, the **Project Explorer** view, **right-click** your **CatalogManagerAPI** project and select **z/OS Connect EE > Deploy API to z/OS Connect EE Server**.
- ___ **31.** If you are redeploying an API, ensure that you check the **Update Existing APIs** box and click **OK**.

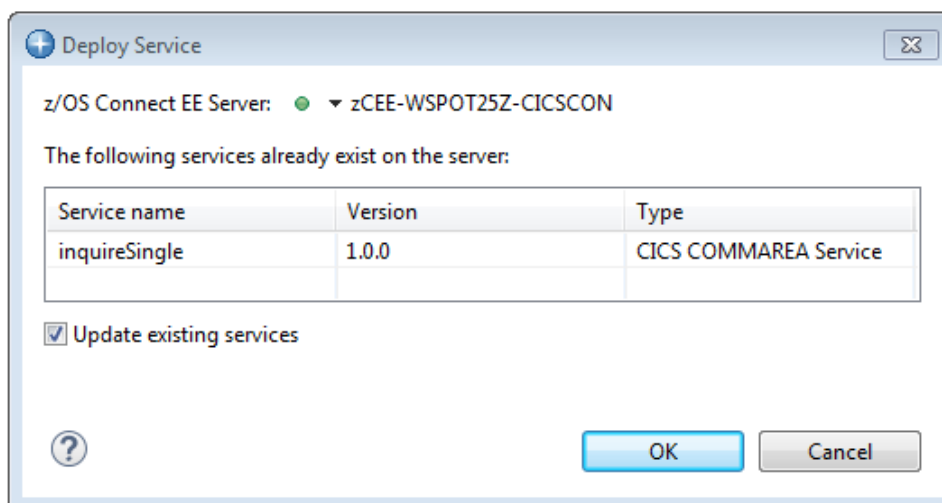


___32. If your API was not successfully deployed, contact a lab instructor.

Deploy Service (inquireSingle) to z/OS Connect EE Server

___33. From the **z/OS Connect Enterprise Edition** perspective, the **Project Explorer** view, **right-click** your **inquireSingle** project and select **z/OS Connect EE > Deploy Service to z/OS Connect EE Server**.

___34. If you are redeploying a service, ensure that you check the **Update existing services** box and click **OK**.



___35. If your Service (inquireSingle) was not successfully deployed, contact a lab instructor.

Part 6: Test Your Work

We will be testing with the RestClient Firefox plug-in as a generic REST client.

Note: Screen shots are just examples.

Open **Firefox** browser, and on the URL line type

<http://wg31.washington.ibm.com:9090/zosConnect/services/inquireSingle>



From the **Firefox browser**, press the button in the upper-right corner.

If you hover over the button, it says “RESTClient”.

Get the Swagger document.

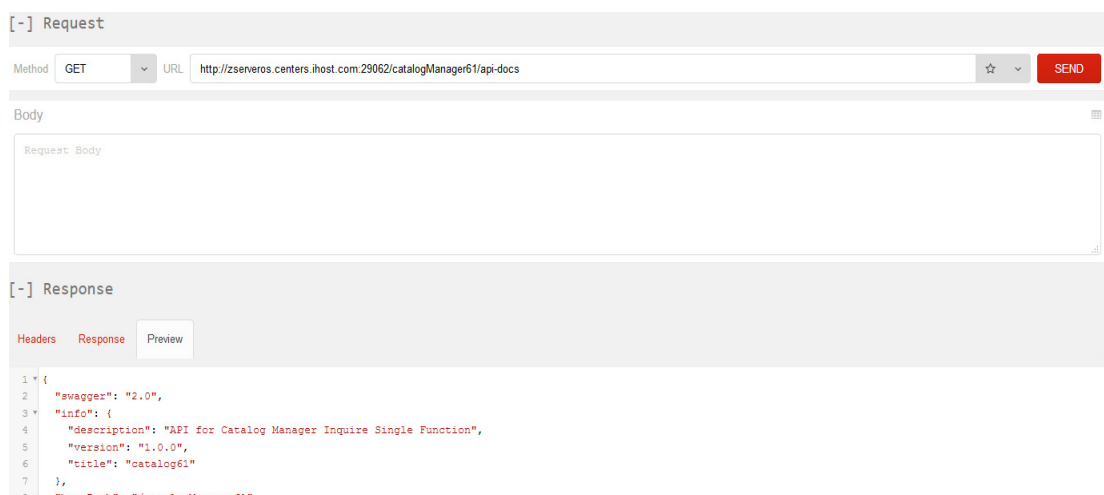
The Swagger (OpenAPI) document represents the API. The API package includes Swagger 2.0 definitions to make it easier for developers to incorporate the APIs into their applications

___**36.** From the **REST Client**, select the Method **GET**, and enter a **URL** of

<http://wg31.washington.ibm.com:9090/catalogmanager/api-docs>

___**37.** Then press the **SEND** button.

___**38.** Select the **Preview** tab in the RESTClient.



Read (GET) a record

We only implemented the GET function. See the “Getting Started Guide” to implement the other functions.

