



# ZCINTRO - IBM z/OS Connect

## An introduction and overview

Mitch Johnson

[mitchj@us.ibm.com](mailto:mitchj@us.ibm.com)

John Brefach

Washington System Center



# Agenda

- An Introduction and Overview of using REST API
- Enabling RESTful API to various z/OS resources, e.g.
  - CICS
  - Db2
  - IMS/TM
  - IMS/DB
  - MQ
  - Outbound REST APIs
- Accessing RESTful API from z/OS COBOL Applications
- A brief introduction to z/OS Connect Security

\*For more on administration and security, contact your local IBM rep regarding the schedule of workshop *zCADMIN IBM z/OS Connect Administration Wildfire Workshop*

# Notes and Disclaimers



- The information in this presentation was derived from various product documentation web sites.
- Additional information included in this presentation was distilled from years of experience implementing security using RACF with z/OS products like CICS, IMS, Db2, MQ, etc. as well as Java runtimes environments like WebSphere Application Server and WebSphere Application Server Liberty which is commonly called Liberty.
- There will be additional information on slides that will be designated as Tech/Tips. These contain information that at perhaps at least interesting and hopefully, useful to the reader.
- **IBM z/OS Connect (OpenAPI 2)** refers to the z/OS Connect EE product prior to service level V3.0.55. **IBM z/OS Connect (OpenAPI 3)** refers to the additional functions and features added with service level V3.0.55. Important - servers configured for OpenAPI 2 can will continue to operate as is with service level V3.0.55 and later.
- A z/OS Connect OpenAPI 2, or a z/OS Connect OpenAPI 3 icon will appear on slides where the information is specific to these products. Don't hesitate to ask questions as to why the icon does or does not appear on certain slides.
- The examples, tips, etc. present in this material are based on firsthand experiences and are not necessarily sanctioned by Liberty or z/OS Connect development.

# This session is part of a series of z/OS Connect workshops. . .



## ZCREQUEST - IBM z/OS Connect An introduction to API Requesters

API Requester Code and Security Considerations

Mitch Johnson

[mitchj@us.ibm.com](mailto:mitchj@us.ibm.com)

Washington System Center

mitchj@us.ibm.com



## z/OS Connect Open API 3

Designer and z/OS Native server  
Experiences and Observations

Mitch Johnson

[mitchj@us.ibm.com](mailto:mitchj@us.ibm.com)

Washington Systems Center

[mitchj@us.ibm.com](mailto:mitchj@us.ibm.com)



© 2022 IBM Corporation  
Slide 1

[mitchj@us.ibm.com](mailto:mitchj@us.ibm.com)

<https://www.ibm.com/support/pages/mainframe-system-education-wildfire-workshops>

© 2017, 2023 IBM Corporation  
Slide 4



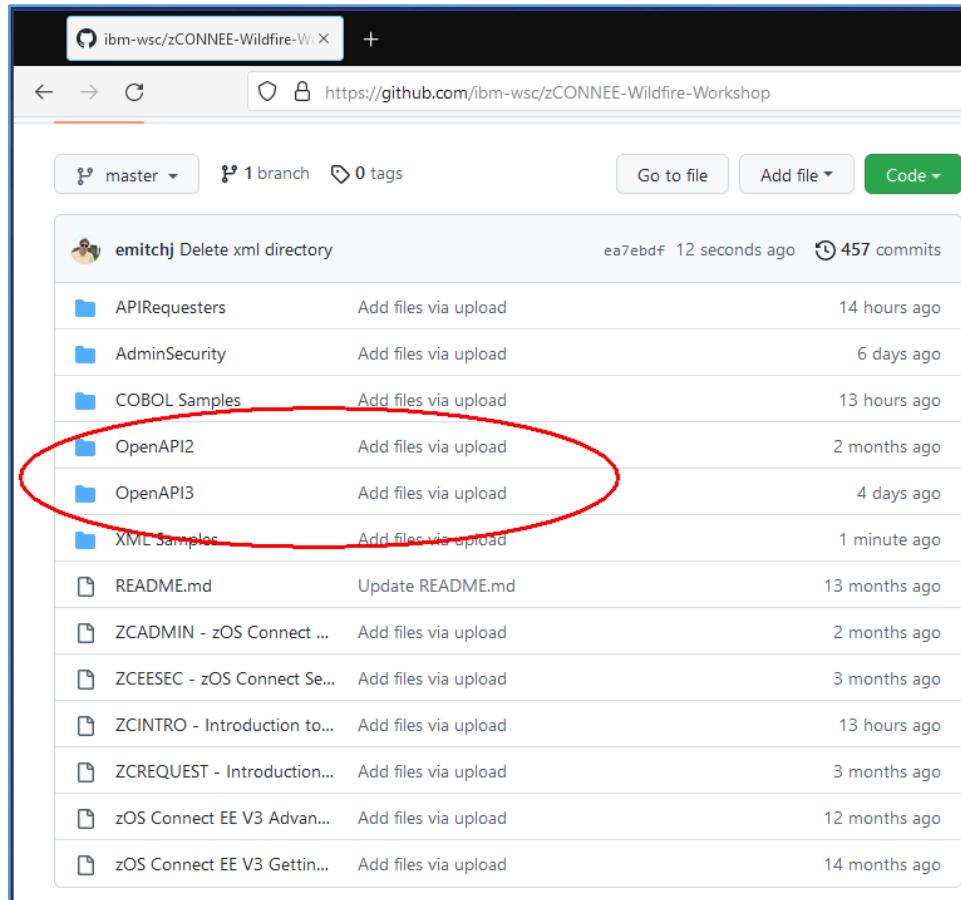
## ZADMIN – IBM z/OS Connect Administration

WebSphere Liberty Profile with  
IBM z/OS Connect (OpenAPI 2) and/or  
IBM z/OS Connect (OpenAPI 3)  
Administration



© 2017, 2022 IBM Corporation  
Slide 1

# z/OS Connect Wildfire Github Site



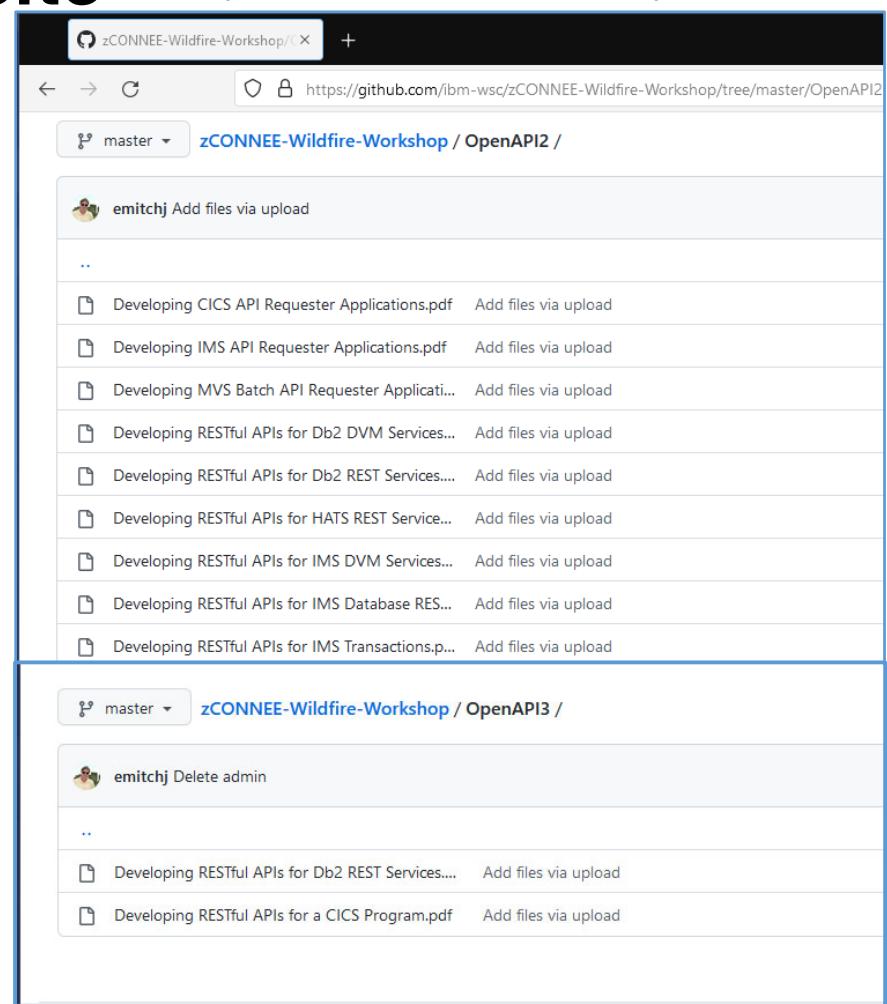
A screenshot of a GitHub repository page. The repository name is 'ibm-wsc/zCONNEE-Wildfire-Workshop'. The commit list shows several entries:

- emitchj Delete xml directory ea7ebdf 12 seconds ago 457 commits
- APIRequesters Add files via upload 14 hours ago
- AdminSecurity Add files via upload 6 days ago
- COBOL Samples Add files via upload 13 hours ago
- OpenAPI2** Add files via upload 2 months ago
- OpenAPI3** Add files via upload 4 days ago
- XML Samples Add files via upload 1 minute ago
- README.md Update README.md 13 months ago
- ZCADMIN - zOS Connect ... Add files via upload 2 months ago
- ZCEESEC - zOS Connect Se... Add files via upload 3 months ago
- ZCINTRO - Introduction to... Add files via upload 13 hours ago
- ZCREQUEST - Introduction... Add files via upload 3 months ago
- zOS Connect EE V3 Advan... Add files via upload 12 months ago
- zOS Connect EE V3 Gettin... Add files via upload 14 months ago

mitchj@us.ibm.com

- Contact your IBM representative to request access to these exercises

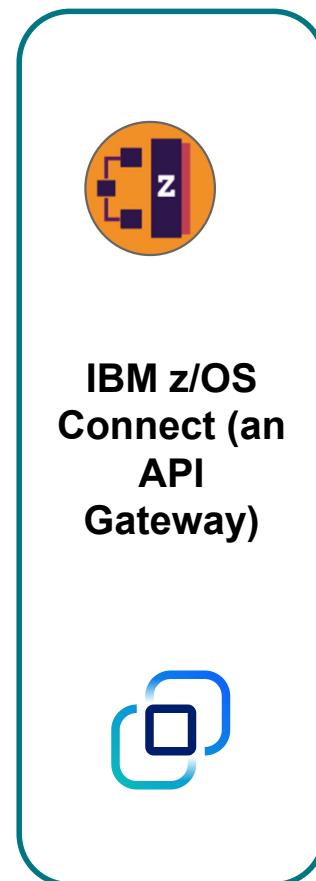
<https://ibm.biz/zCEEWorshopMaterial>



The screenshot shows two sub-folders under the 'zCONNEE-Wildfire-Workshop' repository:

- OpenAPI2 /**
  - emitchj Add files via upload
  - Developing CICS API Requester Applications.pdf
  - Developing IMS API Requester Applications.pdf
  - Developing MVS Batch API Requester Application.pdf
  - Developing RESTful APIs for Db2 DVM Services....
  - Developing RESTful APIs for Db2 REST Services....
  - Developing RESTful APIs for HATS REST Service...
  - Developing RESTful APIs for IMS DVM Services...
  - Developing RESTful APIs for IMS Database RES...
  - Developing RESTful APIs for IMS Transactions.pdf
- OpenAPI3 /**
  - emitchj Delete admin
  - Developing RESTful APIs for Db2 REST Services....
  - Developing RESTful APIs for a CICS Program.pdf

# **z/OS Connect EE exposes z/OS resources to the “cloud” via RESTful APIs**

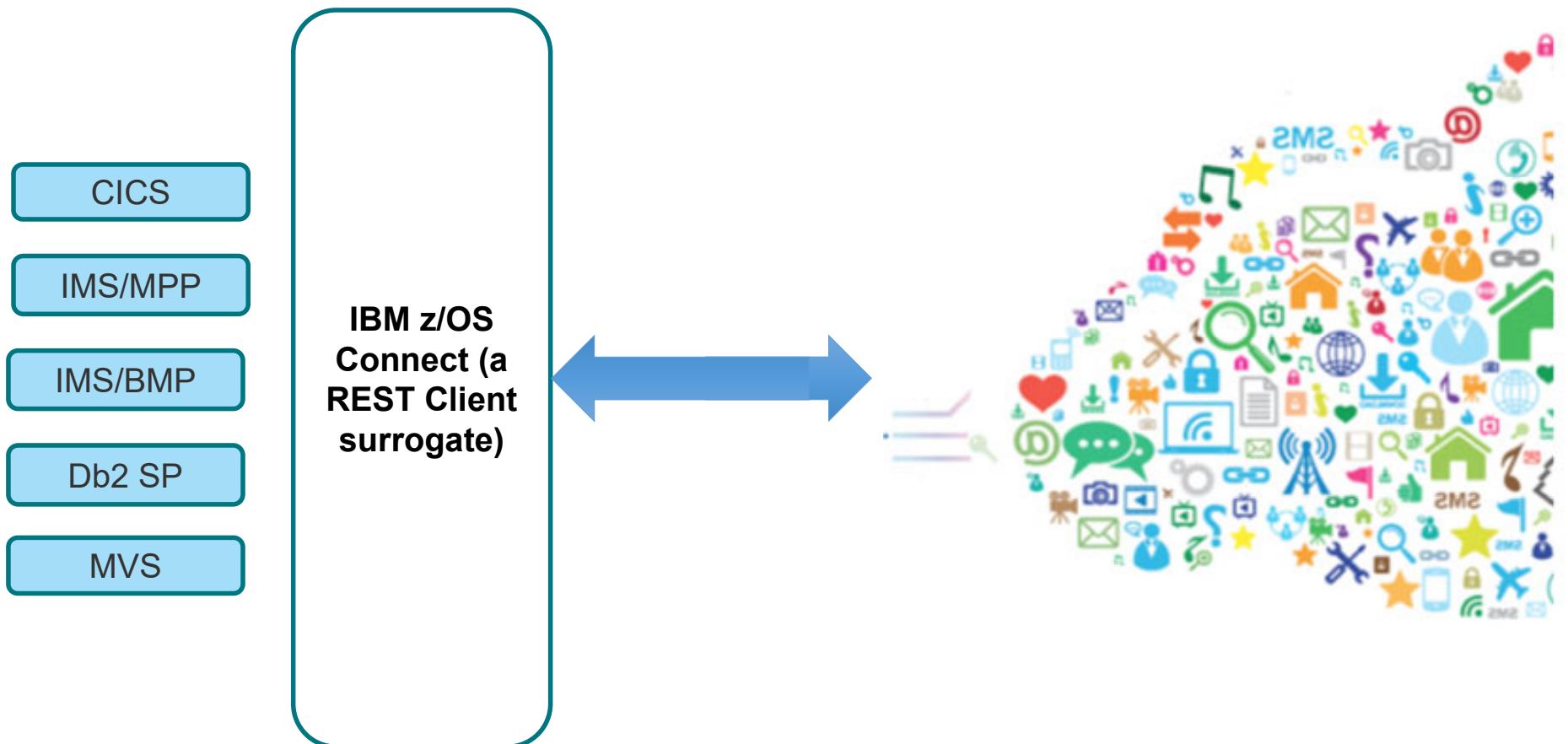


+ HCL and Rocket Software

\*Other Vendors or your own implementation



# z/OS Connect EE exposes external REST APIs in the “cloud” to z/OS applications



**Before we go any further, let ask**

**/what\_is\_REST?**

And what makes an API “RESTful”?



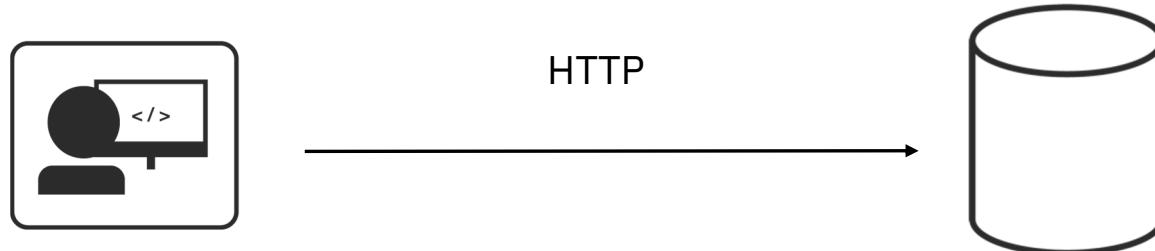
# REST is architectural programming style

**REST** is acronym standing for **R**epresentational **S**tate **T**ransfer.

An architectural programming style for **accessing** and **updating** data over the internet.

Typically using HTTP... but not all HTTP interfaces are “RESTful”.

Simple and intuitive for the end consumer (**the developer**).



Roy Fielding defined REST in his 2000 PhD dissertation "Architectural Styles and the Design of Network-based Software Architectures" at UC Irvine. He developed the REST architectural style in parallel with HTTP 1.1 of 1996-1999, based on the existing design of HTTP 1.0 of 1996.



# Key Principles of the REST API

Use HTTP verbs for Create, Read, Update, Delete (CRUD) operations

POST  
GET  
PUT  
DELETE

`http://<host>:<port>/path/parameter?name=value&name=value`

Use Path and Query parameters to refine the request

URI path identifies a resource (or lists of resources)

URL identifies the protocol, host and port and includes the URI Path

Request/Response Body is used to represent the data object

```
GET http://www.acme.com/customers/12345?personalDetails=true
RESPONSE: HTTP 200 OK
BODY { "id" : 12345
      "name" : "Joe Bloggs",
      "address" : "10 Old Street",
      "tel" : "01234 123456",
      "dateOfBirth" : "01/01/1980",
      "maritalStatus" : "married",
      "partner" : "http://www.acme.com/customers/12346" }
```



# REST vs RESTful

REST is an architectural style of development having these principles plus..

- It should be stateless (transaction management should be managed by the client)
- It should access and/or identify all server resources using only a single URI
- For performing CRUD operations, it should use HTTP verbs such as get, post, put and delete
- It should return the result only in the form of consistent and simple JSON

When an API follows these basic principles, it is considered a RESTful API, whereas a REST API only follows some but not all the above principles

- Remember - Not all REST APIs are RESTful APIs
- The key is consistency, RESTful APIs are consistent with these basic principles, REST APIs are not



# RESTful Examples

**POST** /account/ + (*a JSON request message with Fred's information*)

**GET** /account?number=1234

**PUT** /account/1234 + (*a JSON request message with dollar amount of deposit*)

HTTP Verb conveys the method against the resources; i.e., POST is for create, GET is for balance, etc.

URI conveys the resource to be acted upon; i.e., Fred's account with number 1234

The JSON body carries the specific data for the action (verb) against the resource (URI)

REST APIs are increasingly popular as an integration pattern because it is stateless, relatively lightweight, is relatively easy to program

<https://martinfowler.com/articles/richardsonMaturityModel.html>



# Not every REST API is a RESTful API

(How to know if an API is not RESTful)

## 1. Different URIs with the same method for operations on the same object

POST http://www.acme.com/customers/**GetCustomerDetails**/12345

POST http://www.acme.com/customers/**UpdateCustomerAddress**/12345?**address=**

## 2. Different representations of the same objects between request and response messages

POST http://www.acme.com/customers  
BODY { "firstName": "Joe",  
 "lastName" : "Bloggs",  
 "addr" : "10 Old Street",  
 "phoneNo" : "01234 0123456" }



RESPONSE HTTP 201 CREATED  
BODY { "id" : "12345",  
 "name" : "Joe Bloggs",  
 "address" : "10 New Street"  
 "tel" : "01234 0123456" }

## 3. Operational data (update, etc.) embedded in the request body

POST http://www.acme.com/customers/12345  
BODY { "updateField": "address",  
 "newValue" : "10 New Street" }



RESPONSE HTTP 200 OK  
BODY { "id" : "12345",  
 "name" : "Joe Bloggs",  
 "address" : "10 New Street"  
 "tel" : "01234 123456" }



# The goal is to strive to design API to use RESTful properties

## 1. Use the same URIs for the same resource with the appropriate method for operations

```
GET http://www.acme.com/customers/12345
```

```
PUT http://www.acme.com/customers/12345?address=10%20New%20Street
```

## 2. Use the same JSON property names between request and response messages

```
POST http://www.acme.com/customers/12345  
BODY { "name": "Joe Bloggs",  
       "address": "10 Old Street",  
       "phoneNo": "01234 0123456" }
```



```
RESPONSE HTTP 201  
BODY { "id" : "12345",  
       "name" : "Joe Bloggs",  
       "address" : "10 New Street"  
       "phoneNo": "01234 0123456" }
```

## 3. Use JSON name/value pairs

```
PUT http://www.acme.com/customers/12345  
BODY { "address" : "10 New Street" }
```



```
RESPONSE HTTP 200 OK
```



# Why is REST popular?

|                                    |  |
|------------------------------------|--|
| <b>Ubiquitous Foundation</b>       | It's based on HTTP, which operates on TCP/IP, which is a ubiquitous networking topology.   |
| <b>Relatively Lightweight</b>      | Compared to other technologies (for example, SOAP/WSDL), the REST/JSON pattern is relatively light protocol and data model, which maps well to resource-limited devices.     |
| <b>Relatively Easy Development</b> | Since the REST interface is so simple, developing the client involves very few things: an understanding of the URI requirements (path, parameters) and any JSON data schema. |
| <b>Increasingly Common</b>         | REST/JSON is becoming more and more a de facto "standard" for exposing APIs and Microservices. As more adopt the integration pattern, the more others become interested.     |
| <b>Stateless</b>                   | REST is by definition a stateless protocol, which implies greater simplicity in topology design. There's no need to maintain, replicate or route based on state.             |



# Goal: Client code is unaware of the z/OS infrastructure

**CICS**

```

55
56
57
58
59
60
61
62
63
64
65
66
67
68

// Invoke the REST API using a GET method
URL url = new URL("https://wg31.washington.ibm.com:9453/cscvinc/employee/" + args[1]);
System.out.println("URL: " + url);
HttpsURLConnection conn = (HttpsURLConnection) url.openConnection();
conn.setRequestMethod("GET");
conn.setRequestProperty("Content-Type", "application/json");
byte[] bytesEncoded = Base64.encodeBase64(args[0].getBytes());
conn.addRequestProperty("Authorization", "Basic " + new String(bytesEncoded));
try {
    if (conn.getResponseCode() != 200) {
        throw new RuntimeException("Failed : HTTP error code : " + conn.getResponseCode());
    }
    BufferedReader bufferReader = new BufferedReader(new InputStreamReader((conn.getInputStream())));
    while ((output = bufferReader.readLine()) != null) {
        System.out.println(output);
    }
}

```

Problems @ Javadoc Declaration Console Coverage

<terminated> ZceeCICSGet [Java Application] C:\Program Files\IBM\Java80\jre\bin\javaw.exe (May 7, 2021, 2:54:24 PM)

URL: https://wg31.washington.ibm.com:9453/cscvinc/employee/222222

USERID: CICSUMER  
CEIBRESP0: 0  
CEIBRESP2: 0  
name: DR E. GRIFFITHS  
employeeNumber: 222222  
amount: \$0022.00  
address: FRANKFURT, GERMANY  
phoneNumber: 20034151  
date: 26 11 81  
Response Message: {"cscvincSelectServiceOperationResponse": {"cscvincContainer": {"response": {"CEIBRESP0": "0", "USERID": "CICSUMER", "CEIBRESP2": "0", "name": "DR E. GRIFFITHS", "employeeNumber": "222222", "amount": "0022.00", "address": "FRANKFURT, GERMANY", "phoneNumber": "20034151", "date": "26 11 81"}}}

**Db2**

```

52
53
54
55
56
57
58
59
60
61
62
63
64
65

// Invoke the REST API using a GET method
URL url = new URL("https://wg31.washington.ibm.com:9453/db2/employee/" + args[1]);
System.out.println("URL: " + url);
HttpsURLConnection conn = (HttpsURLConnection) url.openConnection();
conn.setRequestMethod("GET");
conn.setRequestProperty("Content-Type", "application/json");
byte[] bytesEncoded = Base64.encodeBase64(args[0].getBytes());
conn.addRequestProperty("Authorization", "Basic " + new String(bytesEncoded));
try {
    if (conn.getResponseCode() != 200) {
        throw new RuntimeException("Failed : HTTP error code : " + conn.getResponseCode());
    }
    BufferedReader bufferReader = new BufferedReader(new InputStreamReader((conn.getInputStream())));
    while ((output = bufferReader.readLine()) != null) {
        System.out.println(output);
    }
}

```

Problems @ Javadoc Declaration Console Coverage

<terminated> ZceeMqPut [Java Application] C:\Program Files\IBM\Java80\jre\bin\javaw.exe (May 7, 2021, 2:56:06 PM)

URL: https://wg31.washington.ibm.com:9453/db2/employee/000010

Employee Number: 000010  
First Name : CHRISTINE  
Last Name: HAAS  
Middle Initial: I  
Phone Number: 3204

**IMS**

```

53
54
55
56
57
58
59
60
61

URL url = new URL("https://wg31.washington.ibm.com:9453/phonebook/contacts/" + args[1]);
System.out.println("URL: " + url);
HttpURLConnection conn = (HttpURLConnection) url.openConnection();
conn.setRequestMethod("GET");
conn.setRequestProperty("Content-Type", "application/json");
byte[] bytesEncoded = Base64.encodeBase64(args[0].getBytes());
conn.addRequestProperty("Authorization", "Basic " + new String(bytesEncoded));
try {
    if (conn.getResponseCode() != 200) {
        throw new RuntimeException("Failed : HTTP error code : " + conn.getResponseCode());
    }
    BufferedReader bufferReader = new BufferedReader(new InputStreamReader((conn.getInputStream())));
    while ((output = bufferReader.readLine()) != null) {
        System.out.println(output);
    }
}

```

Problems @ Javadoc Declaration Console Coverage

<terminated> ZceeIMSGet [Java Application] C:\Program Files\IBM\Java80\jre\bin\javaw.exe (May 17, 2021, 8:48:25 AM)

URL: https://wg31.washington.ibm.com:9453/phonebook/contacts/LAST1

lastName: LAST1  
firstName: FIRST1  
zipCode: D01/R01  
extension: 8-111-1111  
message: ENTRY WAS DISPLAYED  
HTTP code: 200

# **How do we describe and/or document an API?**



## **/oai/open\_api\_initiative**

The industry standard framework for describing REST APIs

The OpenAPI Initiative (OAI) was created by a consortium of forward-looking industry experts who recognize the immense value of standardizing on how APIs are described. As an open governance structure under the Linux Foundation, the OAI is focused on creating, evolving and promoting a vendor neutral description format. The OpenAPI Specification was originally based on the Swagger Specification, donated by SmartBear Software.



# We use an Open API specification

- It is more than just an API framework



There are a number of tools available to aid consumption:

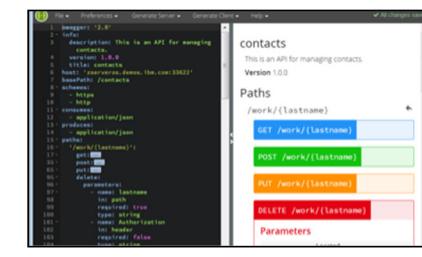
**Code Generation\*** - create stub code to consume APIs from various languages



**Test UIs** - allows API consumers to easily browse and try APIs based on an OpenAPI document.



**Editors** - allows API developers to design their OpenAPI documents.



\* z/OS Connect API Requester

+z/OS Connect, MQ REST support, Zowe

**Important** - You may have used or heard of the term Swagger with the use of APIs. As the use of APIs has grown this term has become in some respects misleading. To be more precise, OpenAPI refers to the API specifications (OpenAPI 2 and OpenAPI3) where Swagger refers to the tooling used to implement the specifications.



## Let's stop and ask what is the significance of the OpenAPI Specification to z/OS Connect?

The industry standard framework for describing REST APIs

- **z/OS Connect and Swagger 2.0 (Open API Specification 2), supported initially by z/OS Connect**

Initially, accessing z/OS resources was the only desire for developing APIs .The interactions with the z/OS resources was driven by the layout of the CICS COMMAREA or CONTAINER, the IMS or MQ messages or the Db2 REST service.

- The details of the interactions with the z/OS resource determined the contents of the API request and response messages and the subsequent specification document.
- **z/OS Connect produces the specification document that describes the methods and request and response messages.**

- **z/OS Connect and Open API Specification 3, supported by z/OS Connect starting in March 2022 service, V3.0.55**

As companies mature their API strategy, they begin to introduce API governance boards to drive consistency in their API design. As more public APIs are created, government and industry standards bodies begin to regulate and drive for standardization. This drives the need for “API first” functional mapping capabilities within the integration platform. The external API design determined the layouts of the API request and response messages provided by the specification documents which was consumed by z/OS Connect to describe the z/OS resource interactions.

- The API details of the methods and layouts of request and response messages are provided in advance and access to the z/OS resource is driven by the API design
- **z/OS Connect consumes the specification document that describes the methods and request and response messages**



# Contrast the z/OS Connect OpenAPI 2 /OpenAPI 3 differences

**z/OS Connect  
OpenAPI2 tooling  
produces an  
OpenAPI 2  
specification  
document, where  
the details of the  
methods and  
request/response  
messages in the API  
specification are  
driven by the nature  
of the z/OS asset  
(JSON format).**

```

cscvinc.json - Notepad
File Edit Format View Help
{
  "swagger": "2.0",
  "info": {
    "description": "",
    "version": "1.0.0",
    "title": "cscvincapi"
  },
  "basePath": "/cscvincapi",
  "schemes": [
    "https",
    "http"
  ],
  "consumes": [
    "application/json"
  ],
  "produces": [
    "application/json"
  ],
  "paths": {
    "/employee/{employee}": {
      "get": {
        "tags": [
          "cscvincapi"
        ],
        "operationId": "getCscvincSelectService",
        "parameters": [
          {
            "name": "Authorization",
            "in": "header",
            "required": false,
            "type": "string"
          },
          {
            "name": "employee",
            "in": "path",
            "required": true,
            "type": "string",
            "maxLength": 6
          }
        ],
        "responses": {
          "200": {
            "description": "OK",
            "schema": {
              "$ref": "#/definitions/getCscvincSelectService_response_200"
            }
          },
          "404": {
            "description": "Not Found"
          }
        }
      }
    }
  }
}

```

Ln 18, Col 7    100%    Windows (CRLF)    UTF-8

```

cscvinc.yaml - Notepad
File Edit Format View Help
openapi: 3.0.1
info:
  title: cscvinc
  description: ""
  version: 1.0.0
servers:
- url: /cscvinc
x-ibm-zcon-roles-allowed:
- Manager
paths:
  /employee:
    post:
      tags:
        - cscvinc
      operationId: postCscvincInsertService
      x-ibm-zcon-roles-allowed:
        - Staff
      parameters:
        - name: Authorization
          in: header
          schema:
            type: string
      requestBody:
        description: request body
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/postCscvincInsertService_request'
      required: true
      responses:
        200:
          description: OK
          content:
            application/json:
              schema:
                $ref: '#/components/schemas/postCscvincInsertService_response_200'
      x-codegen-request-body-name: postCscvincInsertService_request
  /employee/{employee}:
    get:
      tags:
        - cscvinc
      operationId: getEmployee
      x-ibm-zcon-roles-allowed:
        - Staff
      parameters:
        - name: Authorization
          in: header
          schema:
            type: string

```

Ln 44, Col 16    100%    Unix (LF)    UTF-8

**z/OS Connect  
OpenAPI3 tooling  
consumes an  
OpenAPI3 specification  
document and the  
details of the methods  
and request/response  
messages are driven by  
the API specification  
(YAML format\*) and  
not the nature of the  
z/OS asset. Also,  
JSONata can be  
used to augment  
the API response**



## Why /zos\_connect?

Truly RESTful APIs to and from your mainframe.

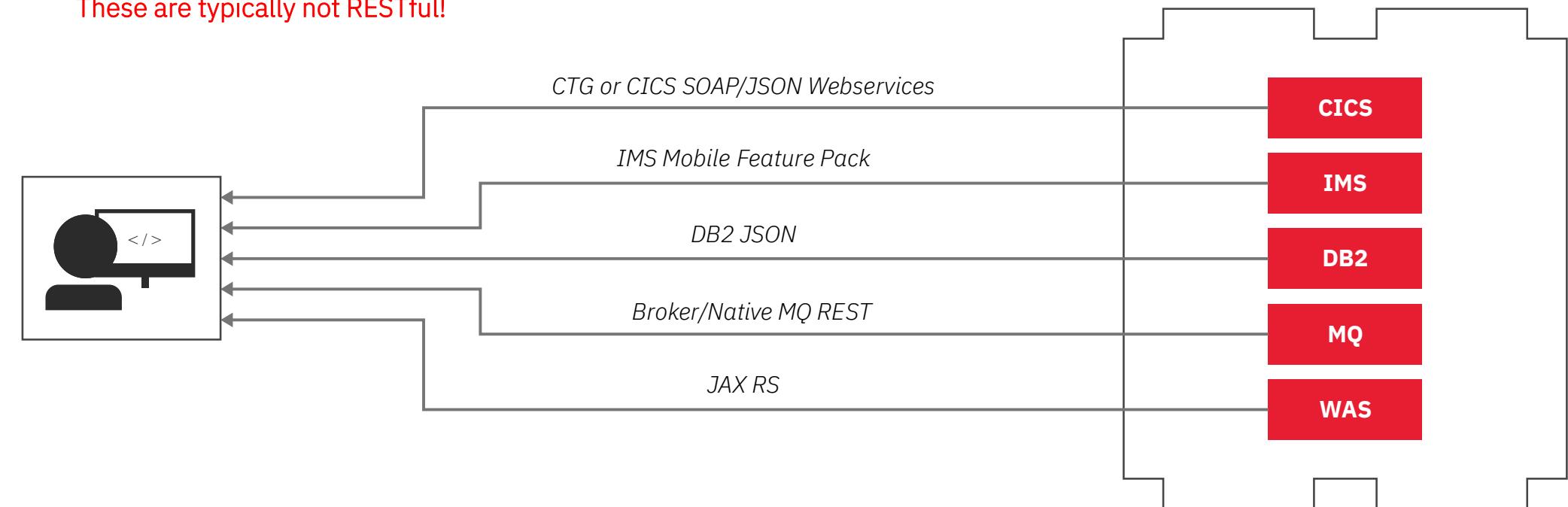


# There was support for REST before z/OS Connect but..

Completely different configuration and management.