<http://www-03.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/WP102724>

___39. From the **REST Client**, select the Method **GET**, and enter a **URL** of

<http://wg31.washington.ibm.com:9090/catalogmanager/items/0010>

___40. Then press the **SEND** button and select the **Preview** tab to see results.

The screenshot displays the REST Client interface. At the top, the 'Request' section is active, showing the Method set to 'GET' and the URL 'http://zserveros.centers.ihost.com:19025/catalogManager/items/0010'. A 'SEND' button is visible. Below this, the 'Body' section is empty. The 'Response' section is also active, with the 'Preview' tab selected. It shows a JSON response with the following structure:

```
1 {
2   "DFHOXCP1": {
3     "CA_RESPONSE_MESSAGE": "RETURNED ITEM: REF =0010",
4     "CA_INQUIRE_SINGLE": {
5       "CA_SINGLE_ITEM": {
6         "CA_SNGL_ITEM_REF": 10,
7         "CA_SNGL_DESCRIPTION": "Ball Pens Black 24pk",
8         "CA_SNGL_DEPARTMENT": 10,
9         "IN_SNGL_STOCK": 132,
10        "CA_SNGL_COST": "002.90",
11        "ON_SNGL_ORDER": 0
12      }
13    },
14    "CA_RETURN_CODE": 0
15  }
16 }
```

Test using API Explorer

- ___41. From the **CICS Explorer**, the **z/OS Connect Enterprise Edition** perspective, the **z/OS Connect EE Servers** view, right-click on your **catalog** API and select **Open In API Explorer** from the pop-up dialog. A Browser/REST API Documentation window will open.
- ___42. Click **List Operations**, then click **GET**. This will expand your GET method.
- ___43. Enter **0010** in the **itemID** field, then click **Try it out!**.

Liberty REST APIs

Discover REST APIs available within Liberty

catalog01
Show/Hide | List Operation

GET /catalogmanager/items/{itemID}

Response Class (Status 200)
OK

Model | Example Value

```
{
  "DFH0XCP1": {
    "CA_RETURN_CODE": 0,
    "CA_RESPONSE_MESSAGE": "string",
    "CA_INQUIRE_SINGLE": {
      "CA_SINGLE_ITEM": {
        "CA_SNGL_ITEM_REF": 0,
        "CA_SNGL_DESCRIPTION": "string",
        "CA_SNGL_DEPARTMENT": 0,
        "CA_SNGL_COST": "string",

```

Response Content Type application/json ▾

Parameters

Parameter	Value	Description	Parameter Type	Data Type
Authorization	<input type="text"/>		header	string
itemID	<input type="text" value="0010"/>		path	string

Try it out! [Hide Response](#)

- ___44. Scroll down to see your results.

Response Body

```
{
  "DFH0XCP1": {
    "CA_RESPONSE_MESSAGE": "RETURNED ITEM: REF =0010",
    "CA_INQUIRE_SINGLE": {
      "CA_SINGLE_ITEM": {
        "CA_SINGL_ITEM_REF": 10,
        "CA_SINGL_DESCRIPTION": "Ball Pens Black 24pk",
        "CA_SINGL_DEPARTMENT": 10,
        "IN_SINGL_STOCK": 135,
        "CA_SINGL_COST": "002.90",
        "ON_SINGL_ORDER": 0
      }
    },
    "CA_RETURN_CODE": 0
  }
}
```

Response Code

200

Contact a lab instructor if you have any questions.

Part 7: Summary

Congratulations, you have just exposed a CICS application as a RESTful JSON service using z/OS Connect EE. You used the bottom-up approach which means that your JSON service conforms to the copybook of your CICS COBOL program.

- You created a service project using your COBOL copybook.
- You built a z/OS Connect EE API Project (API application) using the CICS Explorer.
- You deployed the service (SAR) file to z/OS Connect EE.
- You deployed the API application to z/OS Connect EE.
- You tested your new JSON service.

If you have any questions about anything you did in this lab exercise, please contact one of the lab instructors.