Multiple endpoints for developers to call/maintain access to.

These are typically not RESTful!



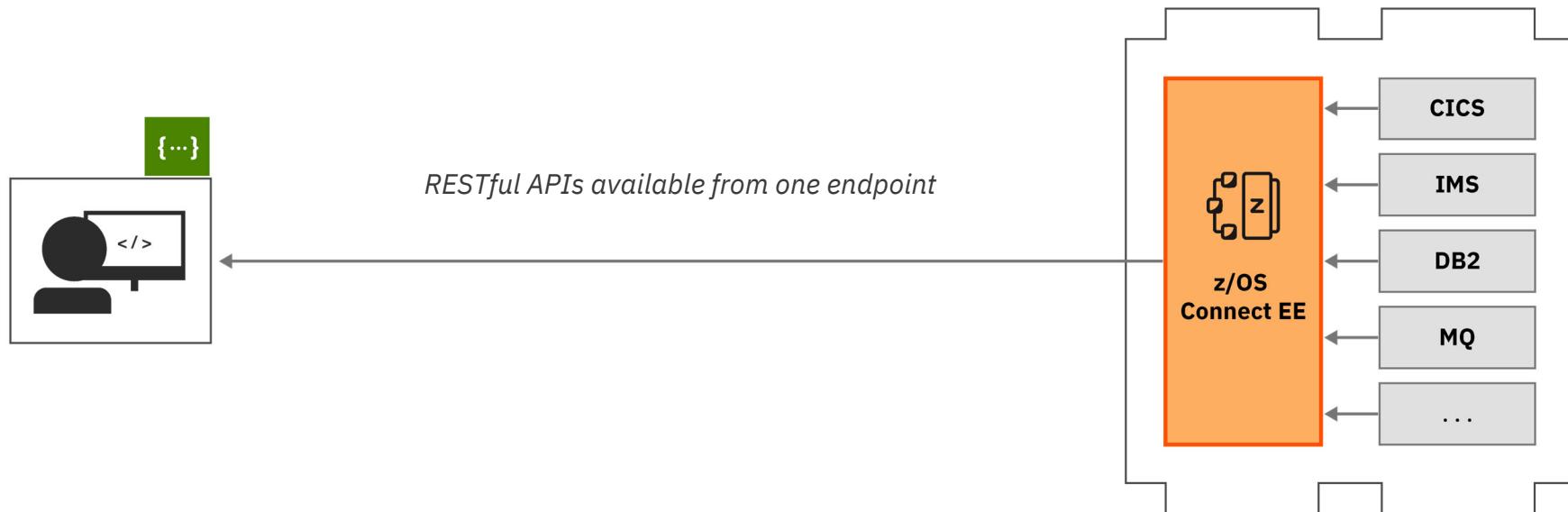


# z/OS Connect provides a single-entry point

- And exposes z/OS resources without writing any code.

z/OS Connect EE provides

- Single Configuration Administration
- Single Security Administration
- With sophisticated mapping of truly RESTful APIs to existing mainframe and services data without writing any code.



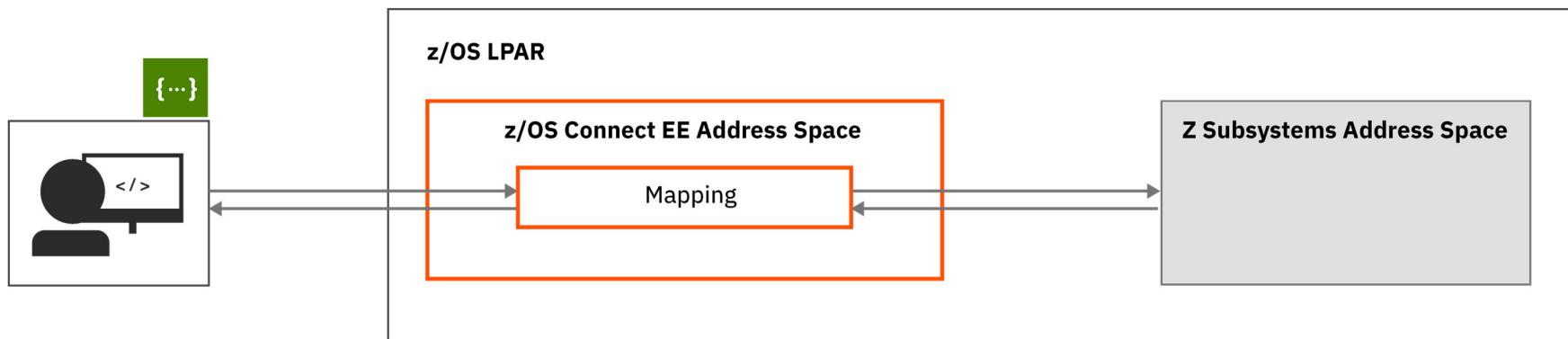


**Other than a RESTful interface,  
what else does z/OS Connect provide?**



# Data mapping/transformation

- Converting the JSON message to the format the target's subsystem expects\*.

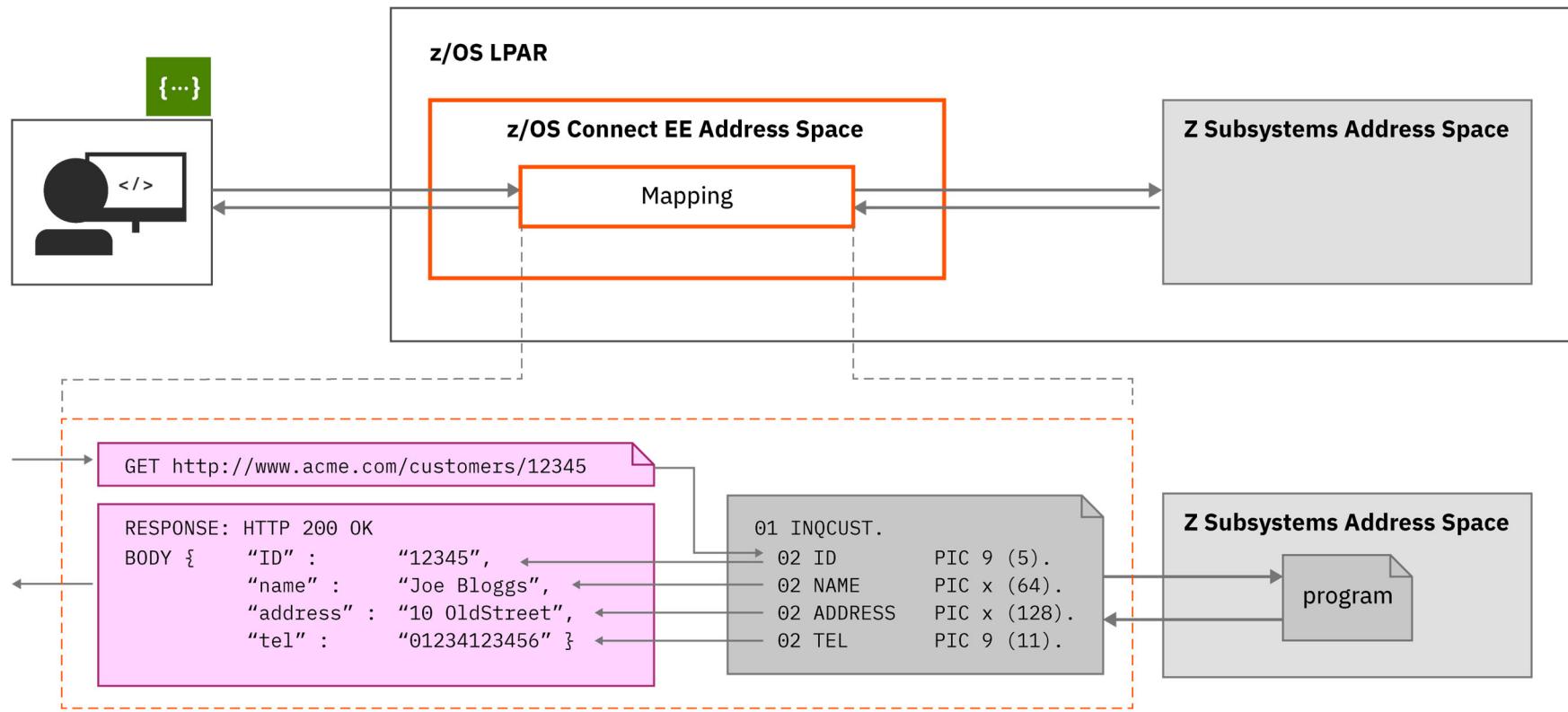


\* Most z/OS subsystems depend on information in a serial data format and do not normally work with JSON request/response messages. Examples of serialized messages are CICS COMMAREAAs and CONTAINERS, IMS or MQ messages, or records stored in sequential or VSAM data sets. Data mapping and transformation refers to the process of converting JSON messages to a serialized layout (e.g., sequentially arranged in storage).



# Data mapping and transformation example

- A closer look





## **z/OS Connect OpenAPI Tooling**

# zCEE - OpenAPI 2 Palette versus the OpenAPI 3 API Designer



z/OS Connect API Toolkit (Eclipse)

The screenshot shows the 'API Editor' window of the z/OS Connect API Toolkit. At the top, there are fields for 'Name' (new), 'Base path' (/new), and 'Version' (1.0.0). Below this, the 'Path' section contains a path entry /newPath1. Under 'Methods (4)', four operations are defined: POST, GET, PUT, and DELETE, each with associated service and mapping configurations.

z/OS Connect Designer (Designer Container)

The screenshot shows the 'IBM z/OS Connect Designer' interface. On the left, a tree view shows a project named 'cscvinc' with version 1.0.0. It includes sections for 'Paths', 'Components', and 'z/OS Assets'. The 'Paths' section shows two paths: '/employee' (POST) and '/employee/{employee}' (GET, PUT, DELETE). The right side of the screen displays the detailed configuration for these paths, including their summary, description, and available operations (POST, GET, PUT, DELETE).

The API toolkit is used to define the URI paths and methods.

The API specification provides predefined URI Paths and methods.

# An OPENAPI 2 versus an OPENAPI 3 Response Message



The screenshot shows the Eclipse-based z/OS Connect API Toolkit interface. A red circle highlights the 'Body - cscvincSelectServiceOperationResponse' section, which displays the response schema for the 'GET.employee.{employee}' operation. The schema includes fields like 'cscvincContainer', 'response', 'CEIBRESP', 'CEIBRESP2', 'USERID', and 'filea'. Below this, another red circle highlights the 'Body - cscvincSelectServiceOperationResponse' section in the 'IBM z/OS Connect Designer' window, showing the same response schema.

Eclipse based - z/OS Connect API Toolkit -  
The contents of the API's response message are  
directly derived from the z/OS asset's response

The screenshot shows the z/OS Connect Designer web browser interface. A red circle highlights the 'Body - cscvincSelectServiceOperationResponse' section in the 'IBM z/OS Connect Designer' window, displaying the response schema for the 'GET.employee.{employee}' operation. The schema includes fields like 'cscvincContainer', 'response', 'CEIBRESP', 'CEIBRESP2', 'USERID', and 'filea'. Below this, another red circle highlights the 'Body - cscvincSelectServiceOperationResponse' section in the '200 - OK' response details window, showing the same response schema.

Web browser - z/OS Connect Designer –  
The contents of the API's response  
message are derived from the z/O asset's  
response and JSONata coding, see URL  
<https://www.ibm.com/docs/en/zos-connect/zos-connect/3.0?topic=concepts-what-is-jsonata> for more information on  
JSONata.



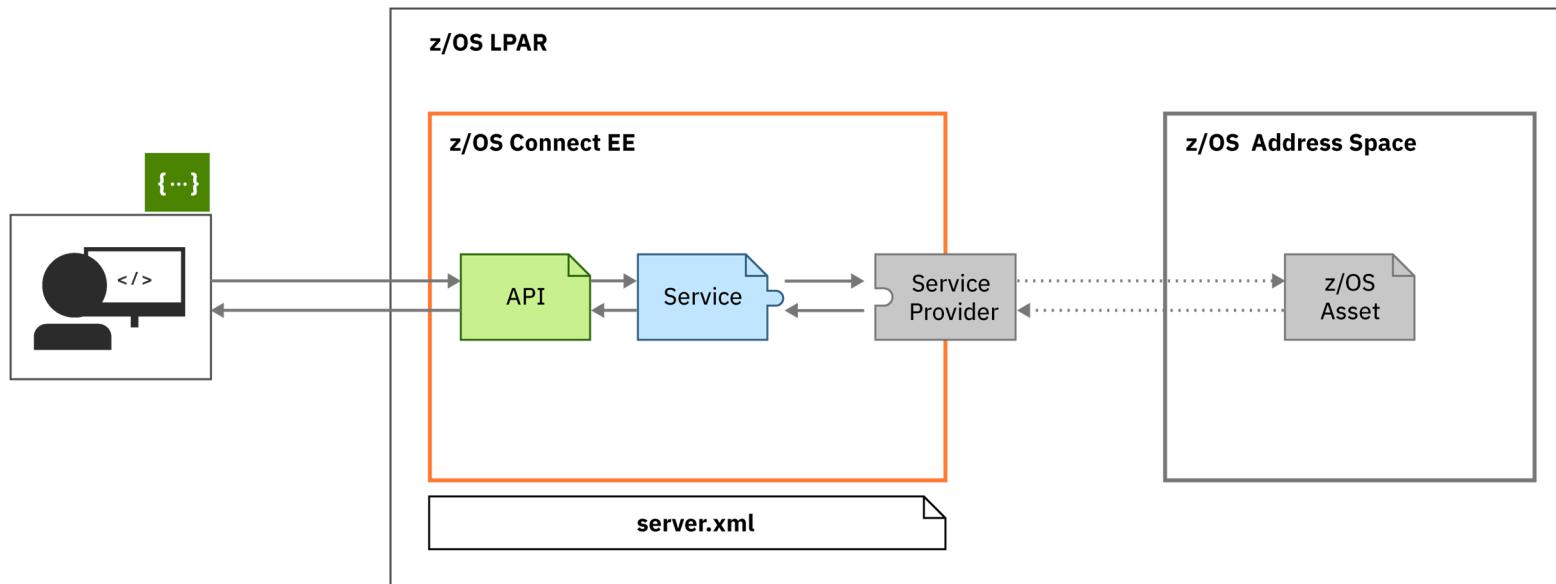
## OpenAPI 2

Simple **service creation** not using the Eclipse Toolkit



# Accessing a z/OS asset (Open API 2)

z/OS Connect OpenAPI 2 does not use a single monolithic interface – but rather separate plug-and-play components



- The API provides the RESTful interface is ready to be consumed by a client and it requires no knowledge that a z/OS resource is being accessed
- The Service provides meta data specific to the z/OS Asset (e.g., CICS program, MQ queue manager, etc.)
- The Service Provider is tightly coupled to a specific instance of a resource (e.g., host and port)

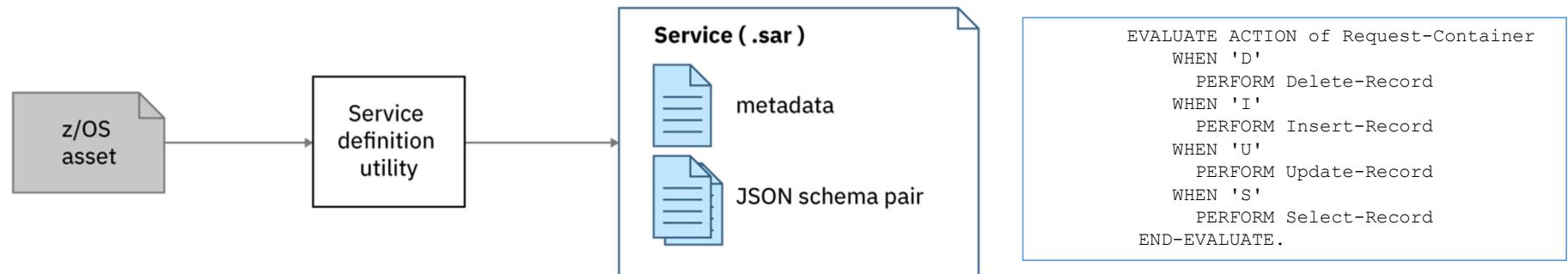


## Describing the interaction (service) with the z/OS resource (OpenAPI 2)

Start by creating a service to represent an interaction with the z/OS resource

To start mapping an API, z/OS Connect EE needs a representation of the underlying z/OS application: in a **Service Archive file (.sar)**.

The metadata consists of data mapping XML and provider specific configuration information



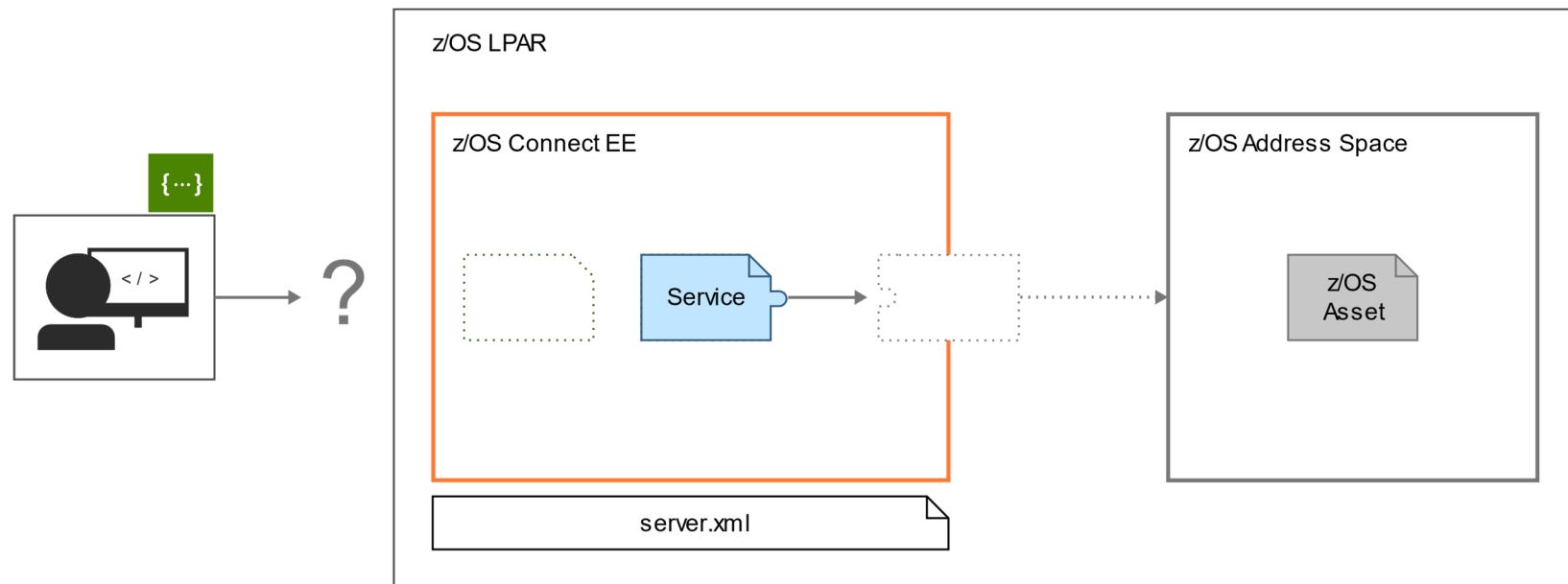
Use a system-appropriate utility to generate a service archive file for the z/OS application

- Eclipse based - API Toolkit (CICS, IMS TM, IMS DB, Db2 and MQ)
- Command line - z/OS Connect EE Build Toolkit (MVS Batch, IBM File Manager and HATS)
- Eclipse based - DVM Toolkit



# Deploy and export the service archive (OpenAPI 2)

Deploy and export the service archive file

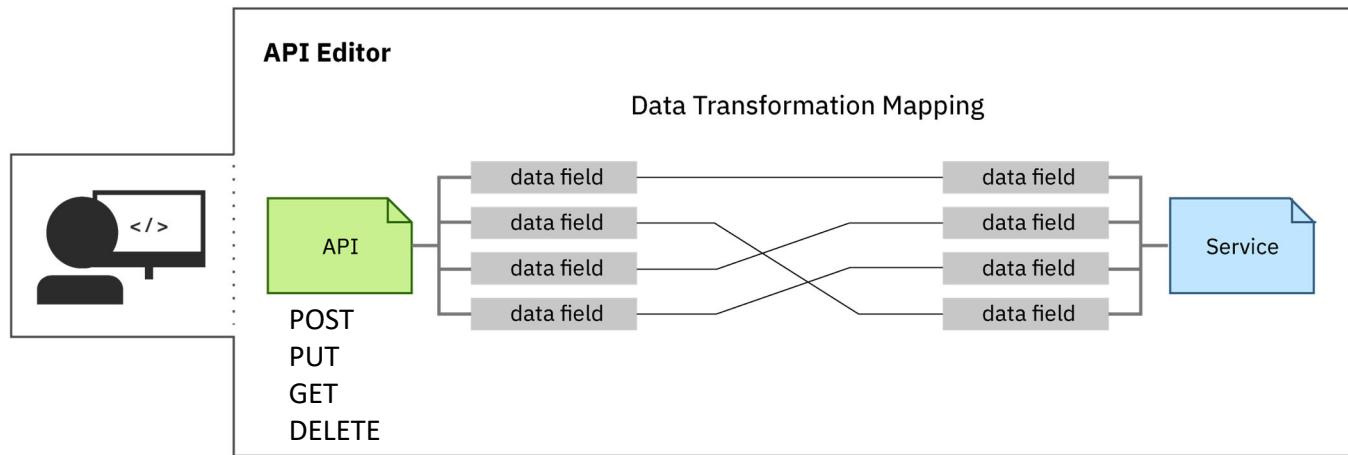


Deploy the service archive file generated in **Step 1** using the right-click deploy in **the API toolkit**.

# Develop an API using the service interactions (Open API 2)



Export the service and then import it to create an API that consumes the service

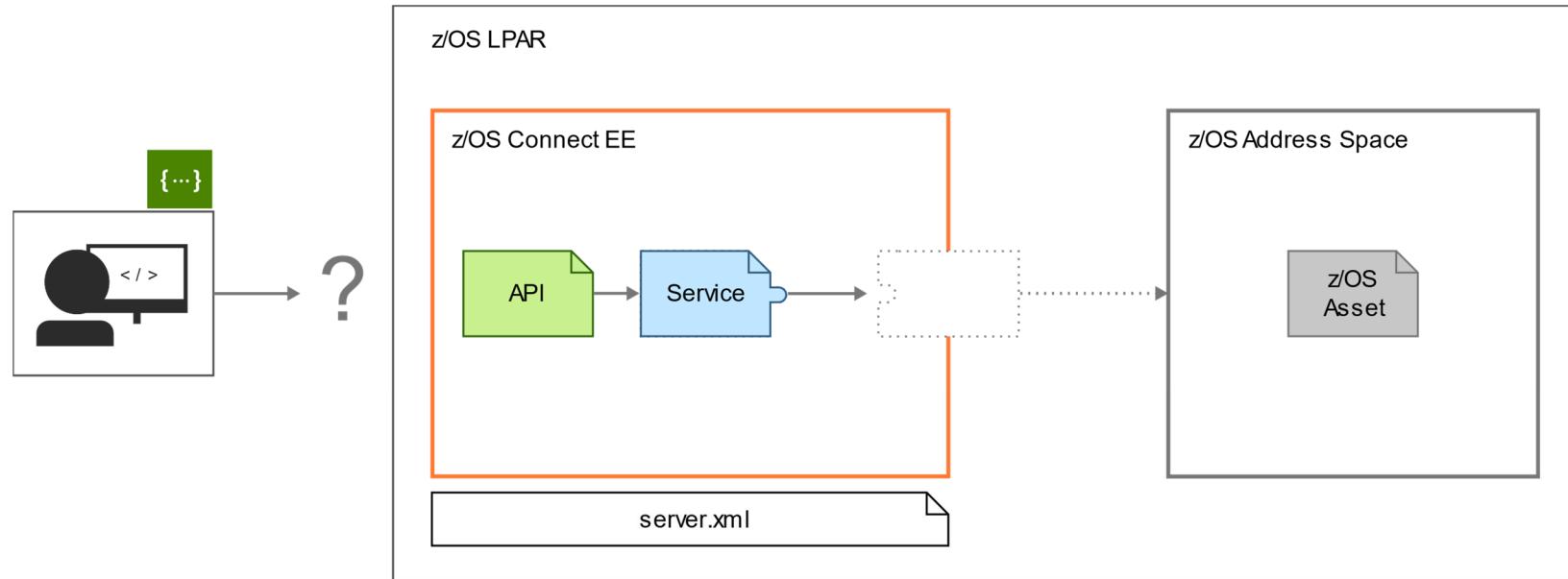


- Import the service archive file into the **API toolkit** and start designing the RESTful API.
- Provides additional data mapping
- Use the editor to describe the API and how it maps to underlying services.



# Deploy the API (Open API 2)

Deploy the API archive file

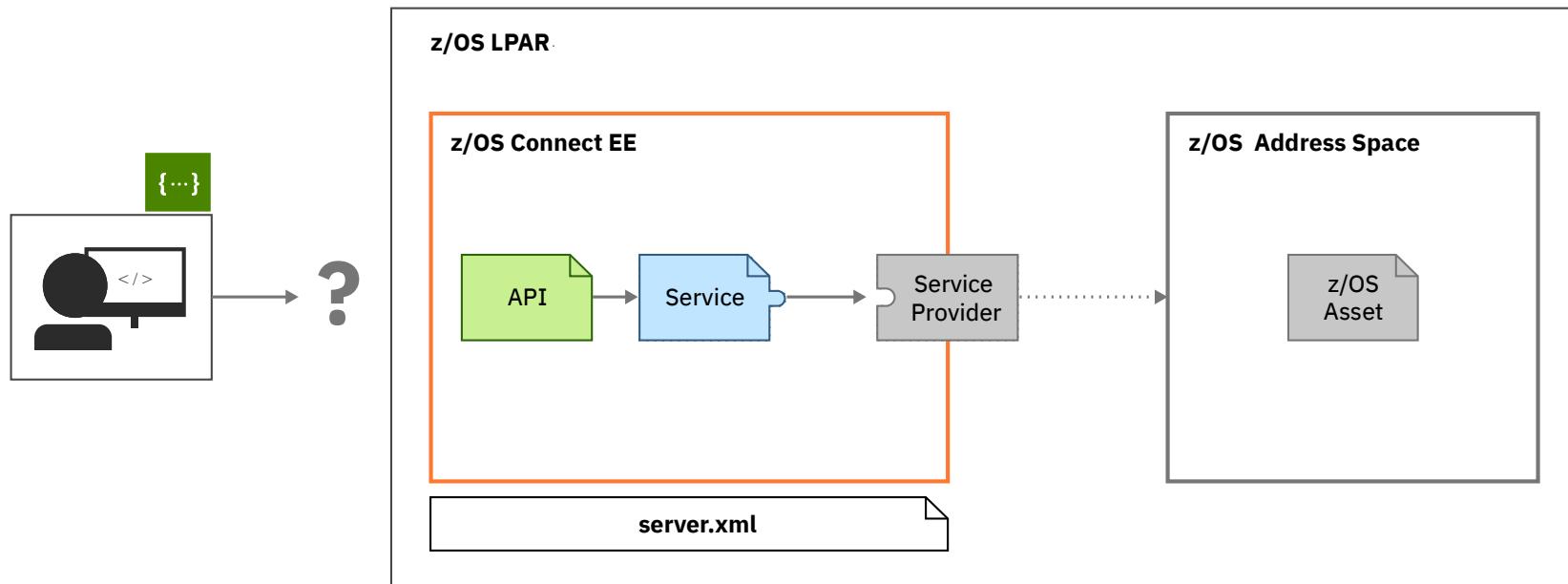


Deploy your API using the right-click deploy in **the API toolkit**



# Configure access the z/OS sub system (Open API 2)

Configure the service provider

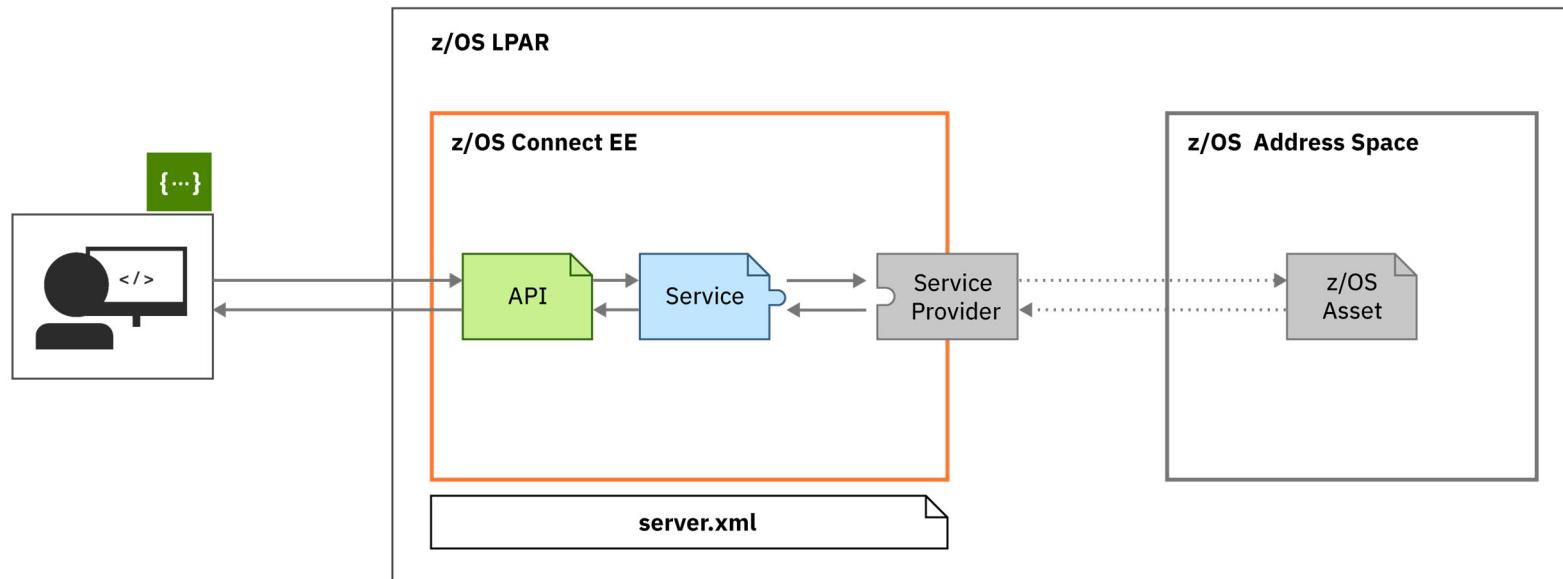


Configure the system-appropriate service provider to connect to your backend system in your **server.xml**.



# Complete access to a z/OS Asset (Open API 2)

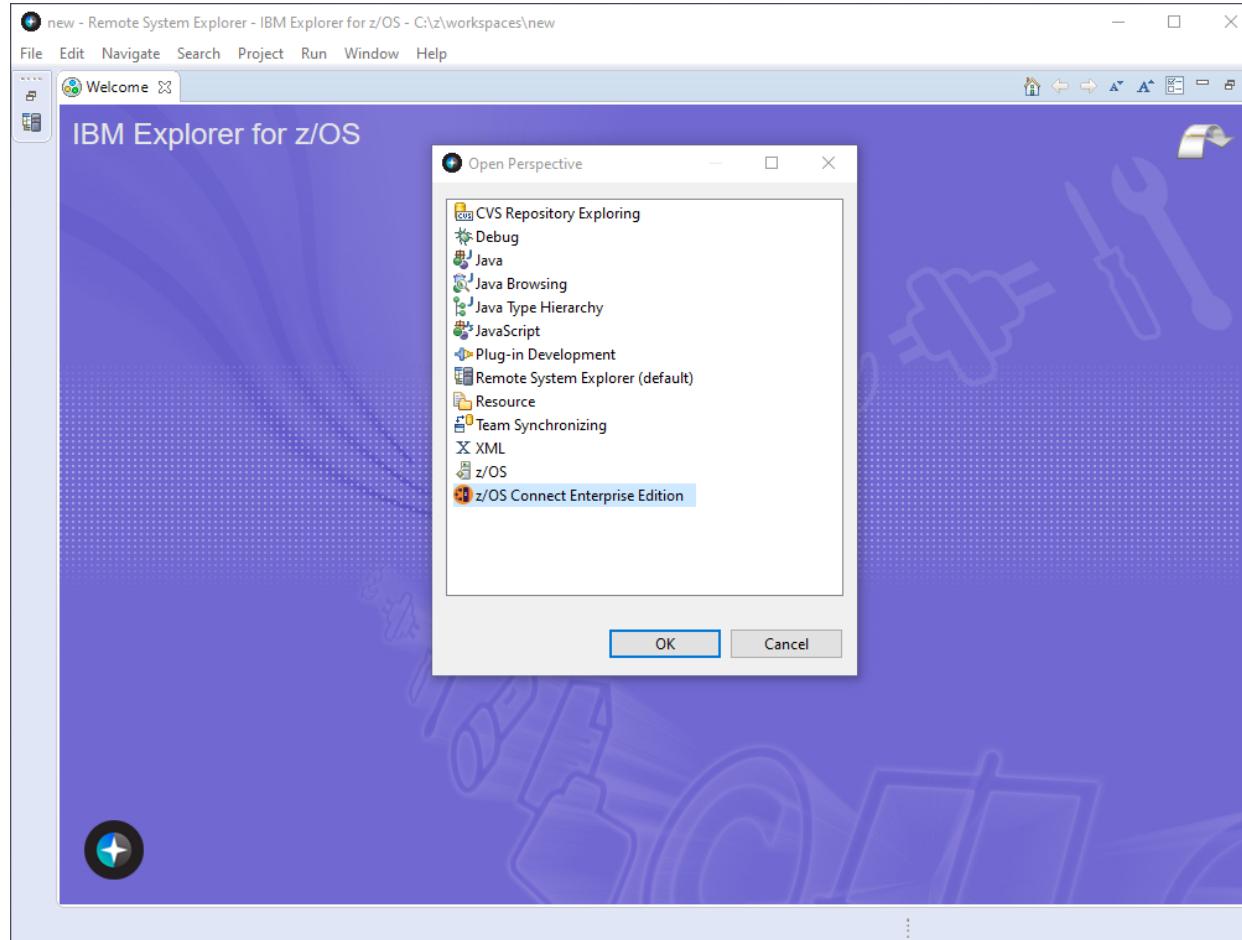
Done



- The API is ready to be consumed and requires no knowledge that a z/OS resource is being accessed
- The Service provides meta data specific to the z/OS Asset (e.g., CICS program, MQ queue manager, etc.)
- The Service Provider is tightly coupled to a specific instance of a resource (e.g., host and port, security)



## Eclipse API toolkit – Creating Services for CICS, IMS TM, IMS DB, Db2 and MQ

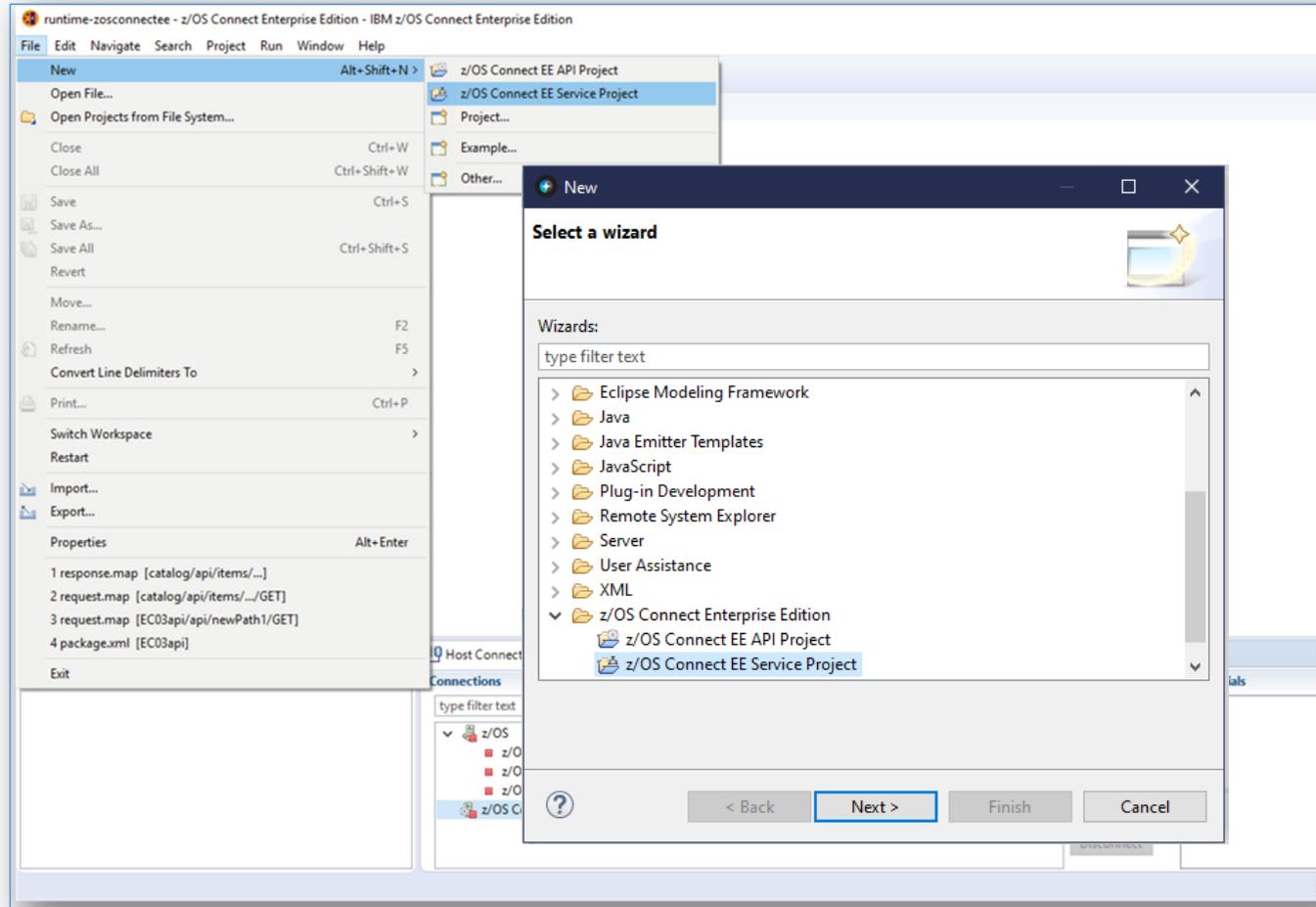


Use the **API toolkit** to create services through Eclipse-based tooling.

The API toolkit is available in the z/OS Connect Enterprise Edition Perspective in an Eclipse environment.



# API toolkit – Creating Services for CICS, IMS TM, IMS DB, Db2 and MQ



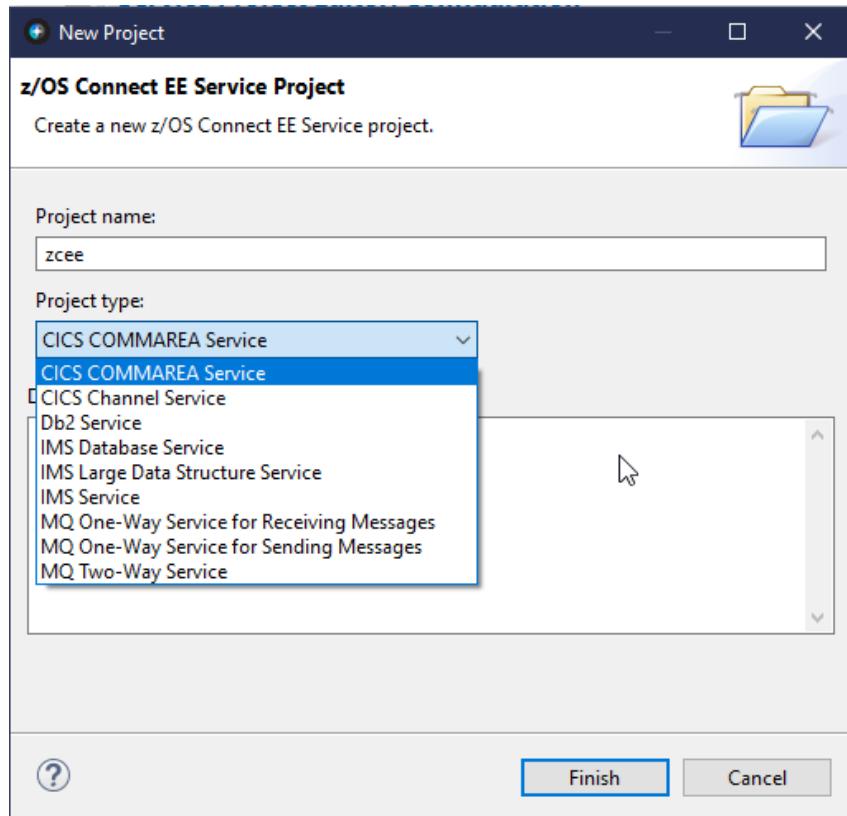
Use the **API toolkit** to create services through Eclipse-based tooling.

Services are described as Eclipse **Projects**, so they can be easily managed in source control.



# API toolkit – Creating Services for CICS, IMS TM, IMS DB, Db2 and MQ

## Service creation – a common interface



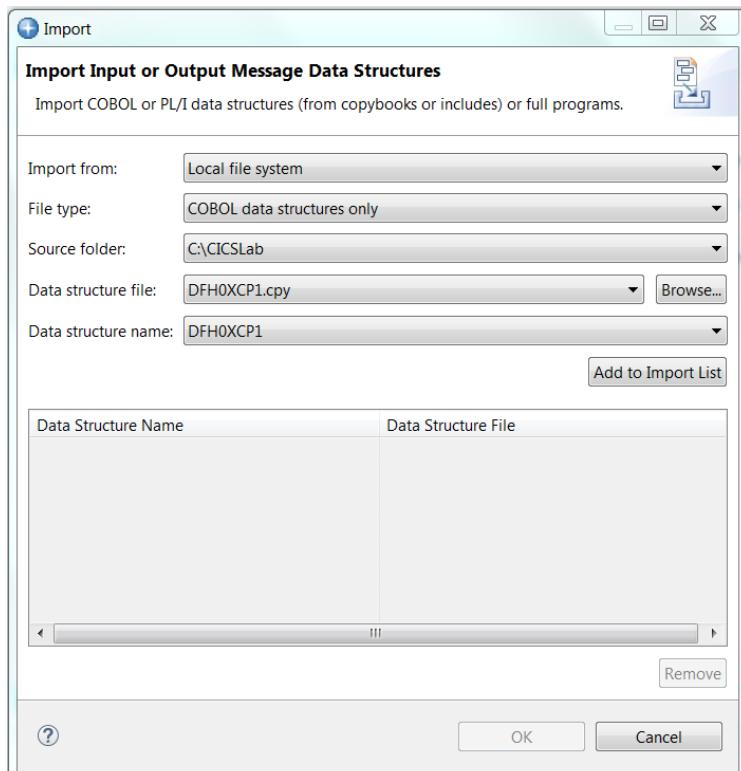
A common interface for service creation, irrespective of back-end subsystem.

- CICS services that can invoke almost any CICS programs accessed by EXEC CICS LINK request (COMMAREA or CHANNEL). See URL <http://www.ibm.com/docs/en/cics-ts/5.6?topic=link-exception-conditions-command> for a list of EXEC CICS APIs not allowed in a program when invoked using a CICS Dynamic Program Link request.
- Db2 services that invoke a Db2 REST service.
- IMS DB services that access an IMS database.
- IMS TM services that sends a messages on an IMS message processing region.
- MQ services that use MQ request/reply queues for two-way services or access a single queue for MQ PUTs and MQ GETs on a either a local or remote queue manager



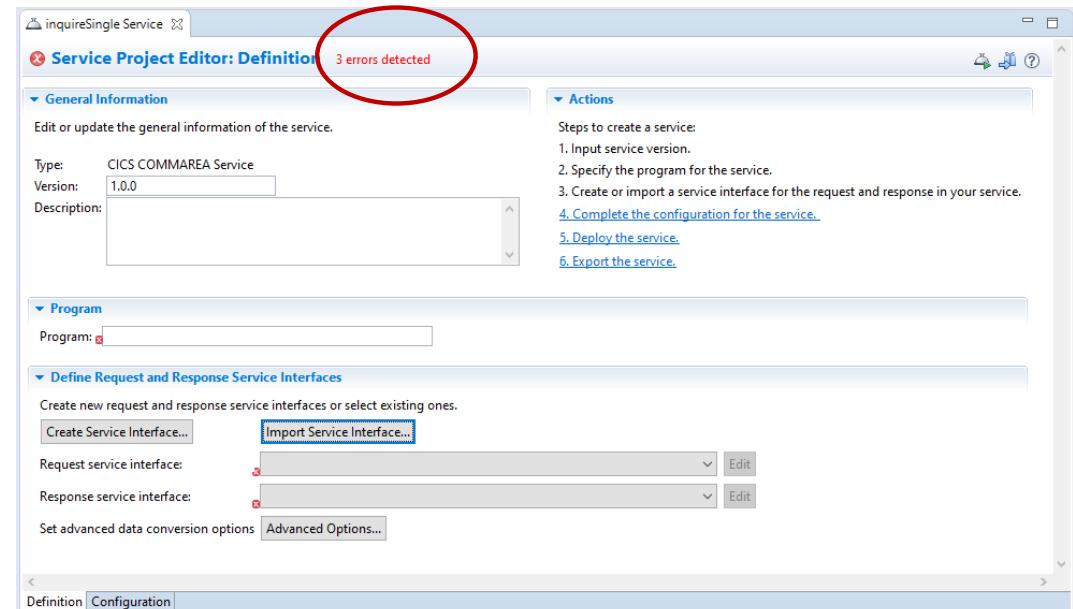
# API toolkit – Creating Services for CICS, IMS TM and MQ

## Creating a service project from source for a COMMAREA, Container or Message



Start by importing data structures into the service interface from the local file system or the workspace to create the request and response service interfaces.

The service interface supports complex data structures, including OCCURS DEPENDING ON and REDEFINES clauses.





# API toolkit – Creating Services for CICS, IMS TM and MQ

Allows editing a request service interface definition

```
*-----  
* Check which operation is being requested  
*-----  
* Uppercase the value passed in the Request Id field  
    MOVE FUNCTION UPPER-CASE(CA-REQUEST-ID) TO CA-REQUEST-ID  
    EVALUATE CA-REQUEST-ID  
        WHEN '01INQC'  
            Call routine to perform for inquire  
                PERFORM CATALOG-INQUIRE  
                WHEN '01INQS'  
            Call routine to perform for inquire for single item  
                PERFORM CATALOG-INQUIRE-SINGLE  
                WHEN '01ORDR'  
            Call routine to place order  
                PERFORM PLACE-ORDER  
                WHEN OTHER  
            Request is not recognised or supported  
                PERFORM REQUEST-NOT-RECOGNISED  
END-EVALUATE
```

See the imported data structure and then can **redact fields, rename fields, and add default values to fields** to make the service more consumable for an API developer.

The screenshot shows a software interface titled "Service Interface Definition". It displays a table of fields with columns: Fields, Include, Interface rename, Default Field Value, Data Type, Field Length, and Start Byte. The table lists various fields from a data structure, including CA\_REQUEST\_ID, CA\_RETURN\_CODE, CA\_RESPONSE\_MESSAGE, CA\_REQUEST\_SPECIFIC, CA\_INQUIRE\_REQUEST, CA\_INQUIRE\_SINGLE, CA\_ITEM\_REF\_REQ, CA\_SINGLE\_ITEM, and CA\_ORDER\_REQUEST. A red circle highlights the "Default Field Value" column for CA\_REQUEST\_ID, which is set to "01INQS". A red box highlights the "Interface rename" column for CA\_INQUIRE\_SINGLE, which is set to "inquireSingle". The "Include" column for CA\_INQUIRE\_SINGLE is checked.

| Fields   | Include                             | Interface rename    | Default Field Value | Data Type | Field Length | Start Byte |
|--|-------------------------------------|---------------------|---------------------|-----------|--------------|------------|
| COMMAREA                                       |                                     |                     |                     |           |              |            |
| DFHXCPI  |                                     |                     |                     |           |              |            |
| CA_REQUEST_ID                                  | <input type="checkbox"/>            | CA_REQUEST_ID       | 01INQS              | CHAR      | 6            | 1          |
| CA_RETURN_CODE                                 | <input type="checkbox"/>            | CA_RETURN_CODE      |                     | DECIMAL   | 2            | 7          |
| CA_RESPONSE_MESSAGE                            | <input type="checkbox"/>            | CA_RESPONSE_MESSAGE |                     | CHAR      | 79           | 9          |
| CA_REQUEST_SPECIFIC (Redefine)                 | <input type="checkbox"/>            | CA_REQUEST_SPECIFIC |                     | CHAR      | 911          | 88         |
| CA_INQUIRE_REQUEST redefine                    | <input type="checkbox"/>            | CA_INQUIRE_REQUEST  |                     | STRUCT    | 911          | 88         |
| CA_INQUIRE_SINGLE redefines CA_INQUIRE_REQUEST | <input checked="" type="checkbox"/> | inquireSingle       |                     | STRUCT    | 911          | 88         |
| CA_ITEM_REF_REQ                                | <input checked="" type="checkbox"/> | itemID              |                     | DECIMAL   | 4            | 88         |
| FILL_0   | <input type="checkbox"/>            | FILL_0              |                     | DECIMAL   | 4            | 92         |
| FILL_1   | <input type="checkbox"/>            | FILL_1              |                     | DECIMAL   | 3            | 96         |
| CA_SINGLE_ITEM                                 | <input type="checkbox"/>            | CA_SINGLE_ITEM      |                     | STRUCT    | 60           | 99         |
| FILL_2   | <input type="checkbox"/>            | FILL_2              |                     | CHAR      | 840          | 159        |
| CA_ORDER_REQUEST redefines CA_INQUIRE_REQUEST  | <input type="checkbox"/>            | CA_ORDER_REQUEST    |                     | STRUCT    | 911          | 88         |



# API toolkit – Creating Services for CICS, IMS TM, IMS DB and MQ

And editing a response message service interface definition

inquireSingleResponse

## 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   | <input type="checkbox"/>            |                     |                     |           |              |            |
| DFH0XCP1   | <input type="checkbox"/>            |                     |                     |           |              |            |
| CA_REQUEST_ID                                      | <input checked="" type="checkbox"/> | CA_REQUEST_ID       |                     | CHAR      | 6            | 1          |
| CA_RETURN_CODE                                     | <input checked="" type="checkbox"/> | returnCode          |                     | DECIMAL   | 2            | 7          |
| CA_RESPONSE_MESSAGE                                | <input checked="" type="checkbox"/> | responseMessage     |                     | CHAR      | 79           | 9          |
| CA_REQUEST_SPECIFIC (Redefines CA_INQUIRE_REQUEST) | <input type="checkbox"/>            | CA_REQUEST_SPECIFIC |                     | CHAR      | 911          | 88         |
| CA_INQUIRE_REQUEST redefines CA_INQUIRE_SINGLE     | <input type="checkbox"/>            | CA_INQUIRE_REQUEST  |                     | STRUCT    | 911          | 88         |
| CA_INQUIRE_SINGLE redefines CA_ITEM_REF_REQ        | <input checked="" type="checkbox"/> | inquireSingle       |                     | STRUCT    | 911          | 88         |
| CA_ITEM_REF_REQ                                    | <input type="checkbox"/>            | CA_ITEM_REF_REQ     |                     | DECIMAL   | 4            | 88         |
| FILL_0   | <input type="checkbox"/>            | FILL_0              |                     | DECIMAL   | 4            | 92         |
| FILL_1   | <input type="checkbox"/>            | FILL_1              |                     | DECIMAL   | 3            | 96         |
| CA_SINGLE_ITEM                                     | <input checked="" type="checkbox"/> | singleItem          |                     | STRUCT    | 60           | 99         |
| CA_SNGL_ITEM_REF                                   | <input checked="" type="checkbox"/> | itemReference       |                     | DECIMAL   | 4            | 99         |
| CA_SNGL_DESCRIPTION                                | <input checked="" type="checkbox"/> | description         |                     | CHAR      | 40           | 103        |
| CA_SNGL_DEPARTMENT                                 | <input checked="" type="checkbox"/> | department          |                     | DECIMAL   | 3            | 143        |
| CA_SNGL_COST                                       | <input checked="" type="checkbox"/> | cost                |                     | CHAR      | 6            | 146        |
| IN_SNGL_STOCK                                      | <input checked="" type="checkbox"/> | inStock             |                     | DECIMAL   | 4            | 152        |
| ON_SNGL_ORDER                                      | <input checked="" type="checkbox"/> | onOrder             |                     | DECIMAL   | 3            | 156        |
| FILL_2   | <input type="checkbox"/>            | FILL_2              |                     | CHAR      | 840          | 159        |
| CA_ORDER_REQUEST redefines CA_USERID               | <input type="checkbox"/>            | CA_ORDER_REQUEST    |                     | STRUCT    | 911          | 88         |
| CA_USERID  | <input type="checkbox"/>            | CA_USERID           |                     | CHAR      | 8            | 88         |
| CA_CHARGE_DEPT                                     | <input type="checkbox"/>            | CA_CHARGE_DEPT      |                     | CHAR      | 8            | 96         |
| CA_ITEM_REF_NUMBER                                 | <input type="checkbox"/>            | CA_ITEM_REF_NUMBER  |                     | DECIMAL   | 4            | 104        |
| CA_QUANTITY_REQ                                    | <input type="checkbox"/>            | CA_QUANTITY_REQ     |                     | DECIMAL   | 3            | 108        |
| FILL_3   | <input type="checkbox"/>            | FILL_3              |                     | CHAR      | 888          | 111        |

See the imported data structure and can **redact fields** and **rename fields**



# API toolkit – Creating Services for CICS

Creating multiple services definitions to the same resource

| Fields       | Include                             | Interface Rename  | Default Field Value | Data Type | Field Length |
|--------------|-------------------------------------|-------------------|---------------------|-----------|--------------|
| Channel      |                                     | CSCCINCContainer  |                     |           |              |
| @ Container1 |                                     | REQUEST_CONTAINER |                     |           |              |
| ACTION       | <input type="checkbox"/>            | ACTION            | S                   | CHAR      | 1            |
| USERID       | <input type="checkbox"/>            | USERID            |                     | CHAR      | 8            |
| FILEA_AREA   | <input checked="" type="checkbox"/> | FILEA_AREA        |                     | STRUCT    | 80           |
| STAT         | <input type="checkbox"/>            | STAT              |                     | CHAR      | 1            |
| NUMB         | <input checked="" type="checkbox"/> | NUMB              |                     | CHAR      | 6            |
| NAME         | <input type="checkbox"/>            | NAME              |                     | CHAR      | 20           |
| ADDRX        | <input type="checkbox"/>            | ADDRX             |                     | CHAR      | 20           |
| PHONE        | <input type="checkbox"/>            | PHONE             |                     | CHAR      | 8            |
| DATEX        | <input type="checkbox"/>            | DATEX             |                     | CHAR      | 8            |
| AMOUNT       | <input type="checkbox"/>            | AMOUNT            |                     | CHAR      | 8            |
| COMMENT      | <input type="checkbox"/>            | COMMENT           |                     | CHAR      | 9            |

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.

| Fields       | Include                             | Interface Rename       | Default Field Value | Data Type | Field Length |
|--------------|-------------------------------------|------------------------|---------------------|-----------|--------------|
| Channel      |                                     | cscvincInsertContainer |                     |           |              |
| @ Container1 |                                     | REQUEST_CONTAINER      |                     |           |              |
| ACTION       | <input type="checkbox"/>            | ACTION                 | I                   | CHAR      | 1            |
| USERID       | <input type="checkbox"/>            | USERID                 |                     | CHAR      | 8            |
| FILEA_AREA   | <input checked="" type="checkbox"/> | FILEA_AREA             |                     | STRUCT    | 80           |
| STAT         | <input checked="" type="checkbox"/> | status                 |                     | CHAR      | 1            |
| NUMB         | <input checked="" type="checkbox"/> | employeeNumber         |                     | CHAR      | 6            |
| NAME         | <input checked="" type="checkbox"/> | employeeName           |                     | CHAR      | 20           |
| ADDRX        | <input checked="" type="checkbox"/> | address                |                     | CHAR      | 20           |
| PHONE        | <input checked="" type="checkbox"/> | phoneNumber            |                     | CHAR      | 8            |
| DATEX        | <input checked="" type="checkbox"/> | startDate              |                     | CHAR      | 8            |
| AMOUNT       | <input checked="" type="checkbox"/> | amount                 |                     | CHAR      | 8            |
| COMMENT      | <input checked="" type="checkbox"/> | comment                |                     | CHAR      | 9            |

mitchj@us.ibm.com

The service developer creates distinct services for each function by setting the ACTION field to S for select, I for insert, U for update or D for delete

General Information

Type: CICS Channel Service  
Version: 1.0.0  
Description: CSCVINC

Actions

1. Input service version.
2. Specify the program for the service.
3. Create or import a service interface for the request and response in your service.
4. Complete the configuration for the service.
5. Deploy the service.
6. Export the service.

Program: CSCVINC

Define Request and Response Service Interfaces

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

Request service interface: cscvincSelectRequest.si  
Response service interface: cscvincSelectResponse.si

```
EVALUATE ACTION of Request-Container
WHEN 'D'
  PERFORM Delete-Record
WHEN 'I'
  PERFORM Insert-Record
WHEN 'U'
  PERFORM Update-Record
WHEN 'S'
  PERFORM Select-Record
END-EVALUATE.
```



# Accessing a CICS program – Transaction ID Usage

**cscvincSelectService Service**

**Service Project Editor: Configuration**

**Required Configuration**

Enter the required configuration for this service.

Coded character set identifier (CCSID):

Connection reference:

**Optional Configuration**

Enter the optional configuration for this service.

Transaction ID:

Transaction ID usage:

Bidi configuration reference:

Use context containers:

Context containers HTTP headers:

Add another

**Definition** **Configuration**

**EIB\_ONLY**

```

TRANSACTION: CSMI PROGRAM: DFHMIRS TASK: 0008501 APPLID: CICS53Z DISPLAY: 00
STATUS: PROGRAM INITIATION
EIBTIME = 104730
EIBDATE = 0122050
EIBTRNID = 'CSMI'
EIBTRSKN = 8501
EIBTRMID = '/RBX'

EIBCPOSN = 0
EIBCALEN = 0
EIBAID = X'00'
EIBFN = X'0000'
EIBRCODE = X'000000000000
EIBDS = '.....'
+ EIBREQID = '.....'

ENTER: CONTINUE PF1 : UNDEFINED PF2 :
PF4 : SUPPRESS DISPLAYS PF5 :
PF7 : SCROLL BACK PF8 :
PF10: PREVIOUS DISPLAY PF11 :
M1 D
Offset:X'001CD6' Line:
Connected to remote server/host wg31a using lu/pool TCP00126 and port 23

```

**EIB\_AND\_MIRROR**

```

TRANSACTION: MJO PROGRAM: DFHMIRS DFH
STATUS: PROGRAM INITIATION
EIBTIME = 109914
EIBDATE = 0122050
EIBTRNID = 'MJO'
EIBTRSKN = 8492
EIBTRMID = '/RBX'

EIBCPOSN = 0
EIBCALEN = 0
EIBAID = X'00'
EIBFN = X'0000'
EIBRCODE = X'000000000000
EIBDS = '.....'
+ EIBREQID = '.....'

ENTER: CONTINUE PF1 : UNDEFINED PF2 :
PF4 : SUPPRESS DISPLAYS PF5 :
PF7 : SCROLL BACK PF8 :
PF10: PREVIOUS DISPLAY PF11 :
M1 D
Offset:X'001CD6' Line:
Connected to remote server/host wg31a using lu/pool TCP00126 and port 23

```

**WG31 - 3270**

```

TRANSACTION: MJO PROGRAM: DFH
STATUS: PROGRAM INITIATION
EIBTIME = 181730
EIBDATE = 0122051
EIBTRNID = 'MJO'
EIBTRSKN = 8837
EIBTRMID = '/RBX'

EIBCPOSN = 0
EIBCALEN = 0
EIBAID = X'00'
EIBFN = X'0E02' LINK
EIBRCODE = X'000000000000
EIBDS = '.....'
+ EIBREQID = '.....'

ENTER: CONTINUE PF1 : UNDEFINED PF2 : SWITCH HEX/CHAR PF3 : END EDF
PF4 : SUPPRESS DISPLAYS PF5 : WORKING STORAGE PF6 : USER DIS
PF7 : SCROLL BACK PF8 : SCROLL FORWARD PF9 : STOP CON
PF10: PREVIOUS DISPLAY PF11 : EIB DISPLAY PF12: UNDEFIN
M1 D
Offset:X'001CD6' Line:
Connected to remote server/host wg31a using lu/pool TCP00126 and port 23

```

- Transaction ID** attaches a CICS transaction (CSMI is the default) that starts the CICS DFHMIRS program.
- Transaction ID Usage** attribute useful for:
  - Transaction security requirements
  - Db2 plan selection
  - Transaction classification and reporting

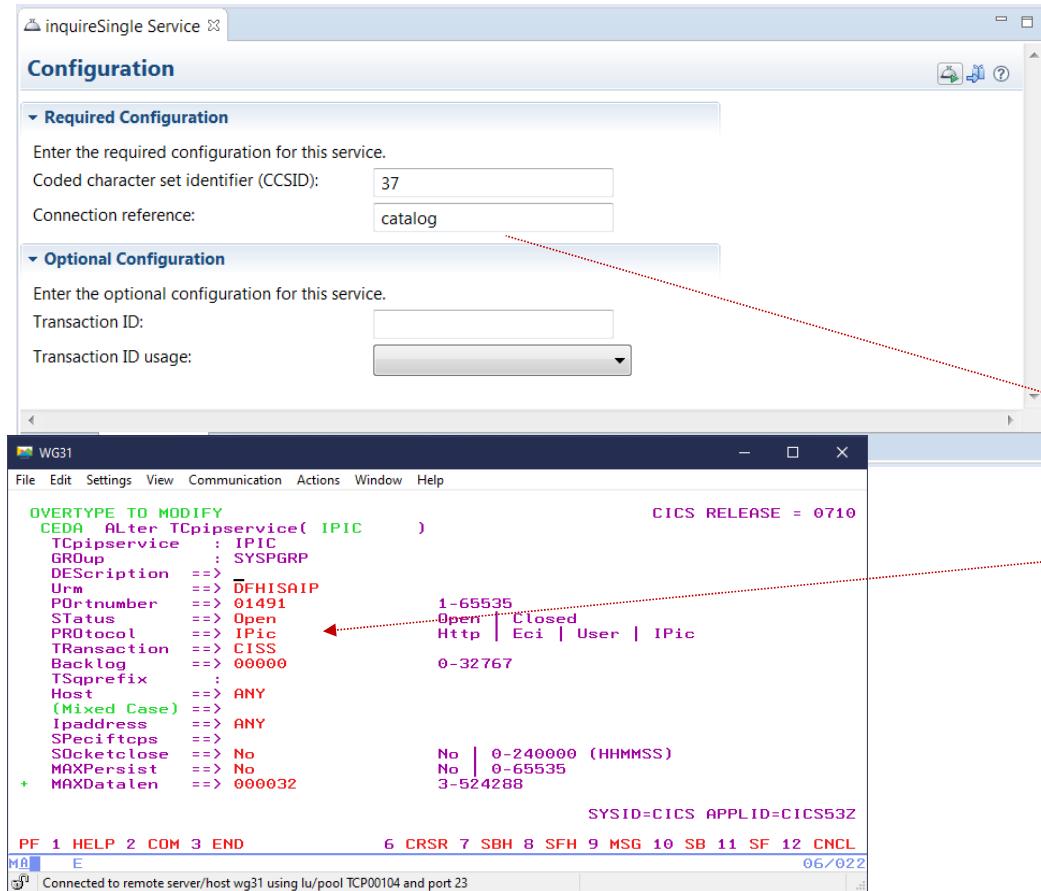
mitchj@us.ibm.com

These attributes also be used in `zosconnect_cicsIpicConnection` and `zosconnect_services>service configuration` elements.

# Accessing a CICS program uses CICS IP interconnectivity (IPIC)



The server.xml file is the key configuration file:



Features are functional building blocks. When configured here, that function becomes available to the Liberty server

```
<server description="CICS IPIC - catalog">
<!-- Enable features -->
<featureManager>
<feature>zosconnect:cicsService-1.0</feature>
</featureManager>
<zosconnect_cicsIpicConnection id="catalog">
<host>wg31.washington.ibm.com</host>
<port>1491</port>
<transid>CSMI</transid>
<transidUsage>EIB_AND_MIRROR</transidUsage>
</zosconnect_cicsIpicConnection>
</server>
```

Define IPIC connection to CICS



# API toolkit – Creating Services for IMS

## Creating a “GET” service interface request definition

```
*-----*
*      ROUTE TO REQUEST HANDLER
*-----*
    SPACE 1
    CLC KADD,IOCMD    IF COMMAND ADD ENTERED ?
    BE  TOADD     ...THEN, GOTO INSERT ENTRY
    CLC KUPD,IOCMD    IF COMMAND UPDATE ENTERED ?
    BE  TOUPD     ...THEN, GOTO UPDATE ENTRY
    CLC KDEL,IOCMD    IF COMMAND DEL ENTERED ?
    BE  TODEL     ...THEN, GOTO DELETE ENTRY
    CLC KDIS,IOCMD    IF COMMAND DIS ENTERED ?
    BE  TODIS     ...THEN, GOTO DISPLAY ENTRY
    CLC KTAD,IOCMD    IF TEST ADD WITH REPLY ?
    BE  TOTAD     ...THEN,
    B   INVREQ1    INVALID REQUEST
```

**ivtnoDisplayRequest**

**Service Interface Editor**

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.

| Fields              | Include                             | Interface Rename | Default Field Value | Data Type | Field Length |
|---------------------|-------------------------------------|------------------|---------------------|-----------|--------------|
| ivtnoDisplayRequest |                                     |                  |                     |           |              |
| Segment 1           |                                     |                  |                     |           |              |
| INPUT_MSG           |                                     | phonebookRequest |                     |           |              |
| IN_LL               | <input type="checkbox"/>            | IN_LL            |                     | SHORT     | 2            |
| IN_ZZ               | <input type="checkbox"/>            | IN_ZZ            |                     | SHORT     | 2            |
| IN_TRANCODE         | <input type="checkbox"/>            | IN_TRANCODE      |                     | CHAR      | 10           |
| IN_COMMAND          | <input checked="" type="checkbox"/> | IN_COMMAND       | IVTNO<br>DISPLAY    | CHAR      | 8            |
| IN_LAST_NAME        | <input type="checkbox"/>            | lastName         |                     | CHAR      | 10           |
| IN_FIRST_NAME       | <input type="checkbox"/>            | IN_FIRST_NAME    |                     | CHAR      | 10           |
| IN_EXTENSION        | <input type="checkbox"/>            | IN_EXTENSION     |                     | CHAR      | 10           |
| IN_ZIP_CODE         | <input type="checkbox"/>            | IN_ZIP_CODE      |                     | CHAR      | 7            |

The service developer creates distinct services for each function.

DISPLAY (GET)  
DELETE (DELETE)  
ADD (POST)  
UPDATE (PUT)

**ivtnoDisplayService Service**

**Service Project Editor: Configuration**

**Required Configuration**

Enter the required configuration for this service.

Connection profile: **IMSCONN**

Interaction profile: **IMSINTER** (circled in red)

**Optional Configuration**

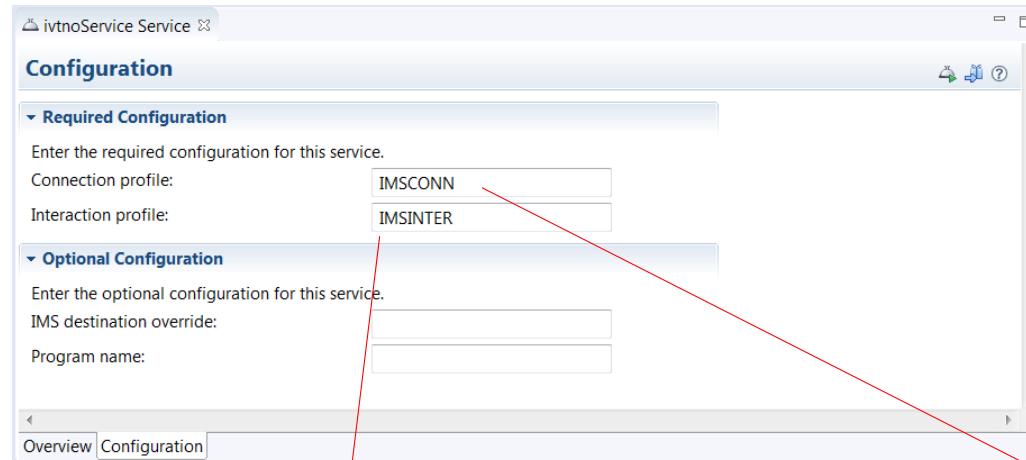
Enter the optional configuration for this service.

IMS destination override:

Program name:



# IMS Connections and Interactions



Connection Profile

```
<server>
<imsmobile_imsConnection comment="" connectionFactoryRef="CF1" connectionTimeout="-1" connectionType="IMSCONNECT" id="IMSCONN"/>
<connectionFactory containerAuthDataRef="Connection1_Auth" id="CF1">
    <properties.gmoa hostName="wg31.washington.ibm.com" portNumber="4000"/>
</connectionFactory>

<authData id="Connection1_Auth" password="encryptedPassword1" user="userName1"/>
</server>
```

Interaction Profile

```
<server>
<imsmobile_interaction comment="" commitMode="1" id="IMSINTER" imsConnectCodepage="Cp1047" imsConnectTimeout="0"
    imsDatastoreName="IVP1" interactionTimeout="-1" ltermOverrideName="" syncLevel="0"/>
</server>
```

IMS Connect HWSCFG

```
HWS=(ID=IMS14HWS,XIBAREA=100,RACF=Y,RRS=N)
TCPIP=(HOSTNAME=TCPIP,PORTID=(4000,LOCAL),RACFID=JOHNSON,TIMEOUT=
5000)
DATASTORE=(GROUP=OTMAGRP, ID=IVP1, MEMBER=HWSMEM, TMEMBER=OTMAMEM)
```

mitchj@us.ibm.com



# API toolkit – Creating Services for IMS DB

## Creating a service project from the IMS Catalog

The screenshot shows the 'Service Project Editor: Definition' window for an 'IMS Database Service'. The 'General Information' section shows 'Type: IMS Database Service', 'Version: 1.0.0', and a 'Description' field. The 'Actions' section lists steps: 1. Input service version, 2. Specify the SQL command for the service, 3. Specify Database Connection Properties and Generate Service Interface, 4. Complete the configuration for the service, 5. Deploy the service, and 6. Export the service. The 'Enter or import SQL Command' section contains the SQL query: 'SELECT FIRSTNAME, ZIPCODE, PHONENBR, A1111111 FROM ATSVPA.A1111111 WHERE A1111111=?'. This query is circled in red. Below it, the 'Specify Database Connection Properties and Generate Service Interface' section shows 'Database Connection: wg31:5555' and 'Database Name: DFSIVPA'. There is also a 'Generate Service Interface...' button.

Use the IMS Catalog to assist with developing and testing SQL SELECT commands used for accessing IMS databases.

```
*-----  
* SEGMENT DESCRIPTION *  
* ROOT ONLY DATABASE *  
* BYTES 1-10 LAST NAME (CHARACTER) - KEY *  
* BYTES 11-20 FIRST NAME (CHARACTER) *  
* BYTES 21-30 INTERNAL PHONE NUMBER (NUMERIC) *  
* BYTES 31-37 INTERNAL ZIP (CHARACTER) *  
* BYTES 38-40 RESERVED *  
*-----  
DBD NAME=IVPDB1,ACCESS=(HIDAM,OSAM)  
DATASET DD1=DFSIVD1,DEVICE=3380,SIZE=2048  
SEGM NAME=A1111111,PARENT=0,BYTES=40,RULES=(LLV,LAST),  
PTR=(TB,CTR)  
FIELD NAME=(A1111111,SEQ,U),BYTES=010,START=00001,TYPE=C  
FIELD NAME=FIRSTNAME,BYTES=010,START=00011,TYPE=C  
FIELD NAME=PHONENBR,BYTES=010,START=00021,TYPE=C  
FIELD NAME=ZIPCODE,BYTES=7,START=00031,TYPE=C  
LCHILD NAME=(A1,IVPDB1I),POINTER=INDX,RULES=LAST  
DBDGEN  
FINISH  
END
```



## API toolkit – Creating Services for IMS DB

The Toolkit allows editing a service interface definitions\*

The screenshot shows the Service Interface Editor window. The left pane displays a tree view of service interfaces and segments, with 'selectByName\_Response' expanded to show 'Segment 1' and 'Output Columns'. Under 'Output Columns', 'Result [0..\*]' is selected, showing five fields: FIRSTNAME, ZIPCODE, PHONENBR, and A1111111. The right pane is a table with columns: Fields, Include, Interface Rename, Default Field Value, Data Type, and Field Length. The table rows correspond to the selected output columns. A red circle highlights the checkbox for 'INCLUDE' next to 'result'. A red box highlights the 'Interface Rename' column for the 'result' row, which contains 'response'. The other rows have 'INCLUDE' checked and 'Interface Rename' empty.

| Fields    | Include                             | Interface Rename | Default Field Value | Data Type | Field Length |
|-----------|-------------------------------------|------------------|---------------------|-----------|--------------|
| FIRSTNAME | <input checked="" type="checkbox"/> |                  |                     | CHAR      | 10           |
| ZIPCODE   | <input checked="" type="checkbox"/> |                  |                     | CHAR      | 7            |
| PHONENBR  | <input checked="" type="checkbox"/> |                  |                     | CHAR      | 10           |
| A1111111  | <input checked="" type="checkbox"/> |                  |                     | CHAR      | 10           |
|           | <input checked="" type="checkbox"/> | response         |                     |           |              |
|           | <input checked="" type="checkbox"/> | result           |                     | ARRAY     | 0            |
|           | <input checked="" type="checkbox"/> | firstName        |                     | CHAR      | 10           |
|           | <input checked="" type="checkbox"/> | zipCode          |                     | CHAR      | 7            |
|           | <input checked="" type="checkbox"/> | phoneNuber       |                     | CHAR      | 10           |
|           | <input checked="" type="checkbox"/> | lastName         |                     | CHAR      | 10           |

\*Using a slightly different process



# IMS Connection Factory in the server XML

Service Project Editor: Configuration

Required Configuration

Enter the required configuration for this service.

Connection profile:

## ConnectionFactory

```
<connectionFactory id="DFSIVPACConn">
<properties.imsudbJLocal
    databaseName="DFSIVPA"
    datastoreName="IVP1"
    datastoreServer="wg31.washington.ibm.com"
    driverType="4"
    portNumber="5555"
    user="USER1"
    password="password"
    flattenTables="True"/>
</connectionFactory>
```

## IMS Connect HWSCFG

```
HWS=(ID=IMS14HWS,XIBAREA=100,RACE=N,RRS=N)
TCPIP=(HOSTNAME=TCPIP,PORTID=(4000,LOCAL),RACFID=JOHNSON,TIMEOUT=5000)
DATASTORE=(GROUP=OTMAGRP,ID=IVP1, MEMBER=HWSMEM, TMEMBER=OTMAMEM)
IMSPLEX=(MEMBER=IMS14HWS, TMEMBER=PLEX1)
ODACCESS=(ODBMAUTOCCONN=Y,
DRDAPORT=(ID=5555,PORTTMOT=6000), ODBMTMOT=6000)
```



# API toolkit – Creating Services for MQ

## Creating a MQ PUT (“POST”) service interface definition

The screenshot displays two windows from the API toolkit:

- Service Interface Editor:** Shows a table of fields for a service interface named "minilnServiceRequest". A red box highlights the "Interface Rename" column for the "loan application" row, which contains the renamed field names: "name", "credit score", "yearly income", "age", "loan amount", "aproved?", "effective date", "yearly interest rate", "yearly payment", "authorization identity", "MESSAGES\_NUM", and "disapproval message". A red circle highlights the "Include" checkboxes for the first four fields.
- Service Project Editor: Configuration:** Shows configuration settings for a service named "twoway Service". A red circle highlights the JNDI name fields:
  - Connection factory JNDI name: jms/qmgrCf
  - Request destination JNDI name: jms/requestQueue
  - Reply destination JNDI name: jms/replyQueue

Again the service developer can then see the imported data structure and can **redact fields**, **rename fields**, and **add default values to fields** to make the service more consumable for an API developer.



# Using JMS to access MQ (One-Way)

mqGetService Service

Service Project Editor: Configuration

Required Configuration

Enter the required configuration for this service.

Connection factory JNDI name: jms/qmgrCf

Destination JNDI name: jms/default

Coded character set identifier (CCSID): 37

Optional Configuration

Enter the optional configuration for this service.

Wait interval:

Message selector:

Definition Configuration

mqClient.xml

Read only Close

Design Source

```
<server description="MQ Service Provider">
<featureManager>
    <feature>zosconnect:mqService-1.0</feature>
</featureManager>
<variable name="wmqJmsClient.rar.location"
    value="/usr/lpp/mqm/V9R1M1/java/lib/jca/wmq.jmsra.rar"/>
<wmqJmsClient nativeLibraryPath="/usr/lpp/mqm/V9R1M1/java/lib"/>
<zosconnect_services>
    <service name="mqPutService">
        <property name="useCallerPrincipal" value="false"/>
    </service>
</zosconnect_services>
<connectionManager id="ConMgr1" maxPoolSize="5"/>
<jmsConnectionFactory id="qmgrCF" jndiName="jms/qmgrCf">
    <connectionManagerRef>ConMgr1</connectionManagerRef>
    <properties.wmqJMS transportType="CLIENT"
        queueManager="ZMQ1"
        channel="LIBERTY.DEF.SVRCONN"
        hostname="wg31.washington.ibm.com"
        port="1422" />
</jmsConnectionFactory>
<jmsQueue id="q1" jndiName="jms/default">
    <properties.wmqJMS
        baseQueueName="ZCEE.DEFAULT.MQZCEE.QUEUE"
        CCSID="37"/>
</jmsQueue>
</server>
```

# Using JMS to access MQ (Two-Way)

\*twoWay Service X

**Service Project Editor: Configuration**

**Required Configuration**

Enter the required configuration for this service.

Connection factory JNDI name:

Request destination JNDI name:

Reply destination JNDI name:

Wait interval:

MQMD format:

Coded character set identifier (CCSID):

Is message persistent:

Reply selection:

Expiry:

**Definition Configuration**

mq.xml

Design Source

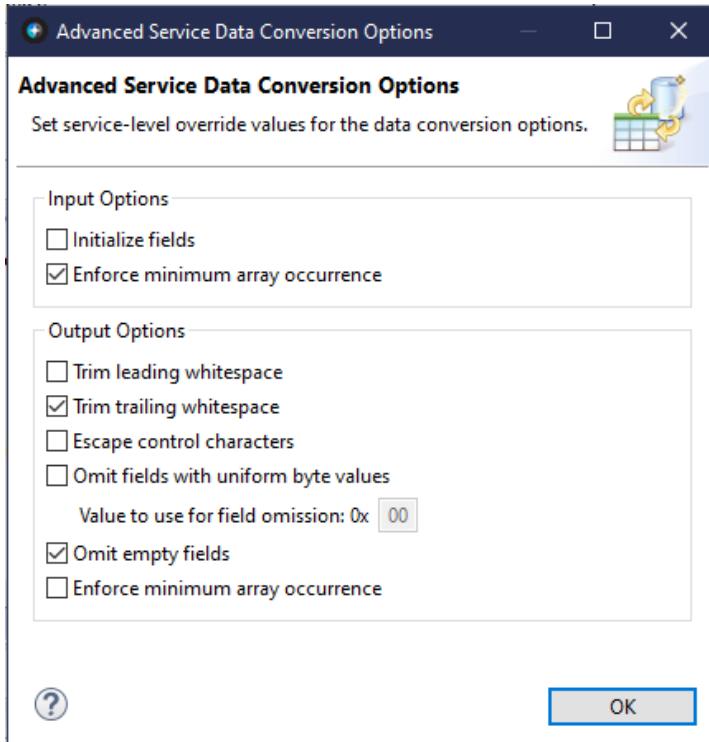
```

2 <featureManager>
3   <feature>zosconnect:mqService-1.0</feature>
4 </featureManager>
5
6 <variable name="wmqJmsClient.rar.location"
7   value="/usr/lpp/mqm/V9R1M1/java/lib/jca/wmq.jmsra.rar"/>
8 <wmqJmsClient nativeLibraryPath="/usr/lpp/mqm/V9R1M1/java/lib"/>
9
10 <connectionManager id="ConMgr1" maxPoolSize="5"/>
11
12 <jmsConnectionFactory id="qmgrCF" jndiName="jms/qmgrCf"
13   connectionManagerRef="ConMgr1">
14   <properties.wmqJms transportType="BROADCASTER"
15     queueManager="QMZ1" />
16 </jmsConnectionFactory>
17
18 <jmsConnectionFactory id="qmgrCF2" jndiName="jms/qmgrCF2"
19   connectionManagerRef="ConMgr1">
20   <properties.wmqJms transportType="CLIENT"
21     queueManager="ZMQ1"
22     channel="LIBERTY.DEF.SVRCONN"
23     hostName="wg31.washington.ibm.com"
24     port="1422" />
25 </jmsConnectionFactory>
26
27 <jmsQueue id="q1" jndiName="jms/default">
28   <properties.wmqJms
29     baseQueueName="ZCONN2.DEFAULT.MQZEE.QUEUE"
30     CCSID="37"/>
31 </jmsQueue>
32
33 <jmsQueue id="requestQueue" jndiName="jms/request">
34   <properties.wmqJms
35     baseQueueName="ZCONN2.TRIGGER.REQUEST"
36     targetClient="MQ"
37     CCSID="37"/>
38 </jmsQueue>
39
40 <jmsQueue id="replyQueue" jndiName="jms/replyQueue">
41   <properties.wmqJms
42     baseQueueName="ZCONN2.TRIGGER.RESPONSE"
43     targetClient="MQ"
44     CCSID="37"/>
45 </jmsQueue>
46
47

```



# API toolkit – Advanced Data Conversion Options



## Request Messages:

- Initialize fields
- Enforce minimum array occurrence

## Response Messages:

- Trim leading whitespace
- Trim trailing whitespace
- Escape control characters
- Omit fields with uniform byte values
- Omit empty fields
- Enforce minimum array occurrence



# API toolkit – Creating Services for Db2

## Creating a service project from Db2 REST service

```
//BIND EXEC PGM=IKJEFT01,DYNAMNBR=20
//STEPLIB DD DSN=DSN1210.DB2.SDSNEXIT,DISP=SHR
//          DD DSN=DSN1210.DB2.SDSNLOAD,DISP=SHR
//SYSTSPRT DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//DSNSTMT DD *
  SELECT EMPNO AS "employeeNumber", FIRSTNAME AS "firstName",
         MIDINIT AS "middleInitial", LASTNAME AS "lastName",
         WORKDEPT AS "department", PHONENO AS "phoneNumber",
         JOB AS "job"
    FROM USER1.EMPLOYEE WHERE EMPNO = :employeeNumber
//SYSTSIN DD *
DSN SYSTEM(DSN2)
BIND SERVICE(SYSIBMSERVICE) -
NAME("selectEmployee") -
SQLENCODING(1047) -
DESCRIPTION('Select an employee from table USER1.EMPLOYEE')
```

Import Db2 service from service manager

Db2 service manager connection: wg31:2446

Type to search...

| Service Name        | Version | Collection ID | Description                                    |
|---------------------|---------|---------------|--|
| selectEmployee      |         | SYSIBMSERVICE | Select an employee from table USER1.EMPLOYEE   |
| deleteEmployee      |         | zCEEService   | Delete an employee from table USER1.EMPLOYEE   |
| displayEmployee     |         | zCEEService   | Display an employee in table USER1.EMPLOYEE    |
| insertEmployee      |         | zCEEService   | Insert an employee into table USER1.EMPLOYEE   |
| selectByDepartments |         | zCEEService   | Select employees by departments                |
| selectByRole        |         | zCEEService   | Select an employee based on job and department |
| selectEmployee      | V1      | zCEEService   | Select an employee from table USER1.EMPLOYEE   |
| selectEmployee      | V2      | zCEEService   | Select an employee from table USER1.EMPLOYEE   |
| updateEmployee      |         | zCEEService   | Update an employee in table USER1.EMPLOYEE     |

Definition Configuration

Import Cancel

\*selectEmployee Service

Service Project Editor: Definition

General Information

Edit or update the general information of the service.

Type: Db2 Service  
Version: 1.0.0  
Description:

Actions

Steps to create a service:

1. Input service version.
2. Import JSON schemas from a Db2 service manager or your local machine.
3. Complete the configuration for the service.
4. Deploy the service.
5. Export the service.

Define Db2 service

Import a Db2 native REST service from a Db2 service manager. Alternatively, enter your Db2 service details and import the JSON schemas from your local machine.

Import from Db2 service manager...

Collection Id: SYSIBMSERVICE  
Db2 native REST service name: selectByRole  
Db2 native REST service version: V1  
Request JSON schema: request-schema.json  
Response JSON schema: response-schema.json

Import from local machine...

The service developer retrieves details about the Db2 REST services

Note there is no service interface editor available

# Accessing a Db2 REST service resource



Service Project Editor: Configuration

Required Configuration

Enter the required configuration for this service.

Connection reference: db2conn

Definition Configuration

DSNL004I -DSN2 DDF START  
COMPLETE  
LOCATION DSN2LOC  
LU  
USIBMWZ.DSN2APPL  
GENERICLU -NONE  
DOMAIN  
WG31.WASHINGTON.IBM.COM  
TCPPORT 2446  
SECPORT 2445  
RESPORT 2447

db2pass.xml

Design Source

```
1 <server description="DB2 REST">
2
3   <zosconnect_zosConnectServiceRestClientConnection id="db2conn"
4     host="wg31.washington.ibm.com"
5     port="2446"
6     basicAuthRef="dsn2Auth" />
7
8   <zosconnect_zosConnectServiceRestClientBasicAuth id="dsn2Auth"
9     applName="DSN2APPL"/>
10
11</server>
12
```

The diagram illustrates the mapping between the Service Project Editor's connection reference and the XML configuration. A red arrow points from the 'db2conn' entry in the 'Connection reference:' field to the 'basicAuthRef="dsn2Auth"' attribute in the XML code. Another red arrow points from the 'wg31.washington.ibm.com' entry in the 'host=' field to the 'applName="DSN2APPL"' attribute in the XML code.



# API toolkit – Services Editor

## Server connection and Services deployment

Manage z/OS Connect EE server connections in the **Host Connections** view:

The screenshot shows the API toolkit interface with two open dialog boxes. On the left, the 'z/OS Connect EE' context menu is open, with 'Deploy Service to z/OS Connect EE Server' highlighted. To the right, two dialogs are displayed: 'Deploy Service to z/OS Connect EE Server Result' and 'Export Service Package'. The 'Deploy Service' dialog shows deployment results for 'z/OS Connect EE Server: wg31:9453' with one service ('cscvincDeleteService') successfully deployed. The 'Export Service Package' dialog allows selecting the export location (Workspace or Local file system), folder ('/Services'), and file name ('cscvincDeleteService.sar'). Both dialogs have 'OK' and 'Cancel' buttons at the bottom.

z/OS Connect EE

- Restore from Local History...
- Compare With
- Configure
- Source

Deploy Service to z/OS Connect EE Server

Export z/OS Connect EE Service Archive

Deploy Service to z/OS Connect EE Server Result

z/OS Connect EE Server: wg31:9453

Deployment results:

| Service name       | Version | Type                 | Result  |
|--------------------|---------|----------------------|---------|
| cscvincDeleteSe... | 1.0.0   | CICS Channel Serv... | Updated |

All services were deployed successfully.

OK

Export Service Package

Select where to export your service package. The package is exported as a service archive file (SAR).

Export service package to:

Workspace

Local file system

Folder: /Services

File name: cscvincDeleteService.sar

Overwrite service package file

OK Cancel

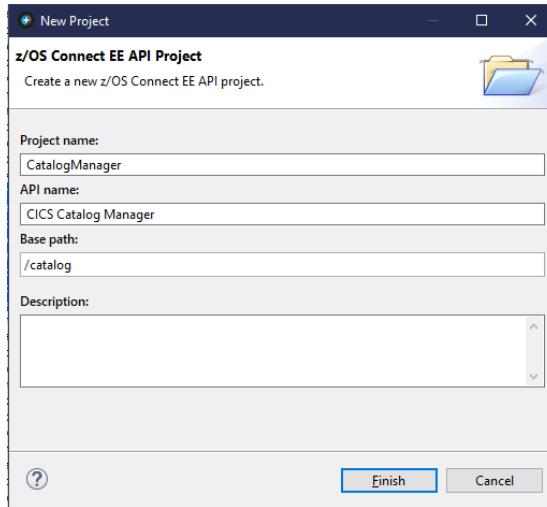


## Once we have a Service Archive (SAR) What's next?

Quick and easy **API mapping**.

*Remember: All service archives files are functionally equivalent regardless of how they are created*

# API toolkit – API Editor



**CICS Catalog Manager API**

**API Editor**

**Describe your API**

Name: CICS Catalog Manager      Description:

Base path: /catalog

Version: 1.0.0

**Contact Information**

**Path**

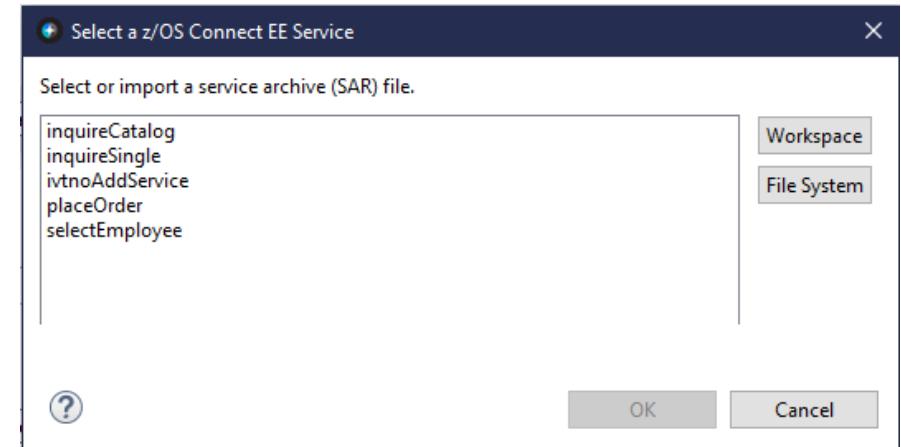
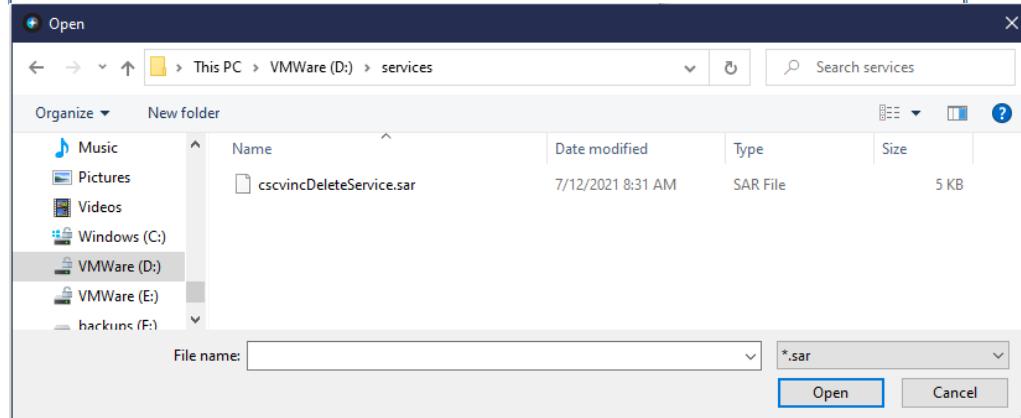
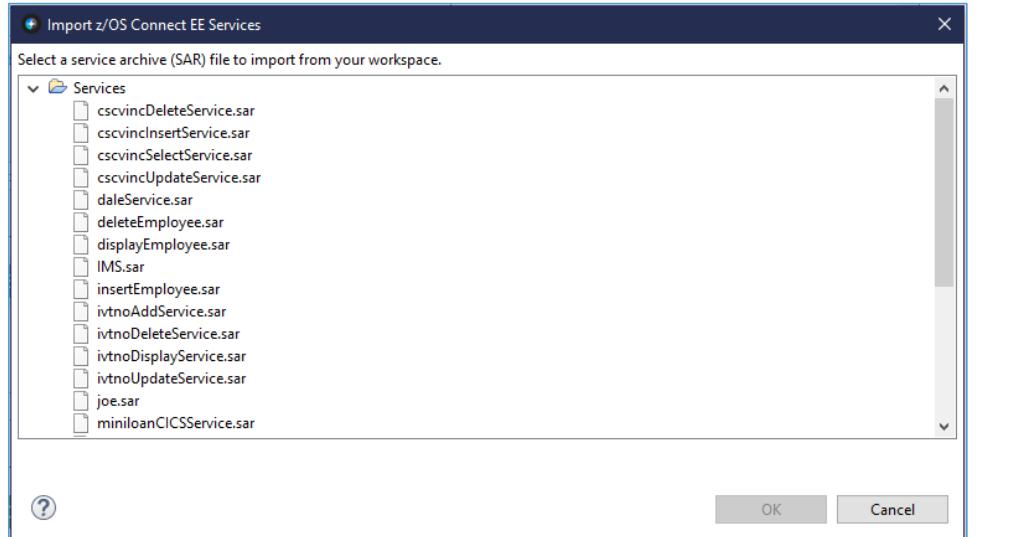
/newPath1

**Methods (4)**

| Method | Service... | Mapping... | Up | Down | Delete |
|--------|------------|------------|----|------|--------|
| POST   |            |            |    |      |        |
| GET    |            |            |    |      |        |
| PUT    |            |            |    |      |        |
| DELETE |            |            |    |      |        |



# Importing the service archives files





# API toolkit – API Editor

The screenshot shows the API Toolkit API Editor interface. It displays three API endpoints:

- catalog**: Path: /catalog, Methods: GET (inquireCatalog), PUT (ivtnoAddService)
- order**: Path: /order, Methods: POST (placeOrder), PUT (selectEmployee)
- item**: Path: /item/{itemID}, Methods: GET (inquireSingle)

A large red oval highlights the **item** endpoint.

mitchj@us.ibm.com

The **API toolkit** is designed to encourage RESTful API design.

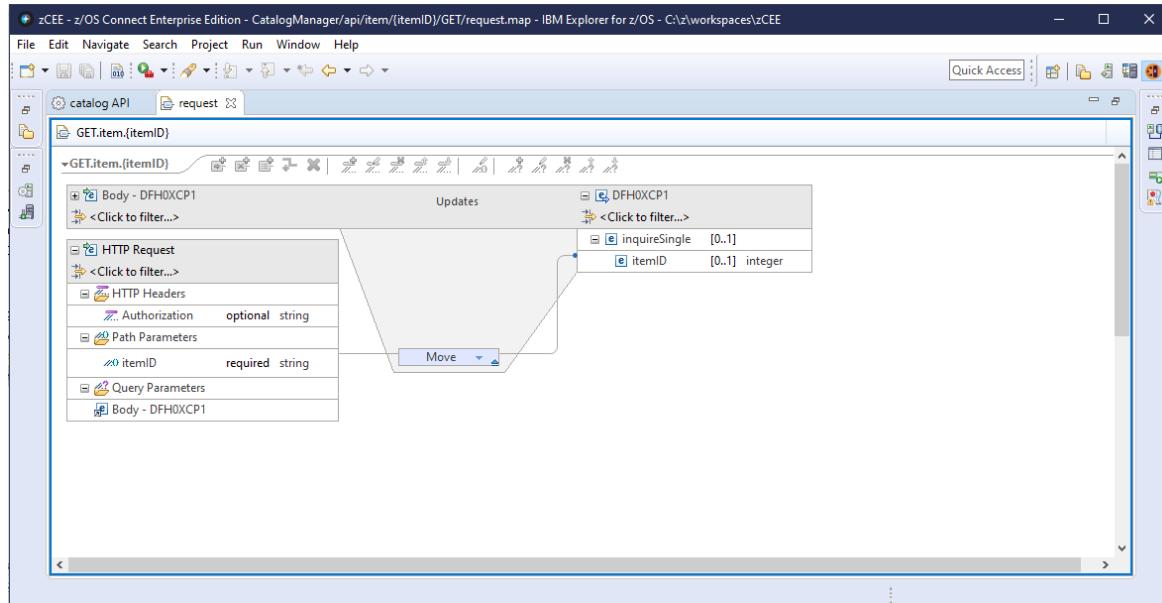
Once you define your API, you can map backend services to each request.

Your services are represented by a **.sar** files, which you import into the **API toolkit**, regardless of how the service archive file was generated.



# API toolkit – API Editor

API mapping: Assign values to the interface fields exposed by the service developer



Map both the request and response for each API.

Map path and query parameters to native data structures.

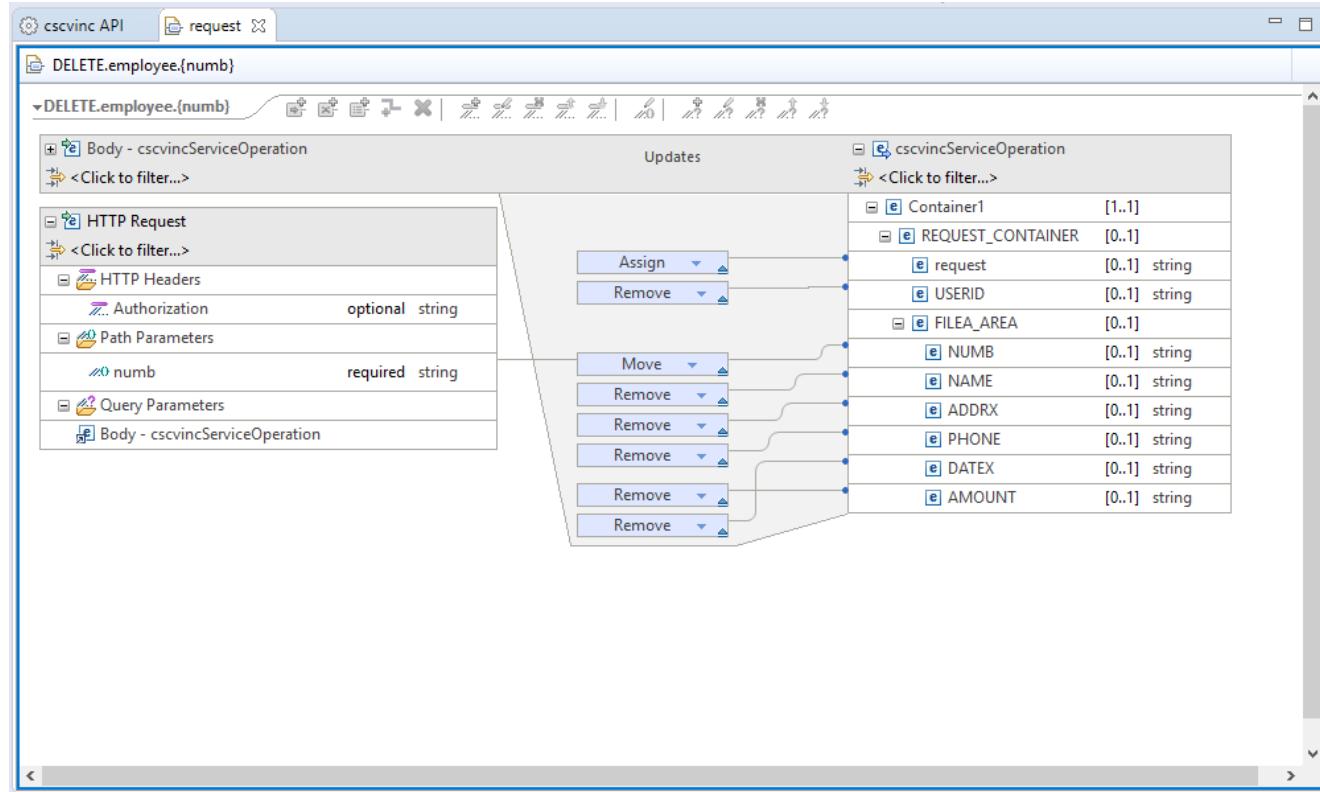
Assign static values to fields, useful for Op codes.

Remove unwanted fields to simplify the API (remember request was set to 01INQC in the SAR).



# API toolkit – API Editor

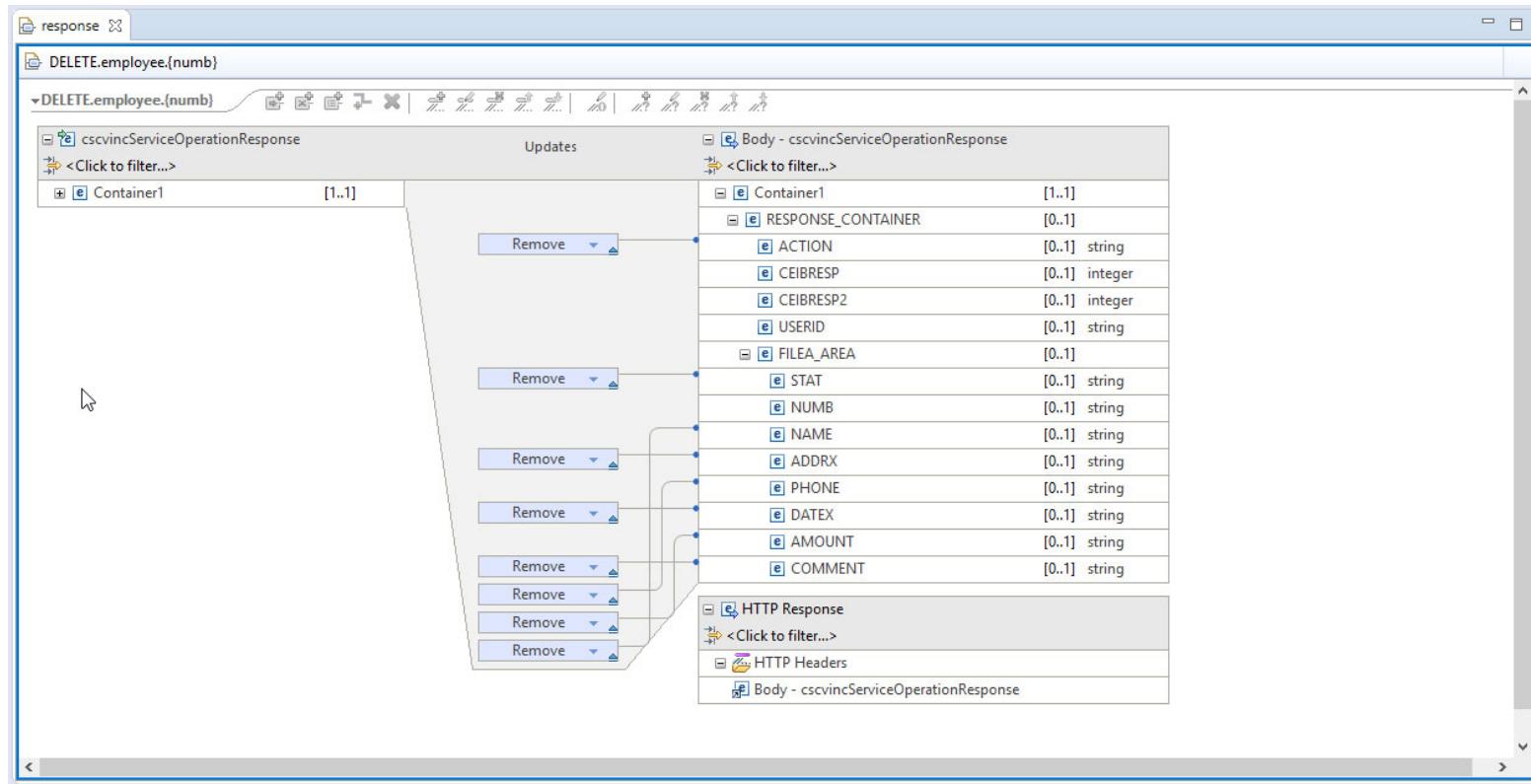
API mapping: Remove or assign values to the fields exposed by service developer





# API toolkit – API Editor

API mapping: Allows the API Developer to remove fields from the response to tailor the API





# API toolkit – Header properties

API mapping: Allows adding HTTP header properties

The screenshot shows the API toolkit interface with a window titled "request" containing a "POST.queue" configuration. On the left, under "Body - MQPUTOOperation", there is a section for "HTTP Request" which includes "HTTP Headers". This section is circled in red, highlighting the "Authorization" and "ibm-mq-md-correlID" fields. To the right, under "MQPUTOOperation", there is a list of message fields:

| Field     | Type          | Default Value |
|-----------|---------------|---------------|
| mqmessage | [1..1]        |               |
| stat      | [1..1] string |               |
| numb      | [1..1] string |               |
| name      | [1..1] string |               |
| addrx     | [1..1] string |               |
| phone     | [1..1] string |               |
| datex     | [1..1] string |               |
| amount    | [1..1] string |               |
| comment   | [1..1] string |               |



# API toolkit

## API mapping: API definition with multiple response codes

The screenshot shows the API toolkit interface for defining API operations. On the left, a tree view shows a path: /employee/{employee}. Underneath it, four methods are listed: GET, POST, DELETE, and PUT, each associated with a service name like cscvinc...

In the center, a detailed view of a GET operation named getCscvincSelectService is shown. It includes fields for Operation id and Response codes. A modal window titled "Edit Response 404" is open, showing the response code as "404 - Not Found" and a description as "Not Found". Below this, rules are defined to determine when this response code is used:

| Rule   | Condition                                 | Value | Operator | Value | Logic |
|--------|---|-------|----------|-------|-------|
| Rule 1 | se/Container1/RESPONSE_CONTAINER/CEIBRESP | =     | =        | 13    | AND   |
| Rule 2 | /Container1/RESPONSE_CONTAINER/CEIBRESP2  | =     | =        | 80    |       |

At the bottom of the modal, a summary shows "Rule 1 AND Rule 2".

The **API toolkit** supports defining multiple response codes per API operation.

Separate mappings can be defined for each response code.

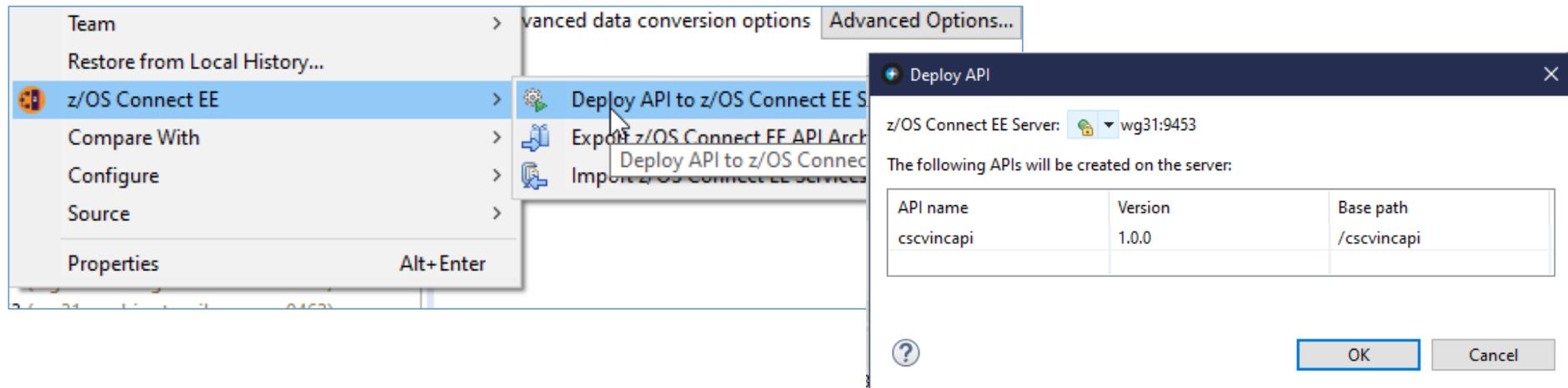
You can define rules based on fields in the service's return interface to tell z/OS Connect EE which response code to return



# API toolkit – API Editor

## Server connection and API deployment

Manage z/OS Connect EE server connections in the **Host Connections** view:



**Right-click deploy to server** enables developers to quickly deploy, test, and iterate on their APIs.

**z/OS Connect EE servers view** allows you to start, stop, and remove APIs from a running server.



# API toolkit – API Editor

## Testing with Swagger UI

Test your deployed APIs directly with **Swagger UI** inside the editor.  
No need to export the Swagger doc to a separate tool.

The screenshot shows the z/OS Connect EE Servers interface with the API Editor open. On the left, the navigation pane shows 'wg31:9443 (wg31.washington.ibm.com:9443)' with 'APIs (1)' expanded, showing 'cscvinc (S)' and its methods: 'Open In Swagger UI', 'Open In API Explorer', 'Start API', 'Stop API', 'Remove API', and 'Show Properties View'. Below this is a list of 'Services (4)'. The main pane displays the 'Swagger UI' interface for the 'cscvinc' service. It shows four methods: POST /employee, DELETE /employee/{employee}, GET /employee/{employee}, and PUT /employee/{employee}. The GET method is selected. To the right, a detailed view of the GET /employee/{employee} endpoint is shown, including the Response Class (Status 200), OK status, Model (Employee), Example Value (JSON object), Response Content Type (application/json), Parameters (Authorization header and employee path parameter), and Response Messages (404 Not Found). The URL in the browser is localhost:55221/?url=/api-docs/wg31.washington.ibm.com:9443/cscvinc/employee/{employee}.



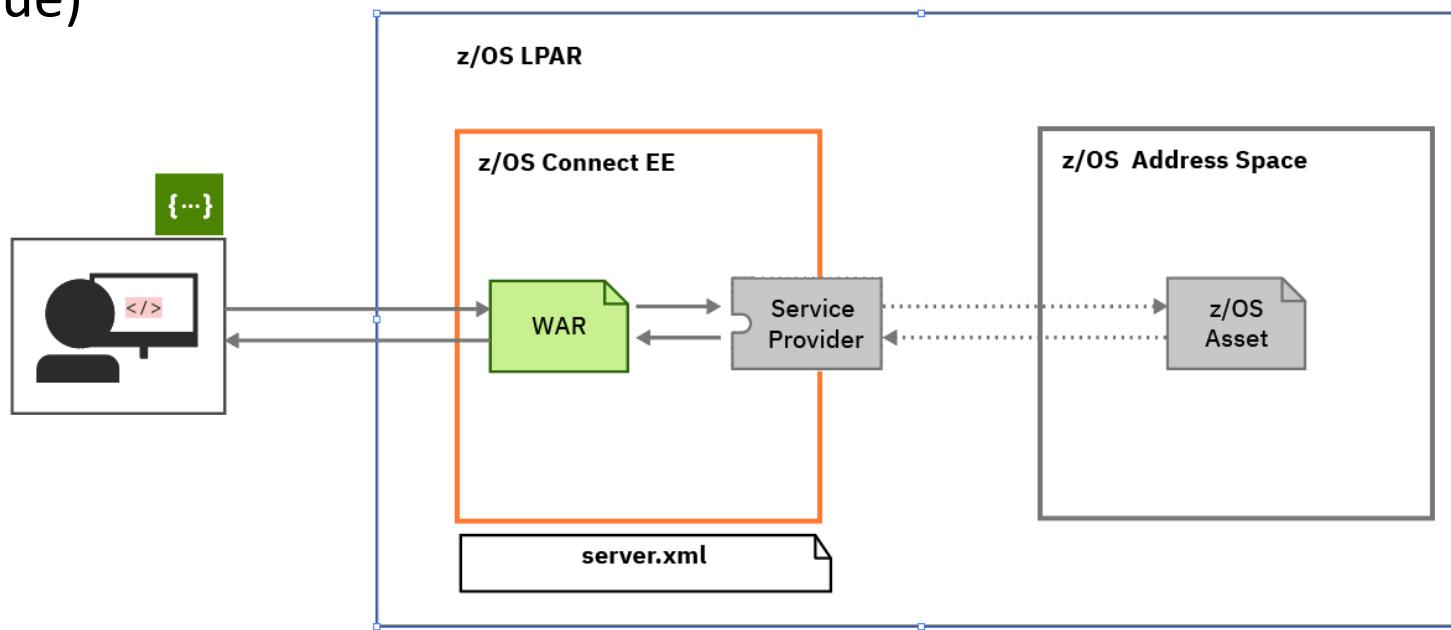
## **/zosConnect/designer**

Simple API Development using the z/OS Connect Designer



# Accessing the CICS or Db2 asset (Open API 3)

- z/OS Connect OpenAPI 3 APIs are developed using a z/OS Connect Designer web browser tool to developer Web ARchive (WAR) files (a traditional Java packaging technique)



- The WAR provides the RESTful interface is ready to be consumed by a client and it requires the client to have no knowledge that a z/OS resource is being accessed as well as the mapping and transformation for accessingg the resource.
- The Service Provider is tightly coupled to a specific instance of a resource (e.g., host and port)

# Accessing a z/OS asset starting with a YAML description of an API (OpenAPI 3)

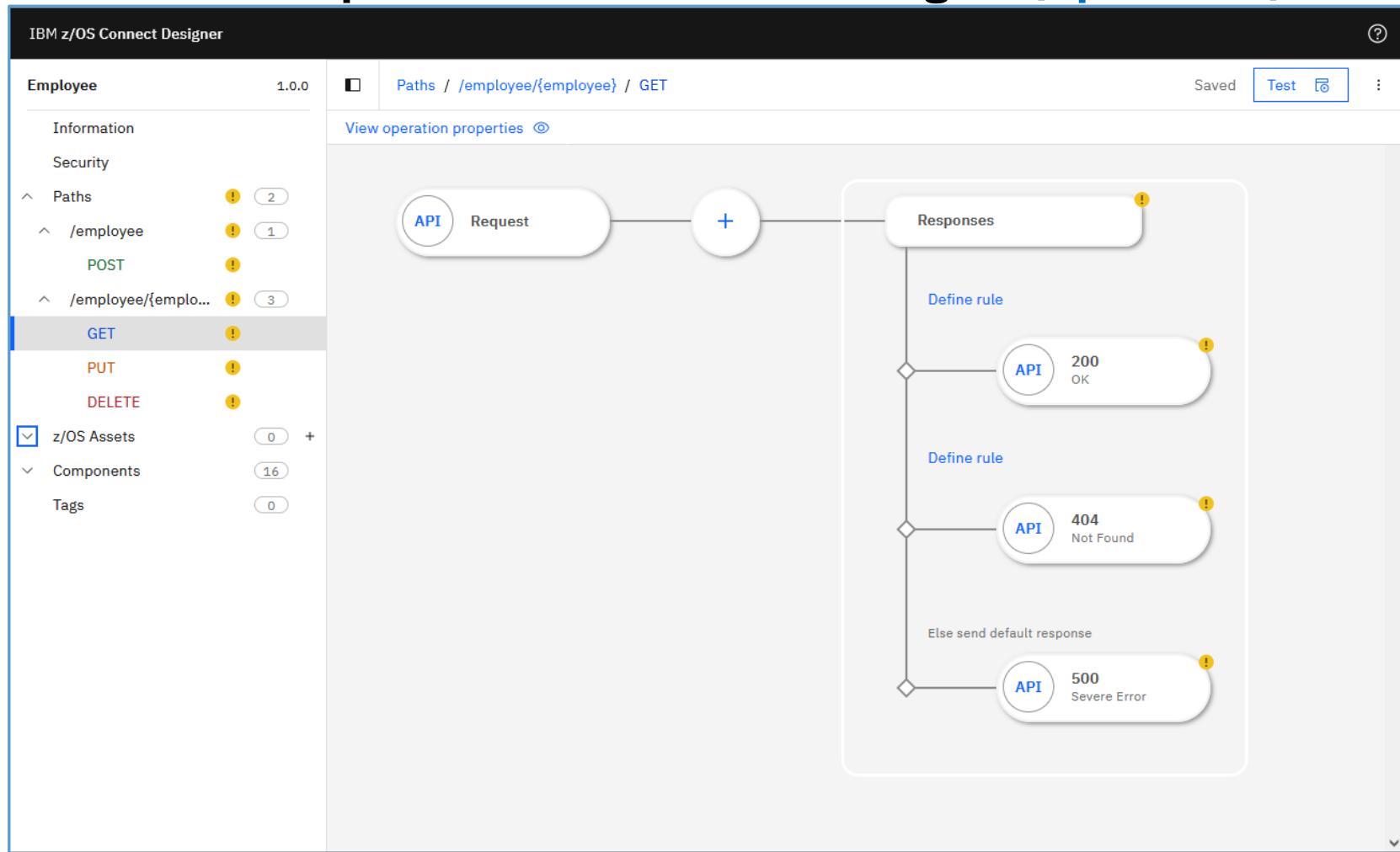


```
cscvinc.yaml - Notepad
File Edit Format View Help
openapi: 3.0.0
info:
  description: "CICS Filea Sample VSAM Application"
  version: 1.0.0
  title: cscvinc
x-ibm-zcon-roles-allowed:
- Manager
paths:
  /employee:
    post:
      tags:
        - cscvinc
      operationId: postCscvincInsertService
      parameters:
        - name: Authorization
          in: header
          required: false
          schema:
            type: string
      requestBody:
        content:
          application/json:
            schema:
              $ref: "#/components/schemas/postCscvincInsertService_request"
        description: request body
        required: true
      responses:
        "200":
          description: OK
          content:
            application/json:
              schema:
                $ref: "#/components/schemas/postCscvincInsertService_response_200"
        "400":
          description: Bad Request
          content:
            application/json:
              schema:
                $ref: "#/components/schemas/postCscvincInsertService_response_400"
Ln 33, Col 74 100% Windows (CRLF) UTF-8
```

```
cscvinc.yaml - Notepad
File Edit Format View Help
getEmployeeSelectService_response_200:
  type: object
  properties:
    summary:
      $ref: '#/components/schemas/getEmployeeSelectService_response_200_message'
    detail:
      $ref: '#/components/schemas/getEmployeeSelectService_response_200_detail'
  getEmployeeSelectService_response_200_message:
    type: object
    properties:
      message:
        type: string
      example:
        message: record retrieved
  getEmployeeSelectService_response_200_detail:
    type: object
    properties:
      EmployeeSelectServiceOperationResponse:
        type: object
        properties:
          employeeData:
            type: object
            properties:
              response:
                type: object
                properties:
                  employeeDetails:
                    type: object
                    properties:
                      employeeNumber:
                        type: string
                        maxLength: 6
                      name:
                        type: string
                        maxLength: 20
                      address:
                        type: string
                        maxLength: 20
                      phoneNumber:
                        type: string
                        maxLength: 8
                      date:
                        type: string
                        maxLength: 8
Ln 196, Col 27 100% Windows (CRLF) UTF-8
```



# Import the YAML description of an API into the Designer (OpenAPI 3)



# Describe the z/OS asset, accessing a CICS program (OpenAPI 3)



IBM z/OS Connect Designer

cscvinc 1.0.0

Information  
Security  
Paths 2  
z/OS Assets 1  
programCscvinc  
Components 8  
Tags 0

Step 2 of 5  
Add z/OS Asset

Select a z/OS Asset type  
CICS channel program

CICS program name  
CSCVINC

Program language  
COBOL

CCSID  
037

Select a CICS connection  
cicsConn

Optional configuration  
Transaction ID (optional)  
Input Transaction ID

Transaction ID usage (optional)  
Select usage

Previous Next



# Or accessing a z/OS IMS transaction (OpenAPI 3)

IBM z/OS Connect Designer

Employee 1.0

Information

Security

Paths 1 2

z/OS Assets 0

Components 16

Tags 0

Step 2 of 5

Add z/OS Asset

Select a z/OS Asset type

IMS transaction

Transaction code

IVTNO

Program language

COBOL

Select an IMS connection

imsConn

Previous

Next



# Or accessing a Db2 REST service (OpenAPI 3)

IBM z/OS Connect Designer

EmployeesApi 1.1

Information

Security

Paths

/employees/{id}

GET

PUT

DELETE

/employees

GET

POST

z/OS Assets

addEmployee

deleteEmployee

getEmployee

getEmployees

selectByRole

updateEmployee

Components

Tags

View

Step 3 of 4

Add z/OS Asset

Select a Db2 connection

db2Conn

Import from Db2 service manager

Db2 native REST service collection ID

e.g. SYSIBMSERVICE

Db2 native REST service name

e.g. myService

Db2 native REST service version (optional)

e.g. V1

Import Db2 native REST service request schema

Drag and drop or select a file  
JSON schema specification draft 4 and 5 supported

Specify a URL

http://github.com/example/api-docs

Import file

Import Db2 native REST service response schema

Drag and drop or select a file  
JSON schema specification draft 4 and 5 supported

Previous

Next



# Select the z/OS Db2 REST Service (OpenAPI 3)

Add z/OS Asset / Import Db2 native REST service

## Import Db2 native REST service

Select a Db2 connection

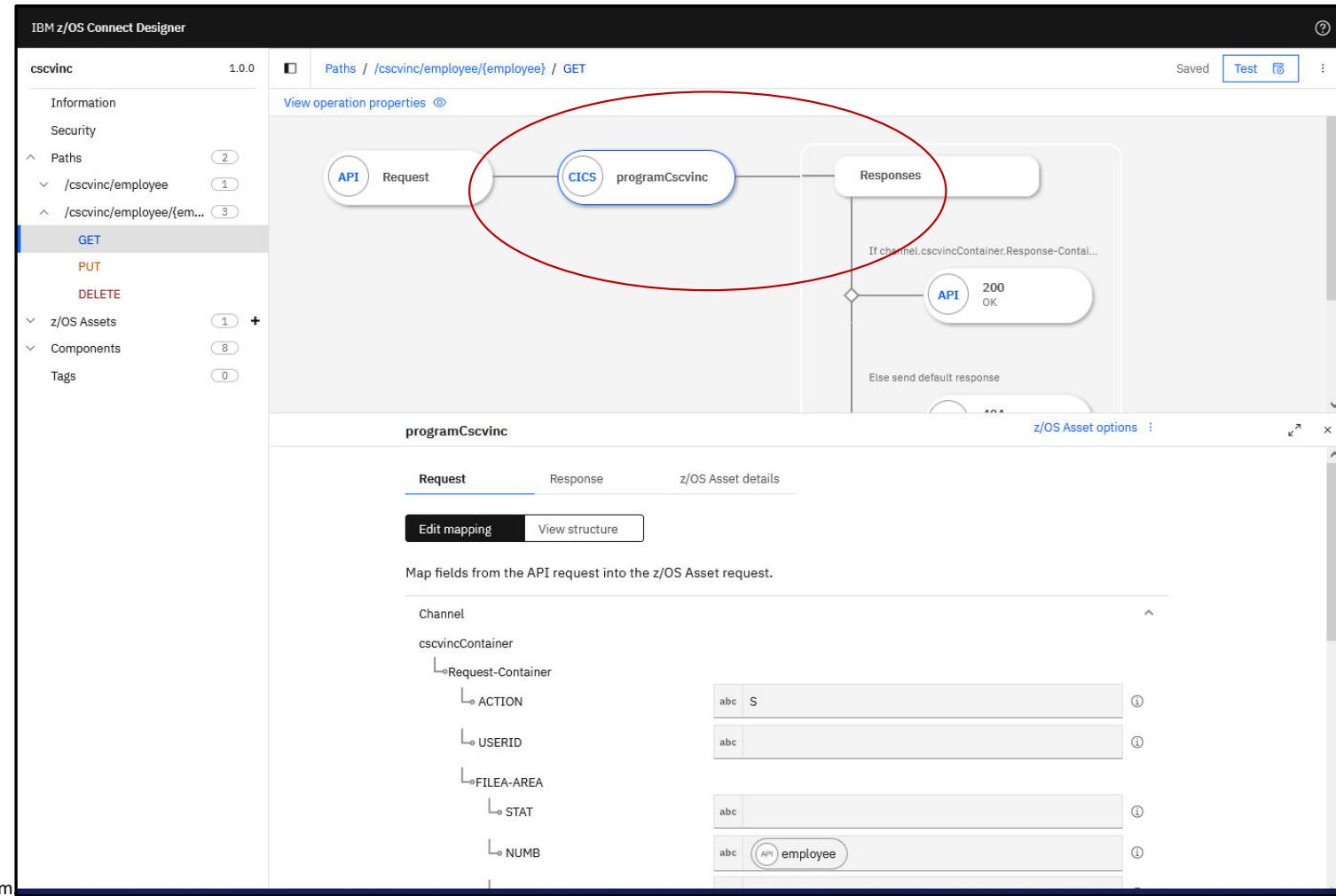
db2Conn

| Service name        | Version | Collection ID | Path                       | Description                  | Status    |
|---------------------|---------|---------------|----------------------------|------------------------------|-----------|
| addEmployee         | V1      | SYSIBMSERVICE | /services/SYSIBMSERVI...   | Add the details of an ind... | Available |
| deleteEmployee      | V1      | SYSIBMSERVICE | /services/SYSIBMSERVI...   | Remove the details of a...   | Available |
| getEmployee         | V1      | SYSIBMSERVICE | /services/SYSIBMSERVI...   | Get the details of a spec... | Available |
| getEmployees        | V1      | SYSIBMSERVICE | /services/SYSIBMSERVI...   | Get the details of all em... | Available |
| updateEmployee      | V1      | SYSIBMSERVICE | /services/SYSIBMSERVI...   | Update the details of an...  | Available |
| deleteEmployee      | V1      | zCEEService   | /services/zCEEService/...  | Delete an employee fro...    | Available |
| displayEmployee     | V1      | zCEEService   | /services/zCEEService/...  | Display an employee in ...   | Available |
| insertEmployee      | V1      | zCEEService   | /services/zCEEService/i... | Insert an employee into...   | Available |
| selectByDepartments | V1      | zCEEService   | /services/zCEEService/s... | Select employees by de...    | Available |
| selectByRole        | V1      | zCEEService   | /services/zCEEService/s... | Select an employee bas...    | Available |

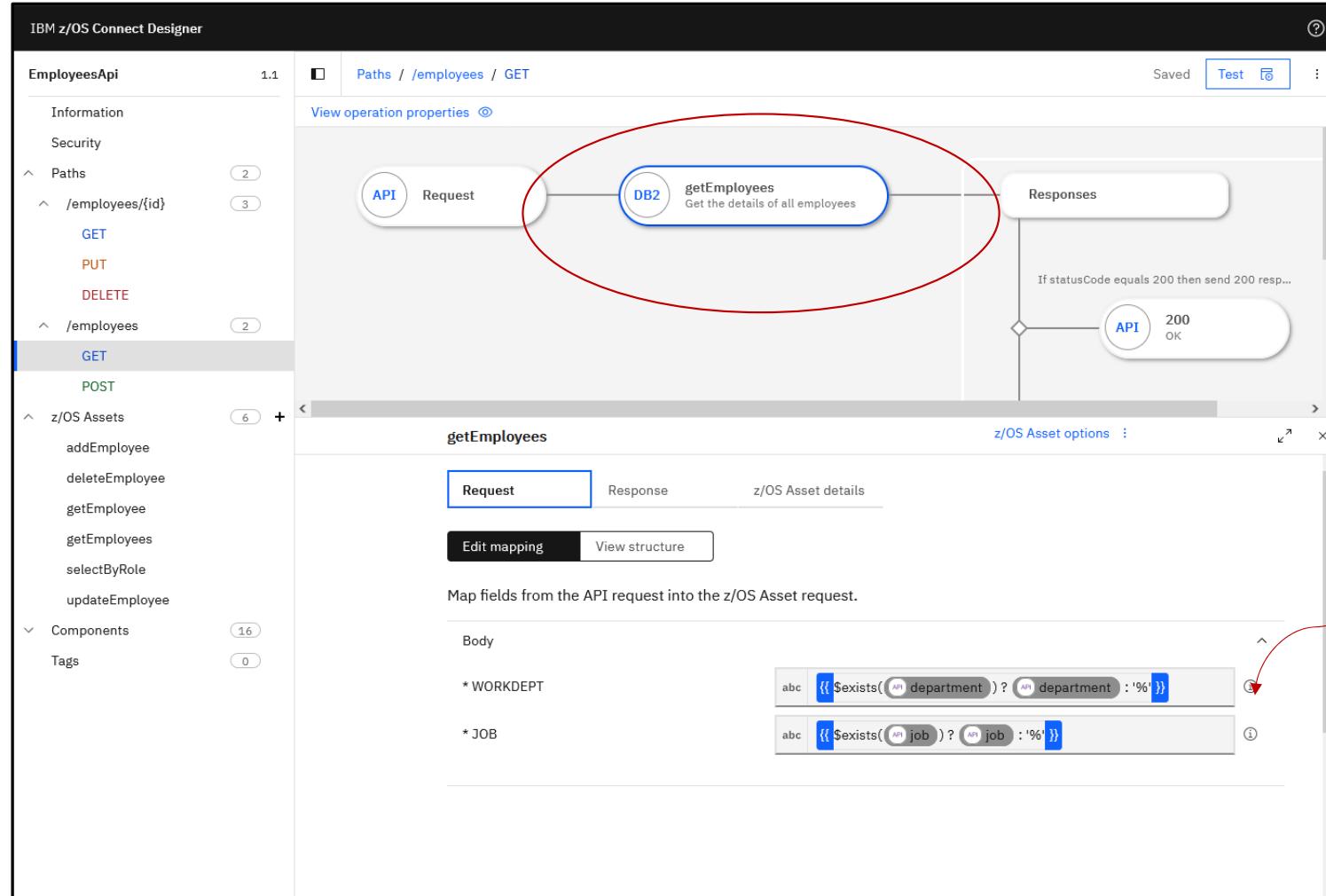
Items per page: 10 1–10 of 12 items 1 of 2 pages

Previous Import

# Map the method and request message with a serialized message (OpenAPI 3)



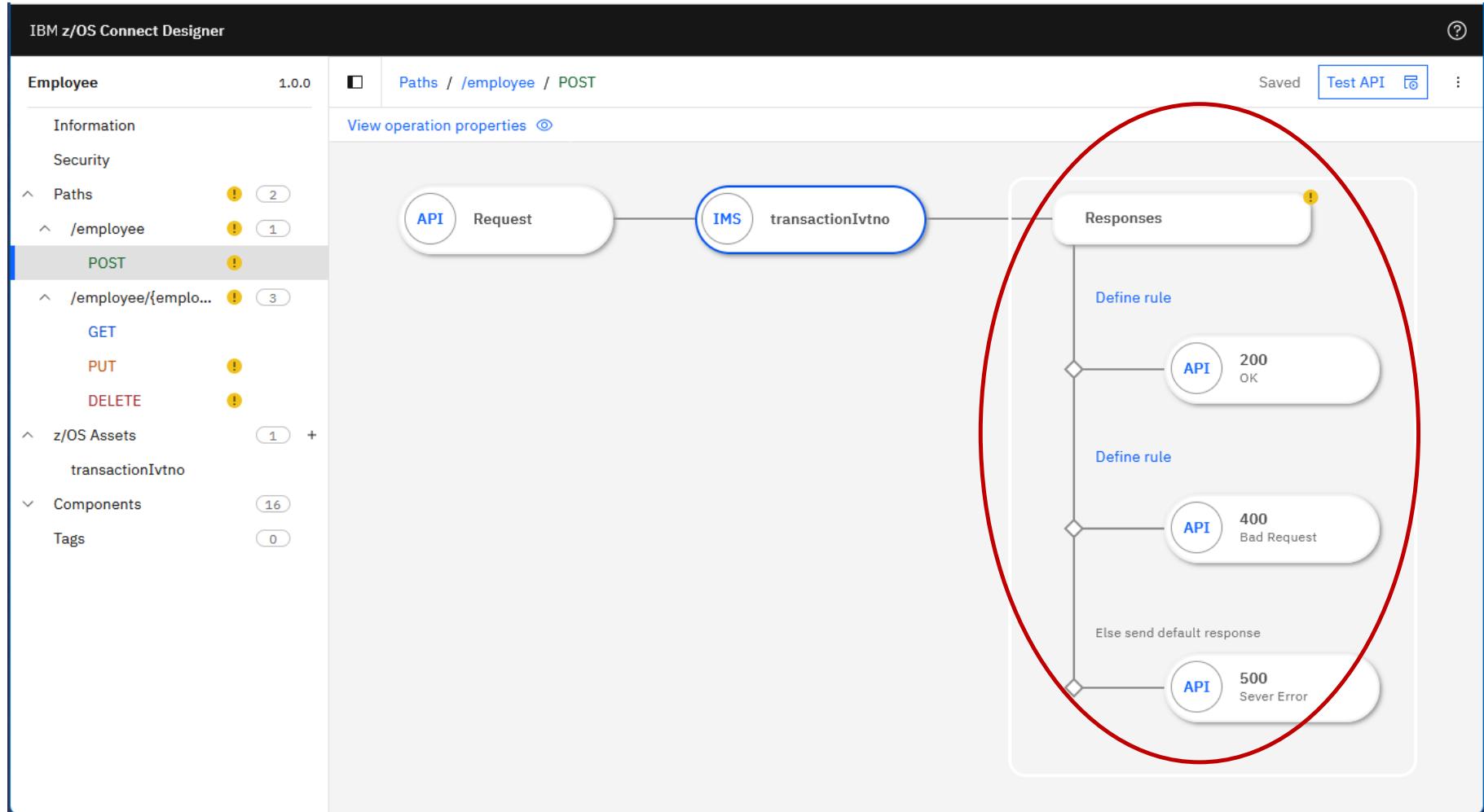
# Map the method and the response message with a Db2 REST service (OpenAPI 3)



JSONata example



# Map results to the specification's responses (OpenAPI 3)





# Map the serialized response message (OpenAPI 3)

Paths / /employee/{employee} / GET

200 - OK

Edit mapping View structure

Map fields from the z/OS Asset response into the API response.

Body

summary

- message

detail

- cscvincSelectServiceOperationResponse
  - \*cscvincContainer
    - response
      - CEIBRESP
      - CEIBRESP2
      - USERID
    - filea
      - employeeNumber
      - name
      - address
      - phoneNumber
      - date
      - amount
      - comment

abc Record (NUMB) successfull retrieved by (USERID)

123

123

abc

abc (NUMB)

abc (NAME)

abc (ADDRX)

abc phone

abc (DATEX)

abc (AMOUNT)

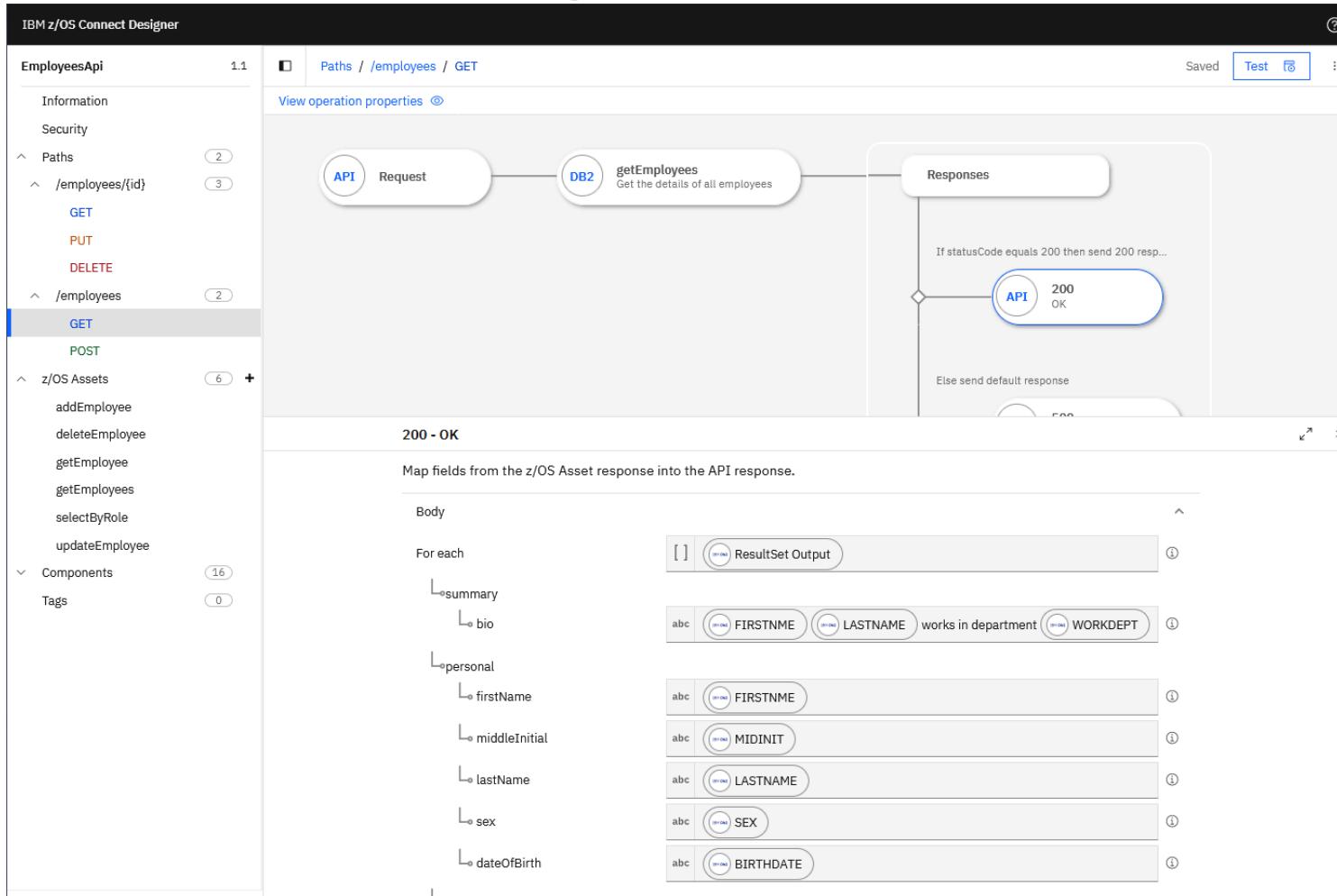
abc (COMMENT)



JSONata example



# Map the response message (OpenAPI 3)



*ResultSet Output* is a list or array of results and the “For each” processes each element in the list.



# z/OS Connect Designer for OpenAPI 3 (200)

The screenshot shows the IBM z/OS Connect Designer interface for creating an API operation. The left sidebar lists the API structure:

- EmployeesApi**:
  - Information
  - Security
  - Paths
    - /employees/{id}:
      - GET
      - PUT
      - DELETE
    - /employees:
      - GET
      - POST
  - z/OS Assets
    - displayEmployee
    - getEmployee
    - getEmployees
  - Components (16)
  - Tags (0)

The main workspace displays the flow for the GET operation on the /employees path:

- Request**: An API request is sent to a **DB2** database node labeled **getEmployees** with the description "Get the details of all employees".
- Responses**: The response is handled based on the status code. A condition "If statusCode equals 200 then send 200 resp..." leads to a **200 OK** response.
- Body**: The response body is defined as "For each" item. It includes:
  - summary**:
    - bio**: abc [FIRSTNME LASTNAME] is in department WORKDEPT
  - personal**:
    - firstName**: abc FIRSTNME
    - middleInitial**: abc MIDINIT



# z/OS Connect Designer for OpenAPI 3 (404)

IBM z/OS Connect Designer

EmployeesApi 1.1 Paths / /employees/{id} / PUT Saved Test

Information Security Paths /employees/{id} 2 3

GET PUT DELETE

/employees 2 z/OS Assets 6 + Components 16 Tags 0

View operation properties

200 Updated

If "Update Count" equals 0 then send 404 res...

404 Not Found

Else send default response

500 Internal Server Error

404 - Not Found

Edit mapping View structure

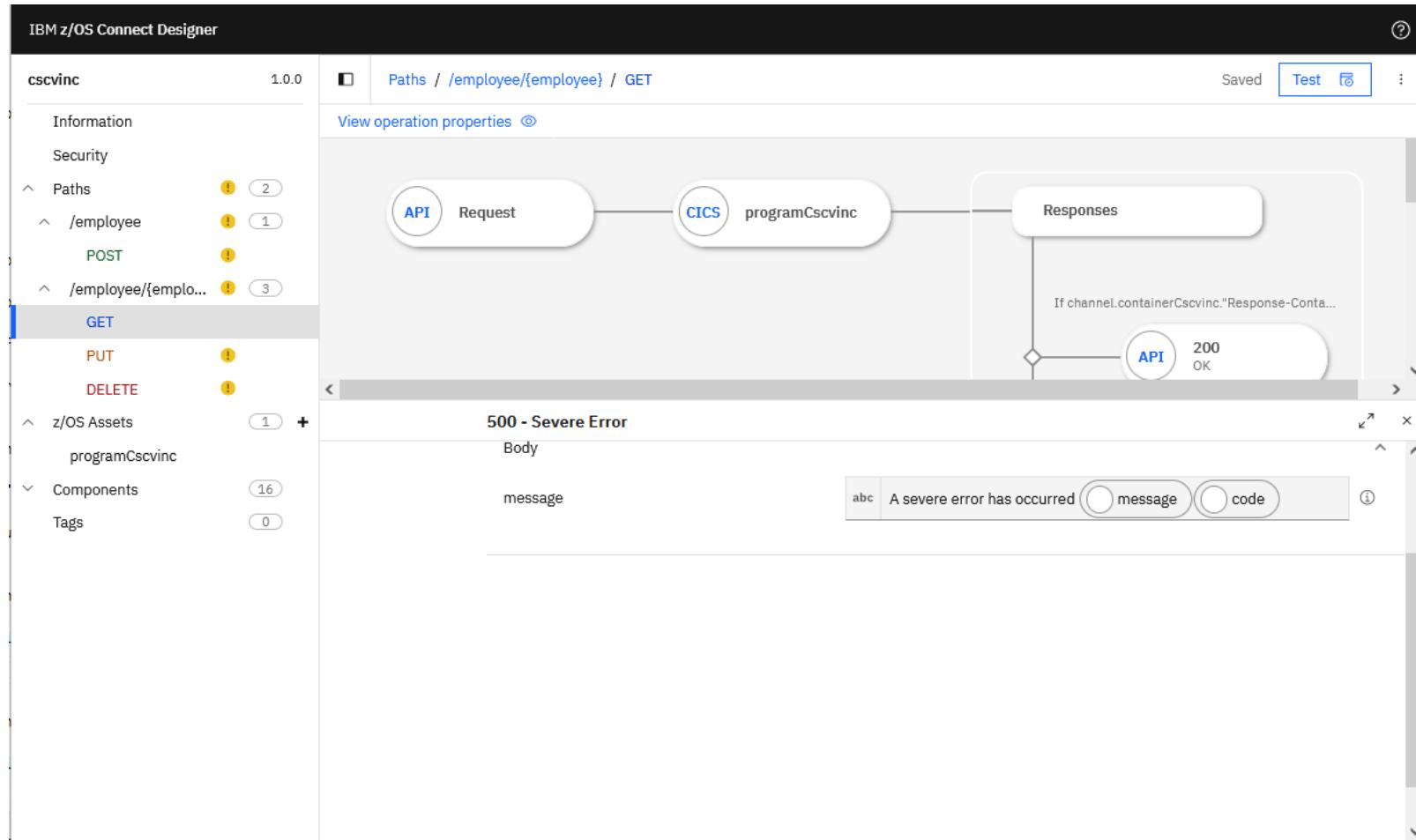
Map fields from the z/OS Asset response into the API response.

Body

message abc Employee not found ⓘ

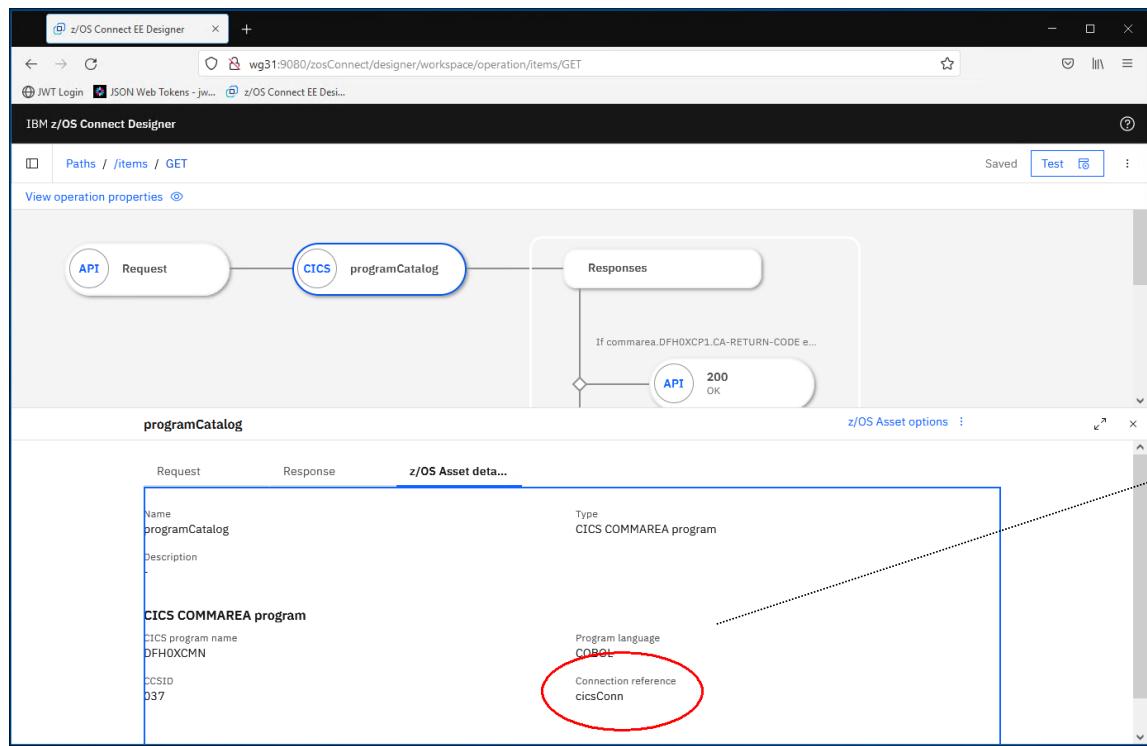


# z/OS Connect Designer for OpenAPI 3 (500)





# Server XML - Accessing a CICS program using IPIC (OpenAPI 3)



The screenshot shows the 'Server Config' interface with a tab for 'cics.xml'. It has 'Design' and 'Source' tabs, with 'Source' selected. The XML code is:

```
1<server description="CICS IPIC connections">
2
3<!-- Enable features -->
4<featureManager>
5   <feature>zosconnect:cics-1.0</feature>
6</featureManager>
7
8<zosconnect_cicsIpicConnection id="cicsConn" host="${CICS_HOST}">
9   port="${CICS_PORT}" />
10
11</server>
12
```

The connection references identifies a `zosconnect_cicsIpicConnection` configuration element. Which provides the connection details to a CICS region.



# Server XML - Accessing an IMS transaction(OpenAPI 3)

The screenshot shows two windows side-by-side. On the left is the 'IBM z/OS Connect Designer' interface. It displays an 'Employee' API version 1.0.0. Under the 'Paths' section, there's a 'transactionIvtno' asset under the '/employee/{employee}' path. This asset is defined as an 'IMS transaction' with transaction code 'IVTNO' and program language 'COBOL'. A connection profile named 'imsConn' is selected. On the right is the 'Server Config' window, showing the 'ims.xml' file. The XML configuration includes a server definition with a zosconnect\_imsConnection element, which points to a connection factory named 'imsConn'.

```
<?xml version="1.0" encoding="UTF-8"?>
<server description="zosconnect_imsConnection" >
  <zosconnect_imsConnection id="imsConn" >
    <connectionFactoryRef="imsConnectionFactory" >
      <imsDatastoreName="${IMS_DATASTORE}" />
    </connectionFactoryRef>
    <connectionFactory id="imsConnectionFactory" >
      <containerAuthDataRef="IMSCredentials" >
        <properties.gmoa hostName="${IMS_HOST}" portNumber="${IMS_PORT}" />
      </connectionFactory>
    </connectionFactory>
    <authData id="IMSCredentials" user="${IMS_USER}" password="${IMS_PASSWORD}" />
  </zosconnect_imsConnection>
</server>
```

The connection references identifies a `zosconnect_imsConnection` element. Which provides the connection details to IMS.



# Server XML - Accessing a Db2 REST service (OpenAPI 3)

The screenshot shows the z/OS Connect EE Designer interface. A flow diagram illustrates a request from an API endpoint to a DB2 native REST service named 'getEmployee-V1'. This service retrieves details of all employees. The response is handled by an API endpoint that checks a condition (If commarea.DFH0XCP1.CA-RETURN-CODE e...) and returns a 200 OK status. Below the diagram, the 'getEmployee-V1' service configuration is detailed, including its name, description, type (Db2 native REST service), version (V1), and connection reference ('Connection reference db2Conn'). A red arrow points from this connection reference to the 'zosconnect\_db2Connection' element in the Server Config XML.

The screenshot shows the Server Config interface with the 'db2.xml' configuration file open. The 'Source' tab displays the XML code for a server definition:

```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <server description="Db2 Connections">
3   <featureManager>
4     <feature>zosconnect:db2-1.0</feature>
5   </featureManager>
6   <zosconnect_credential user="${DB2_USERNAME}">
7     password="${DB2_PASSWORD}" id="commonCredentials" />
8   <zosconnect_db2Connection id="db2Conn" host="${DB2_HOST}">
9     port="${DB2_PORT}" credentialRef="commonCredentials" />
10 </server>
11

```

A red arrow points from the 'zosconnect\_db2Connection' element in the XML to the 'Connection reference db2Conn' entry in the z/OS Asset details panel of the z/OS Connect EE Designer. A callout box contains the text: "Define connections to Db2 using variables defined in bootstrap.properties file".

```

DSNL004I -DSN2 DDF START COMPLETE
LOCATION  DSN2LOC
LU        USIBMWZ.DSN2APPL
GENERICLU -NONE
DOMAIN    WG31.WASHINGTON.IBM.COM
TCPPORT   2446
SECPORT   2445
RESPORT   2447

```

The connection references identifies a `zosconnect_db2Connection` configuration element. Which provides the connection details to a DB2 DDF task.



## EJB roles for z/OS Connect (OpenAPI 3)

```
<safCredentials unauthenticatedUser="WSGUEST" profilePrefix="BBGZDFLT" />  
  
<webApplication id="CatalogManager" location="${server.config.dir}/apps/api.war" contextRoot="catalog"  
name="CatalogManager"/>  
  
<safRoleMapper profilePattern=%profilePrefix%.%resourceName%.%role%
```

```
openapi: 3.0.0  
...  
servers:  
- url: /  
x-ibm-zcon-roles-allowed:  
- Manager  
...  
paths:  
/items:  
  get:  
    operationId: itemsGet  
    ...  
/items/{id}:  
  get:  
    ...  
    operationId: itemsIdGet  
    x-ibm-zcon-roles-allowed:  
    - Staff  
/orders:  
  post:  
    ...  
    operationId: ordersPost  
    x-ibm-zcon-roles-allowed:  
    - Staff
```

*From the OpenApi document, the value for %role% would be either Manager or Staff.*

So, the required SAF EJB roles to be defined would be:

- *BBGZDFLT.CatalogManager.Manager*
- *BBGZDFLT.CatalogManager.Staff*

*REDFINE EJBROLE BBGZDFLT.CatalogManager.Manager  
REDFINE EJBROLE BBGZDFLT.CatalogManager.Staff*

Access to use the GET method to invoke /items would require read access to EJB role *BBGZDFLT.CatalogManager.Manager*.

Access to use the GET method to invoke /items/{id} and the POST method to invoke /orders would require read access to EJB role *BBGZDFLT.CatalogManager.Staff*.



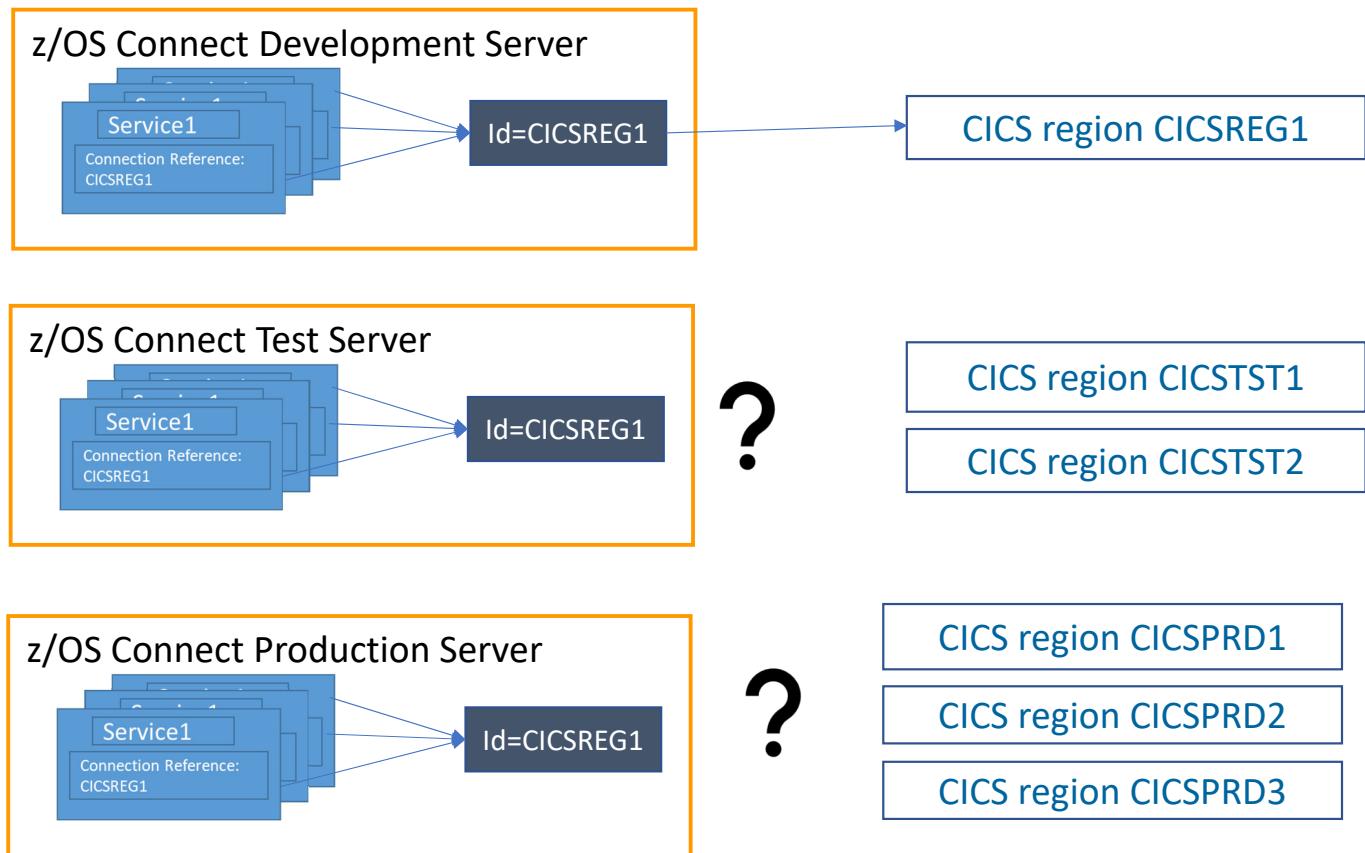
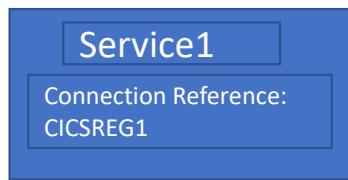
## Connection Reference Consideration

Carefully consider the names used for connections



## Use naming conventions for service/endpoint connection references (OpenAPI 2)

Don't couple service and API requester connection names to specific systems or endpoints





## What REST test tooling is available?



# Testing an API with Postman

The screenshot shows the Postman application interface. At the top, there's a navigation bar with File, Edit, View, Help, Home, Workspaces, Reports, Explore, and a search bar. Below the navigation is a toolbar with icons for cloud, collection, settings, and notifications, along with an Upgrade button.

The main workspace displays a GET request to <https://mpz3.washington.ibm.com:9443/cscvinc/employee/111111>. The request has 10 Headers. The response status is 200 OK, with a time of 205 ms and a size of 899 B. The response body is shown in JSON format:

```
1  "cscvincSelectServiceOperationResponse": {  
2      "cscvincContainer": {  
3          "response": {  
4              "CEIBRESP": 0,  
5              "CEIBRESP2": 0,  
6              "USERID": "CICSUSER",  
7              "filea": {  
8                  "employeeNumber": "111111",  
9                  "name": "C. BAKER",  
10                 "address": "OTTAWA, ONTARIO",  
11                 "phoneNumber": "51212003",  
12                 "date": "26 11 81",  
13                 "amount": "00011 00"  
14             }  
15         }  
16     }  
17 }
```

At the bottom, there are buttons for Find and Replace, Console, Bootcamp, Runner, and Trash.

mitchj@us.ibm.com

<https://www.postman.com/downloads/>



# Testing an API with the API Explorer (zCEE V3.0.48)

IBM

all Filter

## Liberty REST APIs

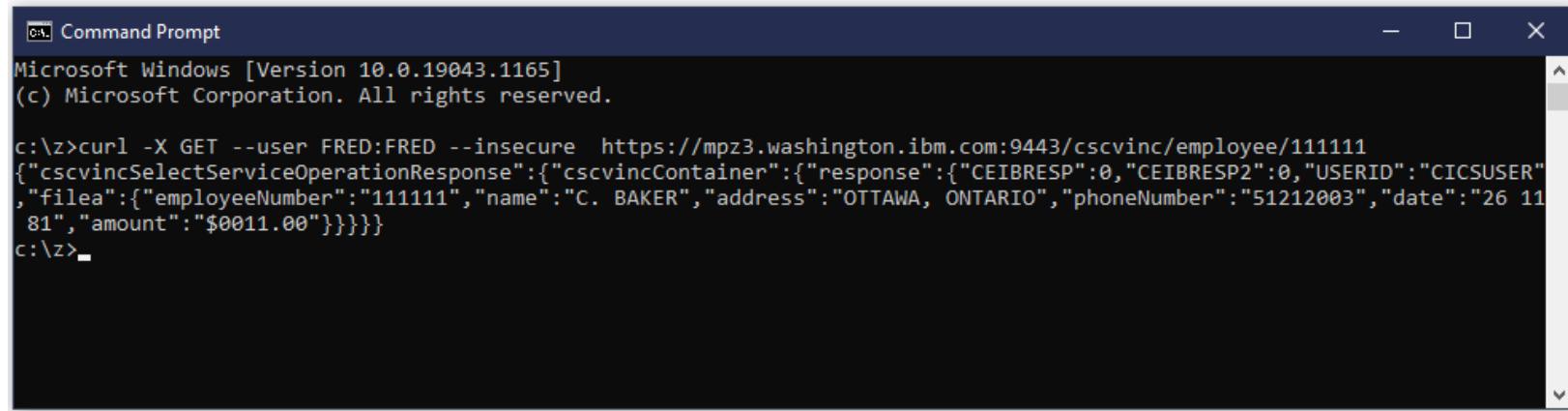
Discover REST APIs available within Liberty

| POST          | /cscvinc/employee            | Show/Hide   List Operations   Expand |
|---------------|------------------------------|--------------------------------------|
| DELETE        | /cscvinc/employee/{employee} |                                      |
| GET           | /cscvinc/employee/{employee} | Show/Hide   List Operations   Expand |
| PUT           | /cscvinc/employee/{employee} |                                      |
| db2employee   |                              |                                      |
| filemgr       |                              |                                      |
| imsPhoneBook  |                              |                                      |
| jwtIvpDemoApi |                              |                                      |
| miniloancics  |                              |                                      |
| mqapi         |                              |                                      |
| phonebook     |                              |                                      |

File Edit View History Bookmarks Tools Help REST API Documentation + https://mpz3.washington.ibm.com:9443/api/explorer/#/cscvinc/employee/111111 Try it out! Hide Response Curl curl -X GET --header 'Accept: application/json' --header 'Authorization: Basic RnJlZDpmcmVk' 'https://mpz3.washington.ibm.com:9443/cscvinc/employee/111111' Request URL https://mpz3.washington.ibm.com:9443/cscvinc/employee/111111 Response Body { "cscvincSelectServiceOperationResponse": { "cscvincContainer": { "response": { "CEIBRESP": 0, "CEIBRESP2": 0, "USERID": "CICCSUSER", "file": { "employeeNumber": "111111", "name": "C. BAKER", "address": "OTTAWA, ONTARIO", "phoneNumber": "51212003", "date": "26 11 81", "amount": "\$0011.00" } } } } } Response Code 200 Response Headers { "content-language": "en-US", "content-length": "269", "content-type": "application/json; charset=UTF-8" }



# Testing an API using the cURL tool



```
Command Prompt
Microsoft Windows [Version 10.0.19043.1165]
(c) Microsoft Corporation. All rights reserved.

c:\z>curl -X GET --user FRED --insecure https://mpz3.washington.ibm.com:9443/cscvinc/employee/111111
{"cscvincSelectServiceOperationResponse":{"cscvincContainer":{"response":{"CEIBRESP":0,"CEIBRESP2":0,"USERID":"CICSUSER","filea":{"employeeNumber":"111111","name":"C. BAKER","address":"OTTAWA, ONTARIO","phoneNumber":"51212003","date":"26 11 81","amount":"$0011.00"}}}}
c:\z>
```

<https://curl.se/download.html>



## z/OS Connect administration API

Interface providing meta-data and life-cycle operations for z/OS Connect services, APIs and API requesters.

### APIs : Operations for working with APIs

Show/Hide | List Operations | Expand Operations

|        |                 |   |
|--------|-----------------|---|
| GET    | /apis           | Returns a list of all the deployed z/OS Connect APIs  |
| POST   | /apis           | Deploys a new API into z/OS Connect                   |
| DELETE | /apis/{apiName} | Undeploys an API from z/OS Connect                    |
| GET    | /apis/{apiName} | Returns detailed information about a z/OS Connect API |
| PUT    | /apis/{apiName} | Updates an existing z/OS Connect API                  |

### Services : Operations for working with services

Show/Hide | List Operations | Expand Operations

|        |   |   |
|--------|---|---|
| GET    | /services                                   | Returns a list of all the deployed z/OS Connect services          |
| POST   | /services                                   | Deploys a new service into z/OS Connect                           |
| DELETE | /services/{serviceName}                     | Undeploys a service from z/OS Connect                             |
| GET    | /services/{serviceName}                     | Returns detailed information about a z/OS Connect service         |
| PUT    | /services/{serviceName}                     | Updates an existing z/OS Connect service                          |
| GET    | /services/{serviceName}/schema/{schemaType} | Returns the request or response schema for a z/OS Connect service |

### API Requesters : Operations that work with API Requesters.

Show/Hide | List Operations | Expand Operations

|        |                                   |  |
|--------|-----------------------------------|--|
| GET    | /apiRequesters                    | Returns a list of all the deployed z/OS Connect API Requesters                 |
| POST   | /apiRequesters                    | Deploys a new API Requester into z/OS Connect and invoke an API Requester call |
| DELETE | /apiRequesters/{apiRequesterName} | Undeploys an API Requester from z/OS Connect                                   |
| GET    | /apiRequesters/{apiRequesterName} | Returns the detailed information about a z/OS Connect API Requester            |
| PUT    | /apiRequesters/{apiRequesterName} | Updates an existing z/OS Connect API Requester                                 |



# Examples of curl in JCL (Rocket Software's tooling) and commands files

```
*****  
/* SET SYMBOLS  
*****  
//EXPORT EXPORT SYMLIST=(*)  
// SET CURL= '/usr/lpp/rocket/curl'  
*****  
/* CURL Procedure  
*****  
//CURL PROC  
//CURL EXEC PGM=IKJEFT01,REGION=0M  
//SYSTSPRT DD SYSOUT=*  
//SYSERR DD SYSOUT=*  
//STDOUT DD SYSOUT=*  
// PEND  
*****  
/* STEP CURL - use curl to deploy API cscvinc  
*****  
//DEPLOY EXEC CURL  
BPXBATCH SH export CURL=&CURL; +  
$CURL/bin/curl -X PUT -s +  
--cacert /u/johnson/CERTAUTH.PEM --user FRED:FRED +  
https://wg31.washington.ibm.com:9445/zosConnect/apis/cscvinc?status=stoped > null; +  
$CURL/bin/curl -X DELETE -s +  
--cacert /u/johnson/CERTAUTH.PEM --user FRED:FRED +  
https://wg31.washington.ibm.com:9445/zosConnect/apis/cscvinc > null; +  
$CURL/bin/curl -X POST -s +  
--cacert /u/johnson/CERTAUTH.PEM --user FRED:FRED +  
--data-binary @/u/johnson/cscvinc.aar +  
--header "Content-Type: application/zip" +  
https://wg31.washington.ibm.com:9445/zosConnect/apis  
*****  
/* STEP CURL - use curl to invoke the API cscvinc  
*****  
//INVOKE EXEC CURL  
//SYSTSIN DD *,SYMBOLS=EXECSYS  
BPXBATCH SH export CURL=&CURL; $CURL/bin/curl -X GET -s +  
--cacert /u/johnson/CERTAUTH.PEM --user FRED:FRED +  
https://wg31.washington.ibm.com:9445/cscvinc/employee/000100
```

mitchj@us.ibm.com

[https://www.rocketsoftware.com/platforms/ibm-z\(curl-for-zos](https://www.rocketsoftware.com/platforms/ibm-z(curl-for-zos)

```
@echo off  
set TARGET=wg31.washington.ibm.com  
set FILE=cscvinc_1.0.0  
curl -X PUT --user user1:user1 --insecure  
https://%TARGET%:9483/zosConnect/apiRequesters/%FILE%?status=stopped  
curl -X DELETE --user user1:user1 --insecure  
https://%TARGET%:9483/zosConnect/apiRequesters/%FILE%  
curl -X POST --user user1:user1 --data-binary @%FILE%.ara --header  
"Content-Type: application/zip" --insecure  
https://%TARGET%:9483/zosConnect/apiRequesters  
echo ''
```

Change the status of API to stopped

Delete or remove the API from the server

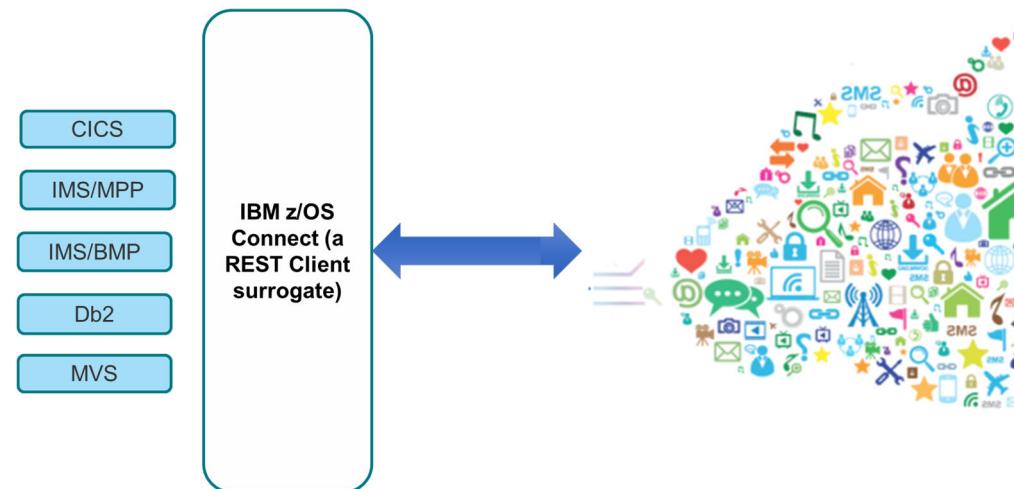
Deploy the API to the server

Execute the API



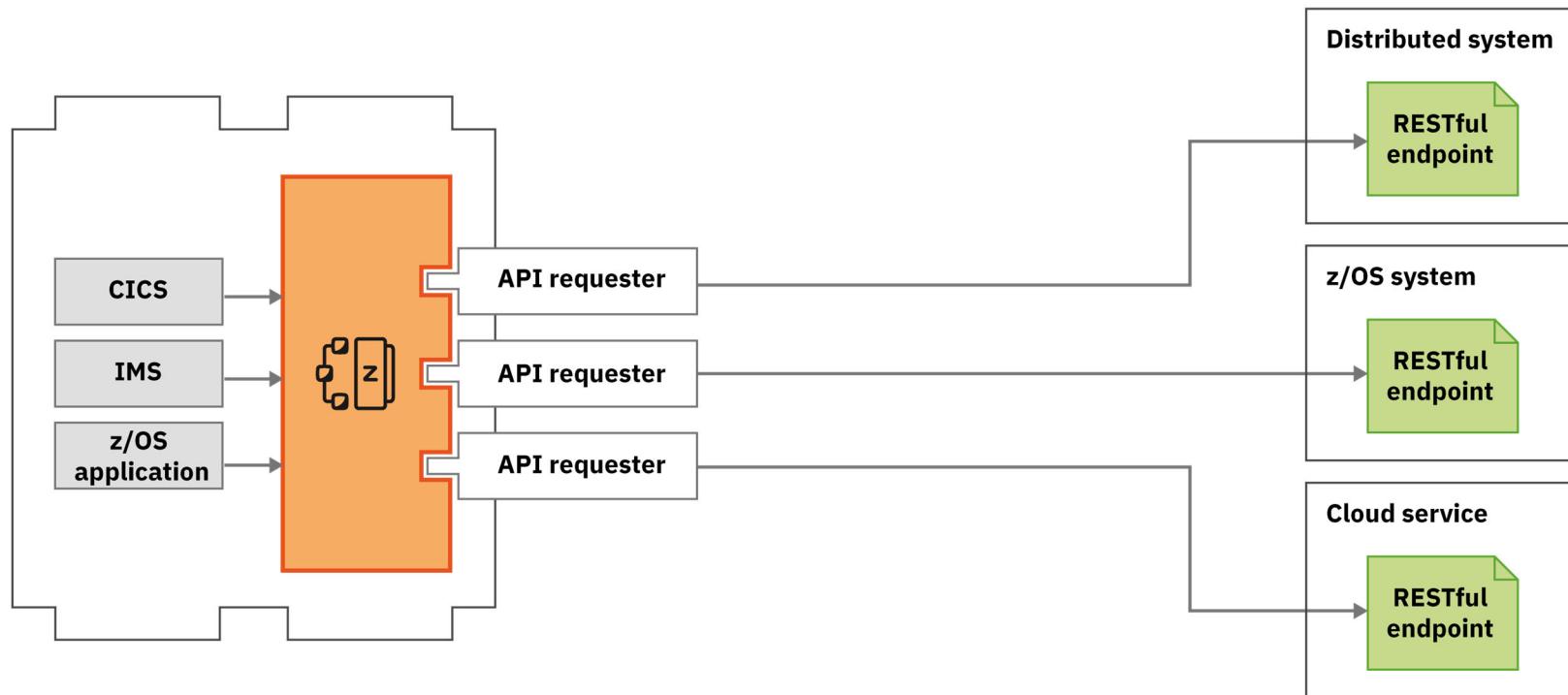
## /api\_toolkit/apiRequesters

Quick and easy API mapping.





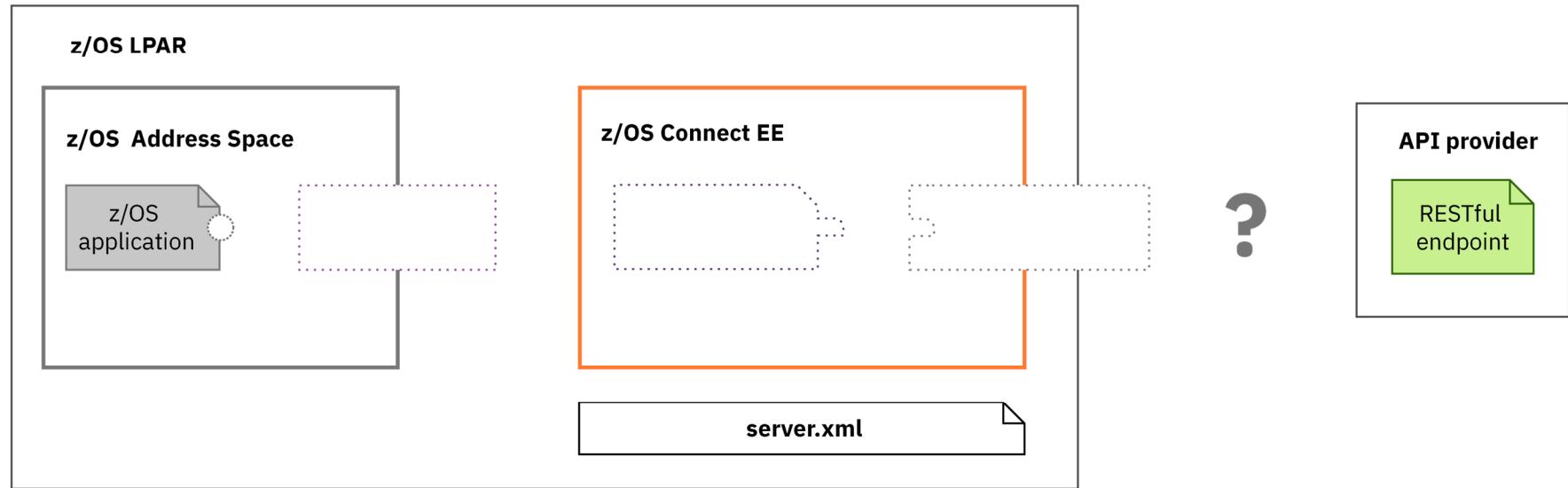
# Use API requester to call external APIs from z/OS assets





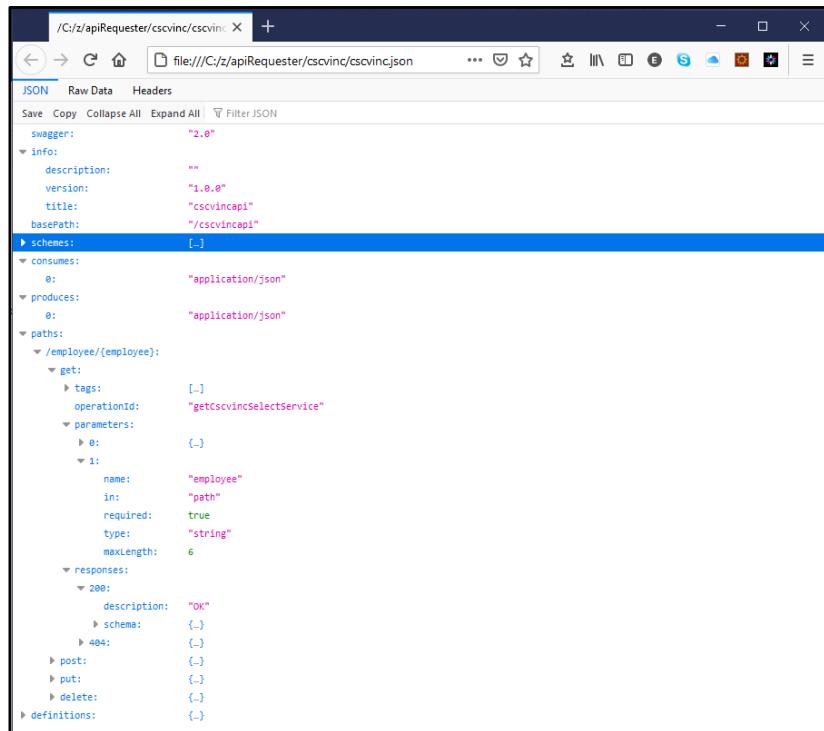
# Steps to calling an external API

Starting point



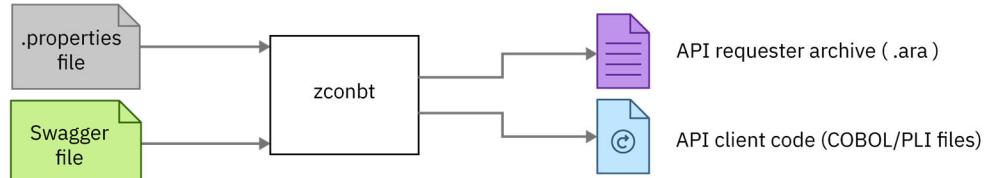
# Calling an external API requires the API's specification document

Start with the API's specification and generate the API requester archive file and API client code



The screenshot shows a JSON editor window displaying the contents of the cscvinc.json file. The file defines an API endpoint for employees, including parameters, responses, and definitions.

```
swagger: "2.0"
info:
  description: ""
  version: "1.0.0"
  title: "cscvincapi"
  basePath: "/cscvincapi"
schemes: []
consumes:
  @: "application/json"
produces:
  @: "application/json"
paths:
  /employee/{employee}:
    get:
      tags:
        @: []
      operationId: "getCscvincSelectService"
      parameters:
        @:
          @: {}
        1:
          name: "employee"
          in: "path"
          required: true
          type: "string"
          maxLength: 6
      responses:
        200:
          description: "OK"
          schema: {}
        404:
      post:
      put:
      delete:
    definitions: {}
```



## properties file#

```
apiDescriptionFile=./cscvinc.json
dataStructuresLocation=./syslib
apiInfoFileLocation=./syslib
logFileDirectory=./logs
language=COBOL
connectionRef=cscvincAPI
requesterPrefix=csc
```

#Additional property file attributes, e.g., `defaultCharacterMaxLength`, `defaultArrayMaxItems`, etc. are described at **The build toolkit properties file** article at URL <https://www.ibm.com/docs/en/zosconnect/3.0?topic=toolkit-build-properties-file>



# The number of specific entries can be ambiguous

The COBOL copy book will include a counter variable (*variable-num*) for each variable whose number of occurrences is ambiguously defined in the specification document. The number of occurrences of these variables must be provided.

```
wg31 base
File Edit Settings View Communication Actions Window Help
Menu Utilities Compilers Help
BROWSE    USER1.ZCEE.SOURCE(PUTAPI) - 01.05      Line 000000062 Col 001 080
Command ==>                               Scroll ==> PAGE
MAINLINE SECTION.

*-----
* Common code
*-----
* initialize working storage variables
  INITIALIZE PUT-REQUEST.
  INITIALIZE PUT-RESPONSE.

*-
* Set up the data for the API Requester call
*-----
  MOVE 1 to cscvincUpdateServiceOp-num in PUT-REQUEST
  request2-num in PUT-REQUEST
  filea2-num in PUT-REQUEST
  name-num in PUT-REQUEST
  Xaddress-num in PUT-REQUEST
  phoneNumber-num in PUT-REQUEST
  Xdate-num in PUT-REQUEST
  amount-num in PUT-REQUEST.

  MOVE num of PARM-DATA TO employee IN PUT-REQUEST.
  MOVE LENGTH of employee in PUT-REQUEST to
        employee-length IN PUT-REQUEST.

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

MA C
Connected to remote server/host wg31z using lu/pool TCP00108 and port 23
04 / 015
```



# Steps to calling an external API

Use the z/OS Connect build toolkit, e.g., `zconbt`, to generate an API requester archive file and at most, 3 copy books per method found in the Swagger

```
zconbt.bat -p=./cscvinc.properties -f=./cscvinc.ara
BAQB0000I: z/OS Connect Enterprise Edition 3.0 Build Toolkit Version 1.5 (20210816-0926).
BAQB0008I: Creating API requester archive from configuration file ./cscvinc.properties.
BAQB0040I: The generated API requester is automatically named cscvincapi_1.0.0 based on the title cscvincapi and version 1.0.0 of the API to be called.
.
.
.
Total 4 operation(s) (success: 4, ignored: 0) defined in api description file: ./cscvinc.json
----- Successfully processed operation(s) -----
operationId: getCsvincSelectService, basePath: /cscvincapi, relativePath: /employee/{employee}, method: GET
- request data structure : CSC00Q01
- response data structure : CSC00P01
- api info file : CSC00I01

operationId: putCsvincUpdateService, basePath: /cscvincapi, relativePath: /employee/{employee}, method: PUT
- request data structure : CSC01Q01
- response data structure : CSC01P01
- api info file : CSC01I01

operationId: postCsvincInsertService, basePath: /cscvincapi, relativePath: /employee/{employee}, method: POST
- request data structure : CSC02Q01
- response data structure : CSC02P01
- api info file : CSC02I01

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

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

# BTW, the z/OS Connect Build Toolkit can be executed on z/OS



```
//JOHNSONS JOB (ACCOUNT),JOHNSON,NOTIFY=&SYSUID,REGION=0M,  
// CLASS=A,MSGCLASS=H,MSGLEVEL=(1,1)  
//*****  
///* SET SYMBOLS  
//*****  
//EXPORT EXPORT SYMLIST=(*  
// SET WORKDIR='u/johnson/zconbt'  
// SET ZCONDIR='/usr/lpp/IBM/zosconnect/v3r0/zconbt/bin'  
//ZCONBT EXEC PGM=IKJEFT01,REGION=0M,MEMLIMIT=4G  
//SYSTSPPRT DD SYSOUT=*  
//SYSERR DD SYSOUT=*  
//STDOUT DD SYSOUT=*  
//SYSTSIN DD *,SYMBOLS=EXECSYS  
BPXBATCH SH +  
  export WORKDIR=&WORKDIR; +  
  export ZCONDIR=&ZCONDIR; +  
  cd $WORKDIR; +  
  $ZCONDIR/zconbt.zos -p cscvinc.properties -f=cscvinc.ara; +  
  cp -v $WORKDIR/syslib/* //'JOHNSON.ZCONBT.COPYLIB'"
```

## cscvinc.properties

```
apiDescriptionFile=./cscvinc.json  
dataStructuresLocation=./syslib  
apiInfoFileLocation=./syslib  
logFileDirectory=./logs  
language=COBOL  
connectionRef=cscvincAPI  
requesterPrefix=csc
```

This assumes the zconbt.zip files was expanded into directory /usr/lpp/IBM/zosconnect/v3r0/zconbt using command *jar -tf zconbt.zip* and that the property file and Swagger JSON document are encoded in ASCII in directory /u/johnson/zconbt.



# Tech-Tip: Copy books naming convention

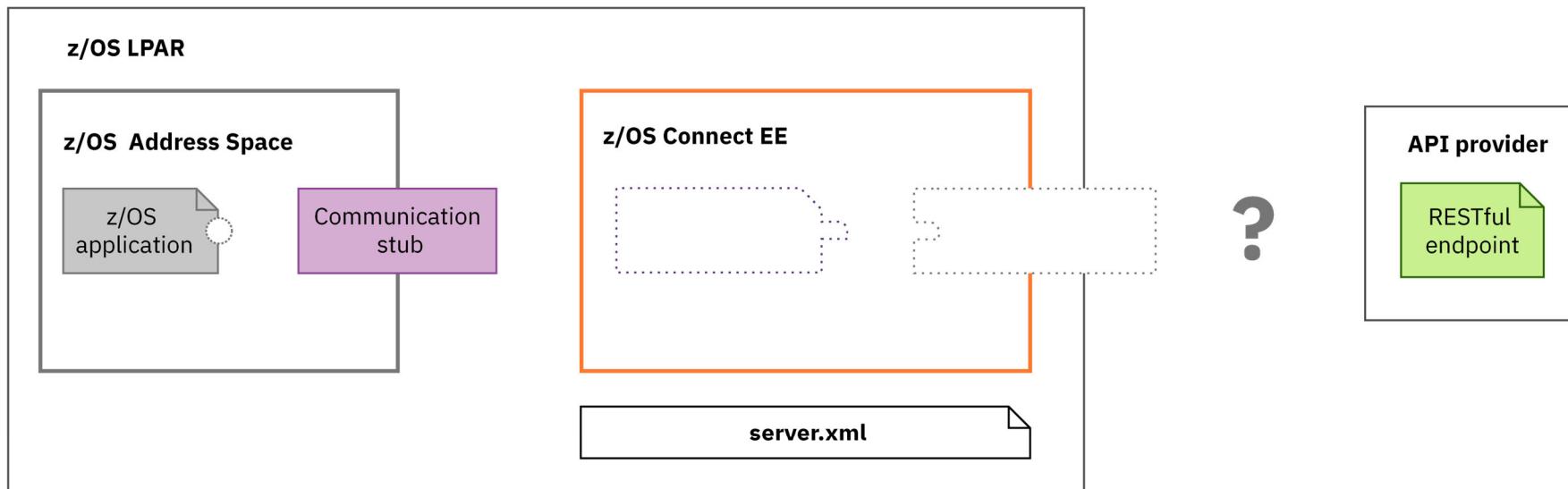
- The naming convention for the generated COBOL copy books is based on the *requesterPrefix* value specified in the properties file. That value was set to CSC in this case, e.g., CSC#####. The next 2 characters in the name are assigned sequentially as each API and method is processed, e.g., CSC00### and CSC01###, and CSC02###.
- The next character will be either a Q, P or an I. A “Q” for a **request** copy book, the “P” for a **response** copy book and the “I” for the copy book which contains **information**, e.g., method, path name etc. derived from the Swagger document
- Up to three copy books are generated for each method of each API found in the Swagger document.
  - In the previous example, there were 4 APIs with each having 1 method for a total of 12 copy books.
  - If there is no request message or no response message, then no copy book will be generated. But the messages stills need to have addressable storage in the application's working area.

```
* Request and Response
 01 GET-REQUEST.
    10 FILLER
      PIC X(1).
 01 GET-RESPONSE.
    COPY MQ000P01 SUPPRESS.
* Structure with the API information
 01 GET-INFO-OPER1.
    COPY MQ000I01 SUPPRESS.
```



# Steps for involving an external API from a COBOL program

Update the application by adding the generated copy books, a common BAQRINFO copy book and a call to communication stub



Configure a communication stub.

- For CICS region systems using URIMAP resources
- For non CICS client the configuration is done via environment variables

```
*-----*
* Call the communication stub
*-----*
*-----*
* Call the subsystem-supplied stub code to send
* API request to zCEE
    CALL COMM-STUB-PGM-NAME USING
        BY REFERENCE    GET-INFO-OPER1
        BY REFERENCE    BAQ-REQUEST-INFO
        BY REFERENCE    BAQ-REQUEST-PTR
        BY REFERENCE    BAQ-REQUEST-LEN
        BY REFERENCE    BAQ-RESPONSE-INFO
        BY REFERENCE    BAQ-RESPONSE-PTR
        BY REFERENCE    BAQ-RESPONSE-LEN
    END-CALL
    RETURN CODE 01 NO-ERRORS
```



# Steps to calling an external API

Include the generated copy books in a COBOL program

```
GETAPI X
  *--> ERROR MESSAGE STRUCTURE
  01  ERROR-MSG.
    03  EM-ORIGIN          PIC X(8)  VALUE SPACES.
    03  EM-CODE            PIC S9(9) COMP-5 SYNC VALUE 0.
    03  EM-DETAIL          PIC X(1024) VALUE SPACES.

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

  * Request and Response
  01 API-REQUEST.
    COPY CSC02Q01.
  01 API_RESPONSE.
    COPY CSC02P01.

  * Structure with the API information
  01 API-INFO-OPER1.
    COPY CSC02I01.

  * Request and Response segment used to store request and
    ***
```

## API-REQUEST

```
CSC00I01  CSC00Q01 X
  * JSON schema keyword 'minLength' value: '0'.
  * JSON schema keyword 'maxLength' value: '6'.
  * This field contains a varying length array of characters or
  * binary data.
  *      09 employee-length          PIC S9999 COMP-5 SYNC.
  *      09 employee                PIC X(6).
  *
  * ++++++
  06 ReqPathParameters.
    09 employee-length          PIC S9999 COMP-5 SYNC.
    09 employee                PIC X(6).
```

## API-RESPONSE

```
CSC00I01  CSC00Q01  CSC00P01 X
  * ++++++
  06 RespBody.
    09 cscvincap0-num    PIC S9(9) COMP-5 SYNC.
    09 cscvincap0-operatio.
      12 Container1.

    15 RESPONSE-CONTAINER2-num  PIC S9(9) COMP-5
      SYNC.
```

## API-INFO-OPER1

```
CSC00I01 X
  03 BAQ-APINAME          PIC X(255)
    VALUE 'cscvincapi_1.0.0'.
  03 BAQ-APINAME-LEN        PIC S9(9) COMP-5 SYNC
    VALUE 16.
  03 BAQ-APIPATH           PIC X(255)
    VALUE '%2Fcvincap%2Femployee%2F%7Bemployee%7D'.
  03 BAQ-APIPATH-LEN        PIC S9(9) COMP-5 SYNC
    VALUE 41.
  03 BAQ-APIMETHOD          PIC X(255)
    VALUE 'GET'.
  03 BAQ-APIMETHOD-LEN        PIC S9(9) COMP-5 SYNC
    VALUE 3.
```



# Steps to calling an external API

Add a call to the communication stub use pointers to pass working storage addresses of the copy books

The diagram illustrates the steps to calling an external API. It shows a main program (GETAPI) and three copy books (CSC00101, CSC00Q01, CSC00P01) with their respective definitions.

**Main Program (GETAPI):**

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

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

* Call the communication stub
*
* Call the subsystem-supplied stub code to send
* API request to zCEE
CALL COMM-STUB-PGM-NAME USING
BY REFERENCE API-INFO-OPER1
BY REFERENCE BAQ-REQUEST-INFO
BY REFERENCE BAQ-REQUEST-PTR
BY REFERENCE BAQ-REQUEST-LEN
BY REFERENCE BAQ-RESPONSE-INFO
BY REFERENCE BAQ-RESPONSE-PTR
BY REFERENCE BAQ-RESPONSE-LEN.

* The BAQ-RETURN-CODE field in 'BAQRINFO' indicates whether this
* API request was successful.
```

**Copy Book Definitions:**

- CSC00101:**

```
03 BAQ-APINAME          PIC X(255)
      VALUE 'cscvincap1_1.0.0'.
03 BAQ-APINAME-LEN      PIC S9(9) COMP-5 SYNC
      VALUE 16.
03 BAQ-APIPATH          PIC X(255)
      VALUE 'S2fcsvincap1%2Femployee%7D'.
03 BAQ-APIPATH-LEN      PIC S9(9) COMP-5 SYNC
      VALUE 41.
03 BAQ-APIMETHOD        PIC X(255)
      VALUE 'GET'.
03 BAQ-APIMETHOD-LEN    PIC S9(9) COMP-5 SYNC
      VALUE 3.
```
- CSC00Q01:**

```
* JSON schema keyword 'minLength' value: '0'.
* JSON schema keyword 'maxLength' value: '6'.
* This field contains a varying length array of characters or
* binary data.
*   09 employee-length      PIC S9999 COMP-5 SYNC.
*   09 employee              PIC X(6).
*
* ++++++
06 ReqPathParameters.
09 employee-length      PIC S9999 COMP-5 SYNC.
09 employee              PIC X(6).
```
- CSC00P01:**

```
* ++++++
06 RespBody.
09 cscvincSelectServiceOp-num  PIC S9(9) COMP-5 SYNC.
09 cscvincSelectServiceOperatio.
12 Container1.
15 RESPONSE-CONTAINER2-num  PIC S9(9) COMP-5
      SYNC.
```



# Steps to calling an external API

Accessing the results that have been stored in working storage

```

GETAPI X
BY REFERENCE BAQ-RESPONSE-LEN.
* The BAQ-RETURN-CODE field in 'BAQRINFO' indicates whether this
* API call is successful.

* When BAQ-RETURN-CODE is 'BAQ-SUCCESS', response is
* successfully returned and fields in RESPONSE copybook
* can be obtained. Display the translation result.
IF BAQ-SUCCESS THEN
    DISPLAY "NUMB: " Numb2 of API_RESPONSE
    DISPLAY "NAME: " name2 of API_RESPONSE
    DISPLAY "ADDRX: " addrx2 of API_RESPONSE
    DISPLAY "PHONE: " phone2 of API_RESPONSE
    DISPLAY "DATEX: " datex2 of API_RESPONSE
    DISPLAY "AMOUNT: " amount2 of API_RESPONSE
    MOVE CEIBRESP of API_RESPONSE to EIBRESP
    MOVE CEIBRESP2 of API_RESPONSE to EIBRESP2
    DISPLAY "EIBRESP: " EIBRESP
    DISPLAY "EIBRESP2: " EIBRESP2
    DISPLAY "HTTP CODE: " BAQ-STATUS-CODE

* Otherwise, some error happened in API, z/OS Connect EE server
* or communication stub. 'BAQ-STATUS-CODE' and
* 'BAQ-STATUS-MESSAGE' contain the detailed information
* of this error.
ELSE
    DISPLAY "Error code: " BAQ-STATUS-CODE
    DISPLAY "Error msg: " BAQ-STATUS-MESSAGE
    MOVE BAQ-STATUS-CODE TO EM-CODE
    MOVE BAQ-STATUS-MESSAGE TO EM-DETAIL
    EVALUATE TRUE
* When error happens in API, BAQ-RETURN-CODE is BAQ-ERROR-IN-API.
* BAQ-STATUS-CODE is the HTTP response code of API.
    LINES BAQ-ERROR-IN-API

```

The diagram illustrates the flow of data from AS/400 assembly code to z/OS command output. A red box highlights the logic for successful API calls, which is then connected by a red arrow to a screenshot of the mpz3 terminal window showing the 'BAQRINFO' copybook definition. Another red box highlights the error handling logic, which is connected by a red arrow to the same terminal window showing the actual command output for an error case.

**mpz3 Terminal Window Output:**

```

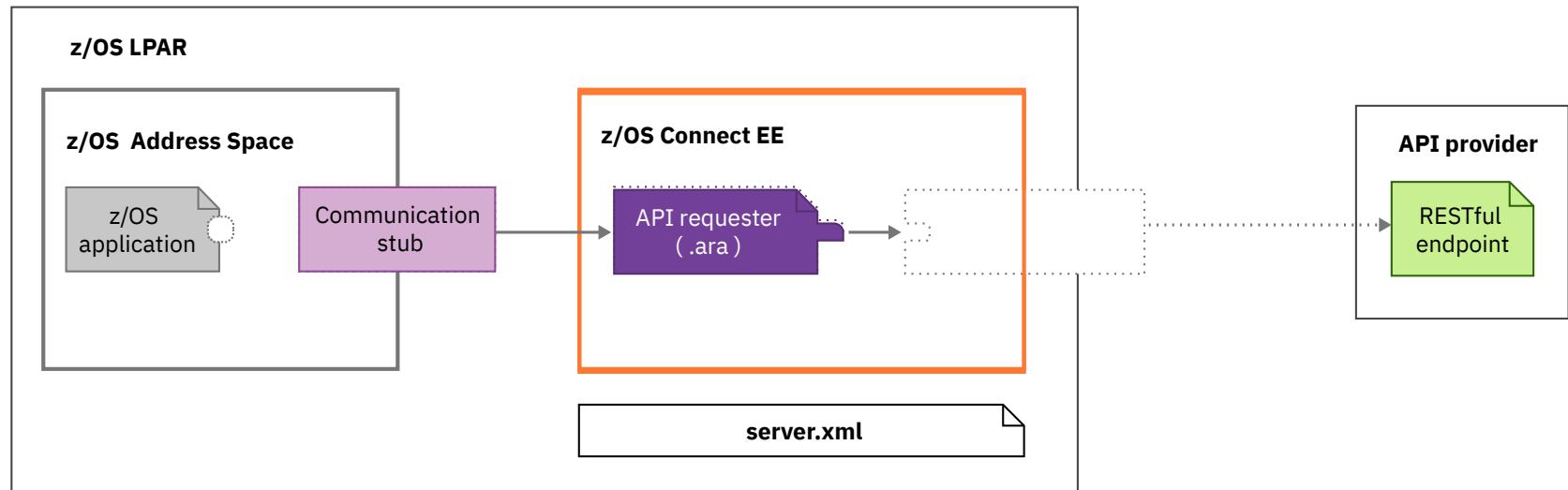
File Edit Settings View Communication Actions Window Help
Menu Utilities Compilers Help
BROWSE ZCEE30.SBAQC0B(BAQRINFO)
Command ==>
Line 000000066 Col 001 080
Scroll ==> PAGE
01 BAQ-RESPONSE-INFO.
03 BAQ-RESPONSE-INFO-COMP-LEVEL PIC S9(9) COMP-5 SYNC VALUE 0.
03 BAQ-STUB-NAME PIC X(8).
03 BAQ-RETURN-CODE PIC S9(9) COMP-5 SYNC.
     88 BAQ-SUCCESS VALUE 0.
     88 BAQ-ERROR-IN-API VALUE 1.
     88 BAQ-ERROR-IN-ZCEE VALUE 2.
     88 BAQ-ERROR-IN-STUB VALUE 3.
     88 BAQ-ERROR-NO-RESPONSE VALUE 4.
03 BAQ-STATUS-CODE PIC S9(9) COMP-5 SYNC.
03 BAQ-STATUS-MESSAGE PIC X(1024).
03 BAQ-STATUS-MESSAGE-LEN PIC S9(9) COMP-5 SYNC.
***** Bottom of Data *****
18/058
Connected to remote server/host mpz3 using lu/pool MPZ30021 and port 23

```



# Steps to calling an external API

Deploy the API requester (.ara) archive

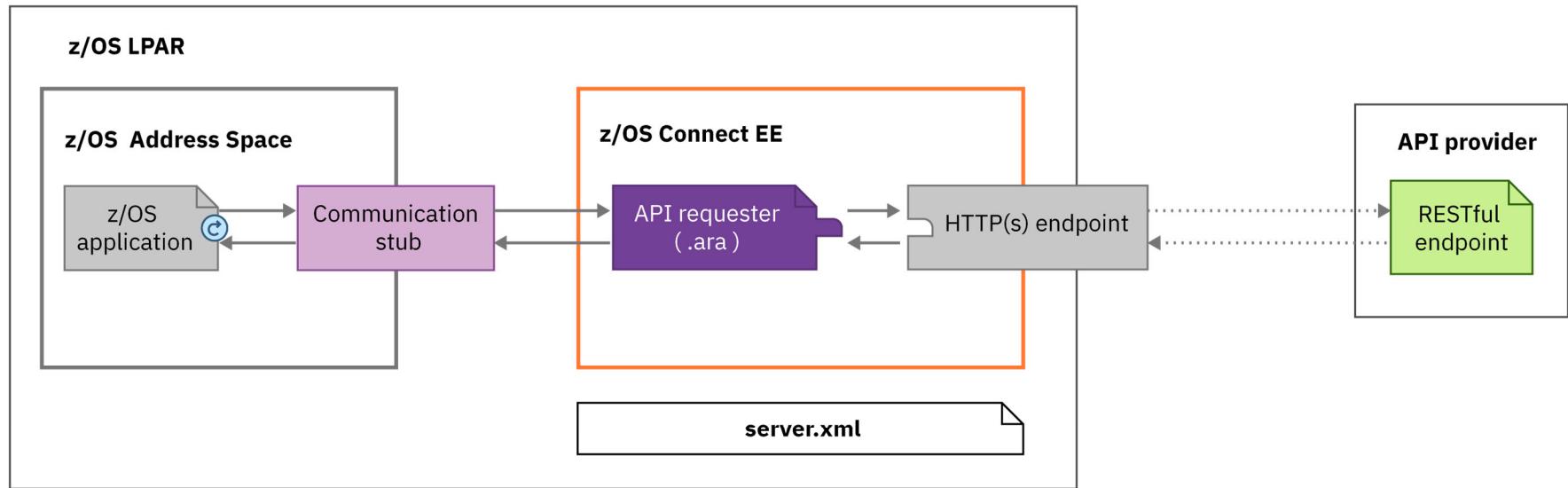


Deploy your API requester archive to the *apiRequesters* directory.



# Steps to calling an external API

Done



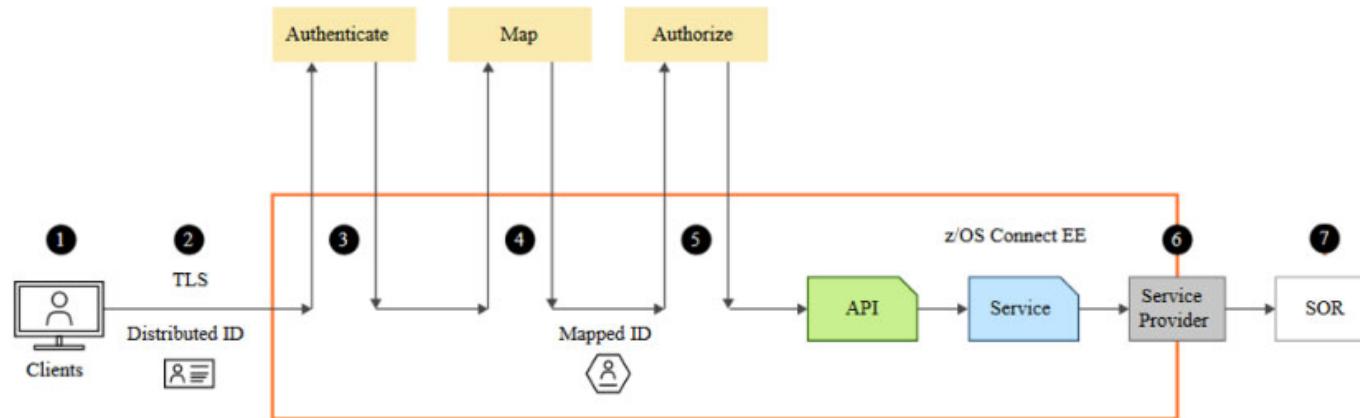


## /security

How is security implement?



# Typical z/OS Connect EE API Provider security flow



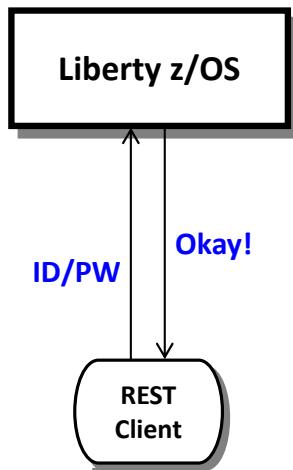
1. The credentials provided by the client
2. Secure the connection to the z/OS Connect EE server
3. Authenticate the client. This can be within the z/OS Connect EE server or by requesting verification from a third-party server
4. Map the authenticated identity to a user ID in the user registry
5. Authorize the mapped user ID to connect to z/OS Connect EE and optionally authorize user to invoke actions on APIs
6. Secure the connection to the System of Record (SoR) and provide security credentials to be used to invoke the program or to access the data resource
7. The program or database request may run in the SoR under the mapped ID



# API Provider Authentication

Several different ways this can be accomplished:

## Basic Authentication



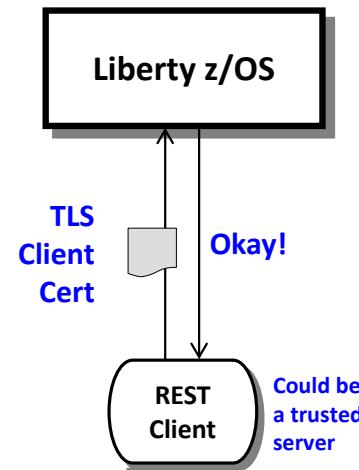
Server prompts for ID/PW

Client supplies ID/PW or  
ID/Passticket

Server checks registry:

- Basic (server.xml)
- LDAP
- SAF

## Client Certificate



Server prompts for cert.

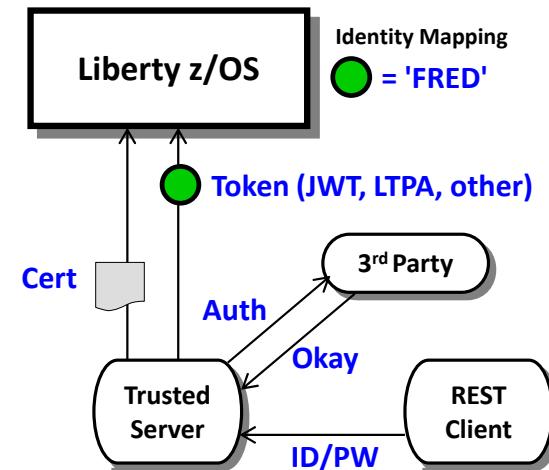
Client supplies certificate

Server validates cert and  
maps to an identity

Registry options:

- LDAP
- SAF

## Third Party Authentication



Client authenticates to 3<sup>rd</sup> party sever

Client receives a trusted 3<sup>rd</sup> party token

Token flows to Liberty z/OS and is  
mapped to an identity

Registry options:

- LDAP
- SAF



# OPENAPI 3 roles

```
File Edit Format View Help
openapi: 3.0.1
info:
  title: cscvinc
  description: ""
  version: 1.0.0
servers:
- url: /cscvinc
  x-ibm-zcon-roles-allowed:
    - Manager
paths:
  /employee:
    post:
      tags:
        - cscvinc
      operationId: postCscvincInsertService
      x-ibm-zcon-roles-allowed:
        - Staff
      parameters:
        - name: Authorization
          in: header
          schema:
            type: string
      requestBody:
        description: request body
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/postCscvincInsertService_request'
            required: true
      responses:
        200:
          description: OK
          content:
            application/json:
              schema:
                $ref: '#/components/schemas/postCscvincInsertService_response_200'
            x-codegen-request-body-name: postCscvincInsertService_request
      /employee/{employee}:
        get:
          tags:
            - cscvinc
          operationId: getCscvincSelectService
          x-ibm-zcon-roles-allowed:
            - Staff
          parameters:
            - name: Authorization
              in: header
              schema:
                type: string
Ln 44, Col 16 100% Unix (LF) UTF-8
```

```
<!-- Enable features -->
<featureManager>
  <feature>appSecurity-2.0</feature>
</featureManager>
<webAppSecurity allowFailOverToBasicAuth="true" />

<basicRegistry id="basic" realm="zosConnect">
  <user name="Fred" password="fredpwd" />
  <user name="user1" password="user1" />
  <user name="user2" password="user2" />
  <user name="user3" password="user3" />
  <group name="Manager">
    <member name="Fred"/>
  </group>
  <group name="Staff">
    <member name="Fred"/>
    <member name="user1"/>
    <member name="user2"/>
  </group>
</basicRegistry>

<authorization-roles id="Manager">
  <security-role name="Manager">
    <group name="managerGroup"/>
  </security-role>
</authorization-roles>
<authorization-roles id="Staff">
  <security-role name="Staff">
    <group name="staffGroup"/>
  </security-role>
</authorization-roles>
```



# Third Party Authentication Examples

The image displays two side-by-side screenshots of web pages illustrating third-party authentication.

**Left Screenshot: UPS Sign Up**

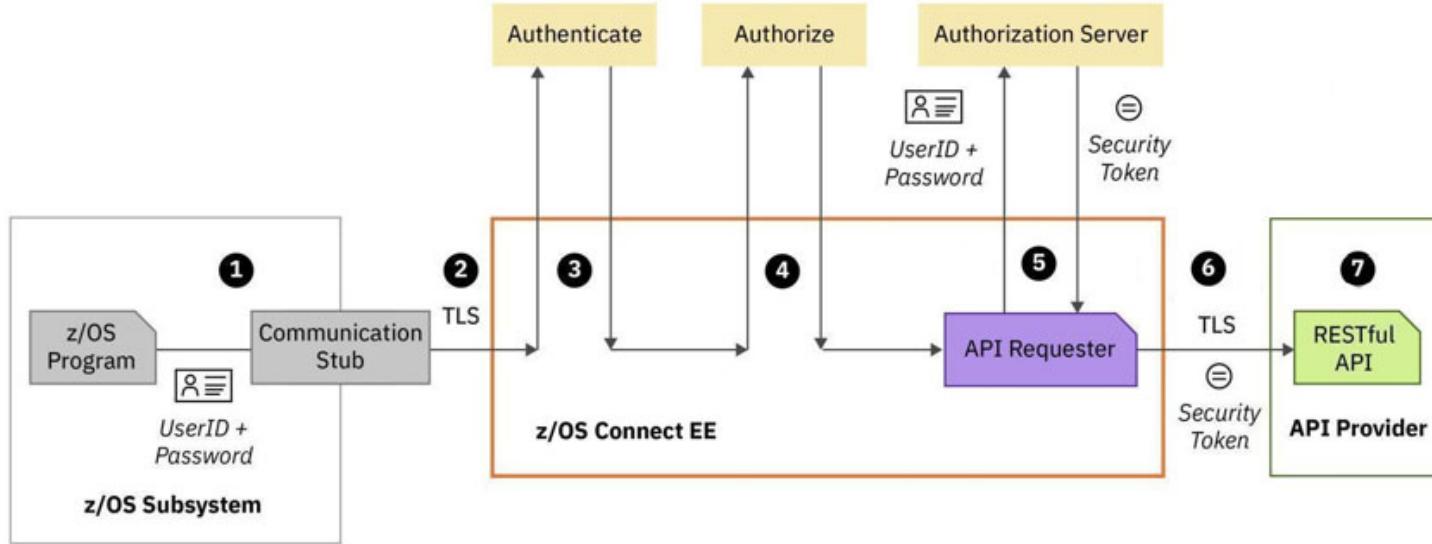
This screenshot shows the UPS Sign Up page. At the top, there's a banner stating "UPS is open for business: Service impacts related to Coronavirus ...More". Below the banner, the UPS logo is displayed. A "Sign Up" button is prominent. Below it, a link for "Already have an ID? Log in" is shown. A section titled "Use one of these sites." lists social media integration options: Google, Facebook, Amazon, Apple, and Twitter. Below these, fields for "Name \*", "Email \*", "User ID \*", "Password \*", and "Phone" are provided. The "Password \*" field includes a "Show" link. A "Feedback" button is located on the right side of the form.

**Right Screenshot: myNCDMV Log In**

This screenshot shows the myNCDMV Log In page. It features a background image of autumn foliage. At the top, there are "Log In" and "Sign Up" buttons. The "Log In" button is highlighted. Below the buttons, there are fields for "Email Address" (with placeholder "name@example.com") and "Password" (with placeholder "\*\*\*\*\*"). A "Remember Me" checkbox is available. Below the password field are "Log In" and "Forgot Password" buttons. Further down, there are three social login options: "Continue with Apple", "Continue with Facebook", and "Continue with Google". A "Continue as Guest" link is also present. A notice at the bottom states: "NOTICE FOR PUBLIC COMPUTER USERS - If you sign in with Google, Apple, or Facebook you are also signing into that account on this computer. Remember to sign out when you're done." The page is powered by "payit".



# Typical z/OS Connect EE API Requester security flow



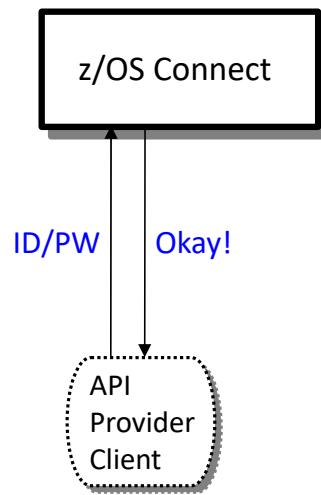
1. A user ID and password can be used for basic authentication by the z/OS Connect EE server
2. Connection between the CICS, IMS, or z/OS application and the z/OS Connect EE server can use TLS
3. Authenticate the CICS, IMS, or z/OS application.
4. Authorize the authenticated user ID to connect to z/OS Connect EE and to perform specific actions on z/OS Connect EE API requesters
5. Pass the user ID and password credentials to an authorization server to obtain a security token.
6. Secure the connection to the external API provider, and provide security credentials such as a security token to be used to invoke the RESTful API
7. The RESTful API runs in the external API provider



# z/OS Application to z/OS Connect API Requester

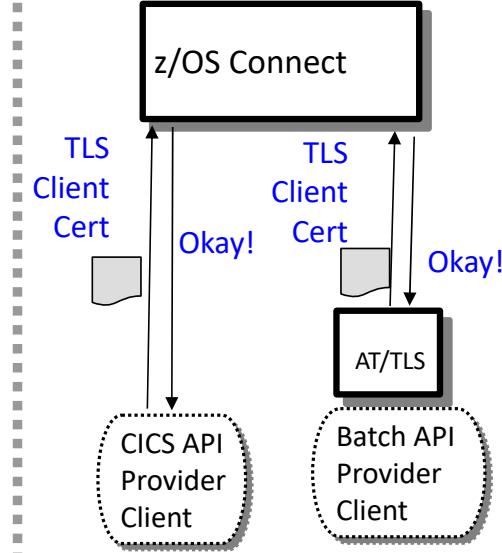
Two options for providing credentials for authentication

## Basic Authentication



**Application provides  
ID/PW or ID/PassTicket**

## Client Certificate



**z/OS Connect requests a  
client certificate**

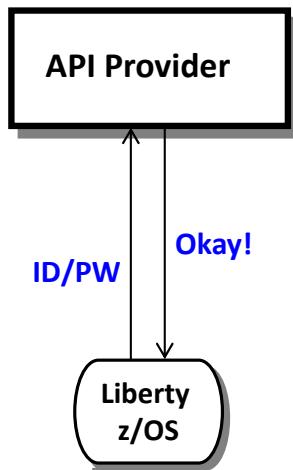
**CICS or AT/TLS supplies a  
client certificate**



# API Requester - API Provider Authentication

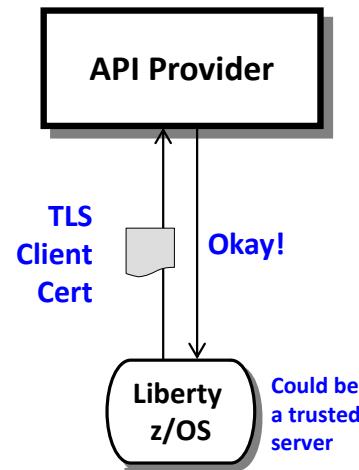
Several different ways this can be accomplished:

## Basic Authentication



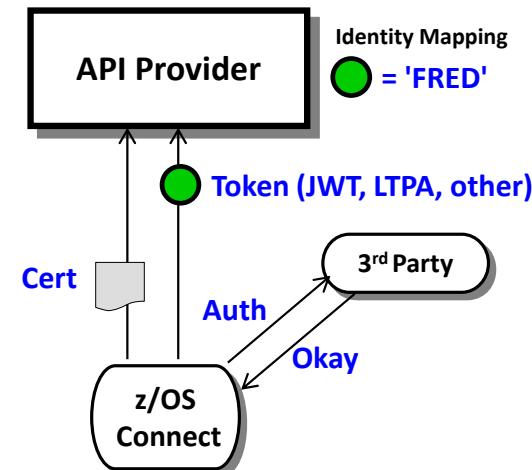
zCEE supplies ID/PW or  
ID/Passticket

## Client Certificate



Server prompts for certificate  
zCEE supplies certificate

## Third Party Authentication



zCEE authenticates to 3<sup>rd</sup> party sever  
zCEE receives a trusted 3<sup>rd</sup> party token  
Token flows to API Provider



## Basic authentication – non-CICS COBOL API Requester

- ❑ A MVS batch, IMS or Db2 stored procedure requester application sends basic authentication information (identity and password) by using environment variables.
  - BAQUSERNAME
  - BAQPASSWORD
- ❑ The variables can be provided in JCL using CEEOPTS DD statement:

```
//CEELOPTS DD *  
  POSIX(ON),  
  ENVAR("BAQURI=wg31.washington.ibm.com",  
"BAQPORT=9080",  
"BAQUSERNAME=USER1",  
"BAQPASSWORD=USER1")
```

- ❑ Or, provided by using a CEEROPT or CEEUOPT module:

```
CEEROPT CSECT  
CEEROPT AMODE ANY  
CEEROPT RMODE ANY  
CEEXOPT POSIX=((ON),OVR),  
  ENVAR=((('BAQURI=wg31.washington.ibm.com',  
'BAQPORT=9120',  
'BAQUSERNAME=USER1',  
'BAQPASSWORD=USER1'),OVR),  
  RPTOPTS=((ON),OVR)  
END
```

**Tech/Tip: This is good opportunity to use a pass ticket rather than a password**



# Tech/Tip: API Requester - HTTP v HTTPS

MVS Batch and IMS with and without an outbound AT-TLS policy

```
CEE0PTS DD *
  POSIX(ON),
  ENVAR("BAQURI=wg31.washington.ibm.com",
  "BAQPORT=9080")
```

```
CEE0PTS DD *
  POSIX(ON),
  ENVAR("BAQURI=wg31.washington.ibm.com",
  "BAQPORT=9443")
```

## CICS URIMAPS

```
WG31
File Edit Settings View Communication Actions Window Help
OVERTYPE TO MODIFY
CEDA ALTER Urimap( BAQURIMP )
Urimap      : BAQURIMP
Group       : SYSGRP
DESCription ==> URIMAP for z/OS Connect EE server
Status      ==> Enabled   Enabled | Disabled
USAge       ==> Client    Server | Client | Pipeline | Atom
              | Jvmserver
UNIVERSAL RESOURCE IDENTIFIER
SCHEME     ==> HTTP      HTTP | HTTPS
POrt       ==> 09120    No | 1-65535
HOST       ==> wg31.washington.ibm.com
            ==
PATH       ==> /
(Mixed Case) ==>
            ==
            ==
            ==
+ OUTBOUND CONNECTION POOLING
SYSID=CICS APPL
PF 1 HELP 2 COM 3 END      6 CRSR 7 SBH 8 SFH 9 MSG 10 SB 11
MBI C
Connected to remote server/host wg31 using lu/pool TCP00133 and port 23
```

```
CICS RELEASE = 0710
File Edit Settings View Communication Actions Window Help
OVERTYPE TO MODIFY
CEDA ALTER Urimap( BAQURIMP )
Urimap      : BAQURIMP
Group       : SYSGRP
DESCription ==> URIMAP for z/OS Connect EE server
Status      ==> Enabled   Enabled | Disabled
USAge       ==> Client    Server | Client | Pipeline | Atom
              | Jvmserver
UNIVERSAL RESOURCE IDENTIFIER
SCHEME     ==> HTTPS     HTTP | HTTPS
POrt       ==> 09443    No | 1-65535
HOST       ==> wg31.washington.ibm.com
            ==
PATH       ==> /
(Mixed Case) ==>
            ==
            ==
            ==
+ OUTBOUND CONNECTION POOLING
SYSID=CICS APPLID=CICS53Z
PF 1 HELP 2 COM 3 END      6 CRSR 7 SBH 8 SFH 9 MSG 10 SB 11 SF 12 CNCL
MBI C
Connected to remote server/host wg31 using lu/pool TCP00133 and port 23
13/022
```

Field BAQ-ZCON-SERVER-URI was added to BAQRINFO in V3.0.37.

MOVE "URIMAP01" TO BAQ-ZCON-SERVER-URI.



# Tech/Tip: API Requester - HTTP v HTTPS

MVS Batch and IMS with and without an outbound AT-TLS policy

```
CEE0PTS DD *
  POSIX(ON),
  ENVAR("BAQURI=wg31.washington.ibm.com",
  "BAQPORT=9080")
```

```
CEE0PTS DD *
  POSIX(ON),
  ENVAR("BAQURI=wg31.washington.ibm.com",
  "BAQPORT=9443")
```

## CICS URIMAPS

```
WG31
File Edit Settings View Communication Actions Window Help
OVERTYPE TO MODIFY
CEDA ALTER Urimap( BAQURIMP )
Urimap      : BAQURIMP
Group       : SYSGRP
DESCription ==> URIMAP for z/OS Connect EE server
Status      ==> Enabled   Enabled | Disabled
USAge       ==> Client    Server | Client | Pipeline | Atom
              | Jvmserver
UNIVERSAL RESOURCE IDENTIFIER
SCHEME     ==> HTTP      HTTP | HTTPS
POrt       ==> 09120    No | 1-65535
HOST       ==> wg31.washington.ibm.com
            ==
PATH       ==> /
(Mixed Case) ==>
            ==
            ==
            ==
+ OUTBOUND CONNECTION POOLING
SYSID=CICS APPL
PF 1 HELP 2 COM 3 END      6 CRSR 7 SBH 8 SFH 9 MSG 10 SB 11
MBI C
Connected to remote server/host wg31 using lu/pool TCP00133 and port 23
```

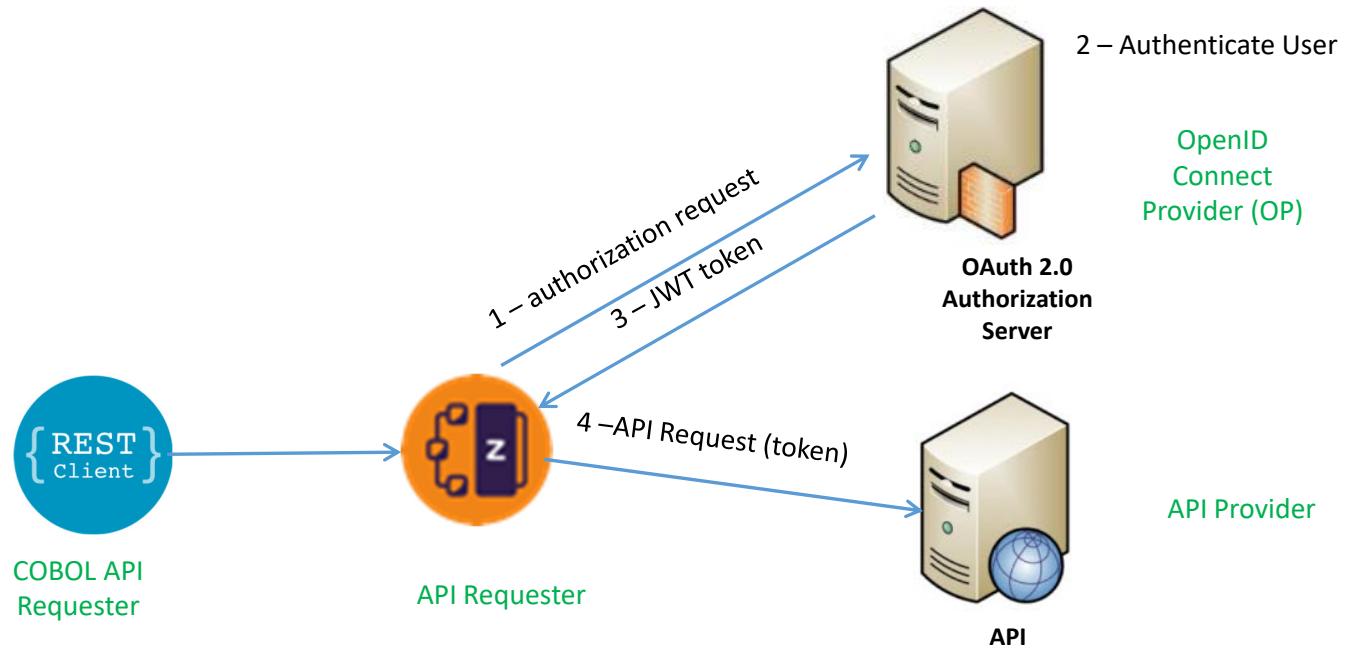
  

```
CICS RELEASE = 0710
File Edit Settings View Communication Actions Window Help
OVERTYPE TO MODIFY
CEDA ALTER Urimap( BAQURIMP )
Urimap      : BAQURIMP
Group       : SYSGRP
DESCription ==> URIMAP for z/OS Connect EE server
Status      ==> Enabled   Enabled | Disabled
USAge       ==> Client    Server | Client | Pipeline | Atom
              | Jvmserver
UNIVERSAL RESOURCE IDENTIFIER
SCHEME     ==> HTTPS     HTTP | HTTPS
POrt       ==> 09443    No | 1-65535
HOST       ==> wg31.washington.ibm.com
            ==
PATH       ==> /
(Mixed Case) ==>
            ==
            ==
            ==
+ OUTBOUND CONNECTION POOLING
SYSID=CICS APPLID=CICS53Z
PF 1 HELP 2 COM 3 END      6 CRSR 7 SBH 8 SFH 9 MSG 10 SB 11 SF 12 CNCL
MBI C
Connected to remote server/host wg31 using lu/pool TCP00133 and port 23
13/022
```

Field BAQ-ZCON-SERVER-URI was added to BAQRINFO in V3.0.37.

MOVE "URIMAP01" TO BAQ-ZCON-SERVER-URI.

# z/OS Connect OAuth Flow for API requester



## Grant Types:

- client\_credentials
- password

# Configuring OAuth support – BAQRINFO copy book



```
File Edit Settings View Communication Actions Window Help
Menu Utilities Compilers Help
BROWSE ZCEE30.SBAQCOB(BAQRINFO) Line 0000000028 Col 001 080
Command ==> Scroll ==> PAGE
01 BAQ-REQUEST-INFO.
03 BAQ-REQUEST-INFO-COMP-LEVEL PIC S9(9) COMP-5 SYNC VALUE 4.
03 BAQ-REQUEST-TINFO-USER.
05 BAQ-DAUTH.
07 BAQ-DAUTH-USERNAME PIC X(256).
07 BAQ-DAUTH-USERNAME-LEN PIC S9(9) COMP-5 SYNC
    VALUE 0.
07 BAQ-DAUTH-PASSWORD PIC X(256).
07 BAQ-DAUTH-PASSWORD-LEN PIC S9(9) COMP-5 SYNC
    VALUE A.
07 BAQ-DAUTH-CLIENTID PIC X(256).
07 BAQ-DAUTH-CLIENTID-LEN PIC S9(9) COMP-5 SYNC
    VALUE 0.
07 BAQ-DAUTH-CLIENT-SECRET PIC X(256).
07 BAQ-DAUTH-CLIENT-SECRET-LEN PIC S9(9) COMP-5 SYNC
    VALUE A.
07 BAQ-DAUTH-SCOPE-PTR USAGE POINTER.
07 BAQ-DAUTH-SCOPE-LEN PIC S9(9) COMP-5 SYNC
    VALUE 0.
05 BAQ-AUTHTOKEN.
07 BAQ-TOKEN-USERNAME PIC X(256).
07 BAQ-TOKEN-USERNAME-LEN PIC S9(9) COMP-5 SYNC
    VALUE 0.
07 BAQ-TOKEN-PASSWORD PIC X(256).
07 BAQ-TOKEN-PASSWORD-LEN PIC S9(9) COMP-5 SYNC
    VALUE 0.
05 BAQ-ZCON-SERVER-URI PIC X(256)
    VALUE SPACES.
MA A 04/015
Connected to remote server/host wg31z using lu/pool TCP00145
```

**Grant Type: *password*** - The identity of the user provided by the CICS, IMS, or z/OS application, or it might be another entity.  
Client\_credentials can be supplied by the program or in the server XML configuration.

**Grant Type: *client\_credentials*** - the identity associated with the combination of the CICS, IMS, or z/OS application, and the z/OS Connect EE server that calls the RESTful API on behalf of the CICS, IMS, or z/OS application

**Scope is always required.**

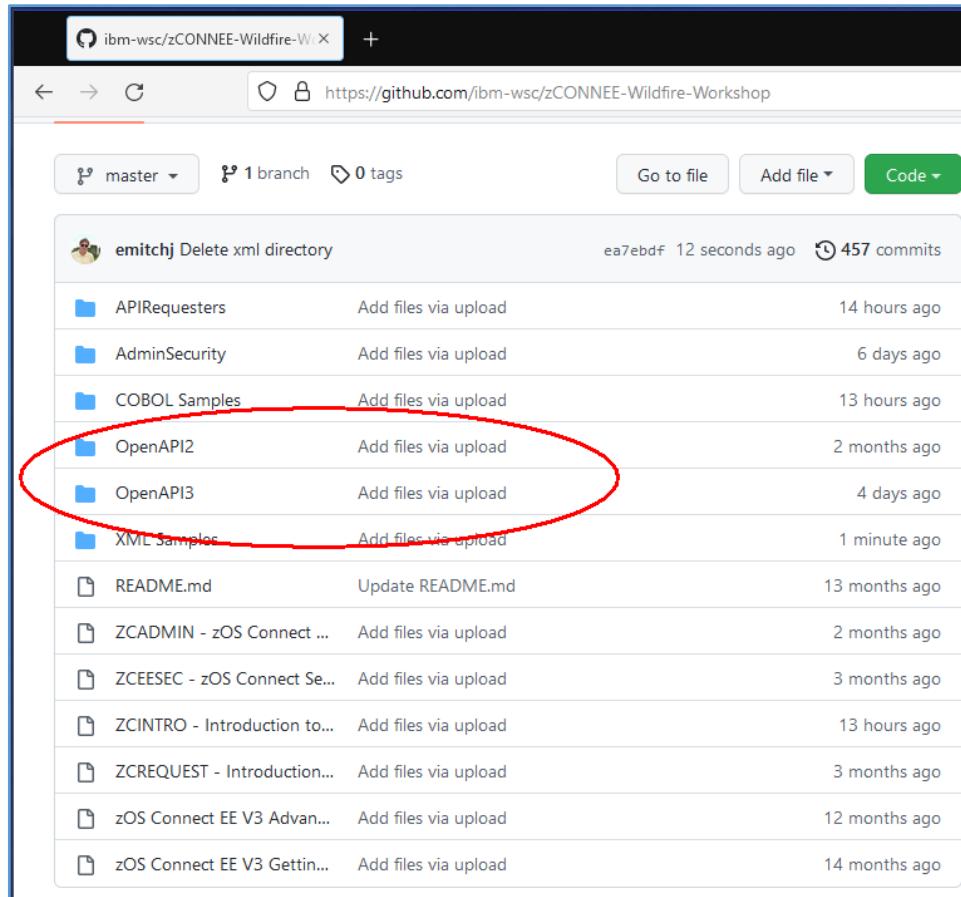
| OAuth 2.0 specification entity | password | client_credentials | Where Set                    |
|--------------------------------|----------|--------------------|------------------------------|
| Client ID                      | required | Required           | server.xml or by application |
| Client Secret                  | optional | Required           | server.xml or by application |
| Username                       | required | N/A                | by application               |
| Password                       | required | N/A                | by application               |

# Agenda

- An Introduction and Overview of using REST API
- Enabling RESTful API to various z/OS resources, e.g.
  - CICS
  - Db2
  - IMS/TM
  - IMS/DB
  - MQ
  - Outbound REST APIs
- Accessing RESTful API from z/OS COBOL Applications
- A brief overview of z/OS Connect Security\*

\*For more on security, contact your local IBM rep regarding the schedule of workshop *zOSSEC1 IBM z/OS Connect Administration/Security Wildfire Workshop*

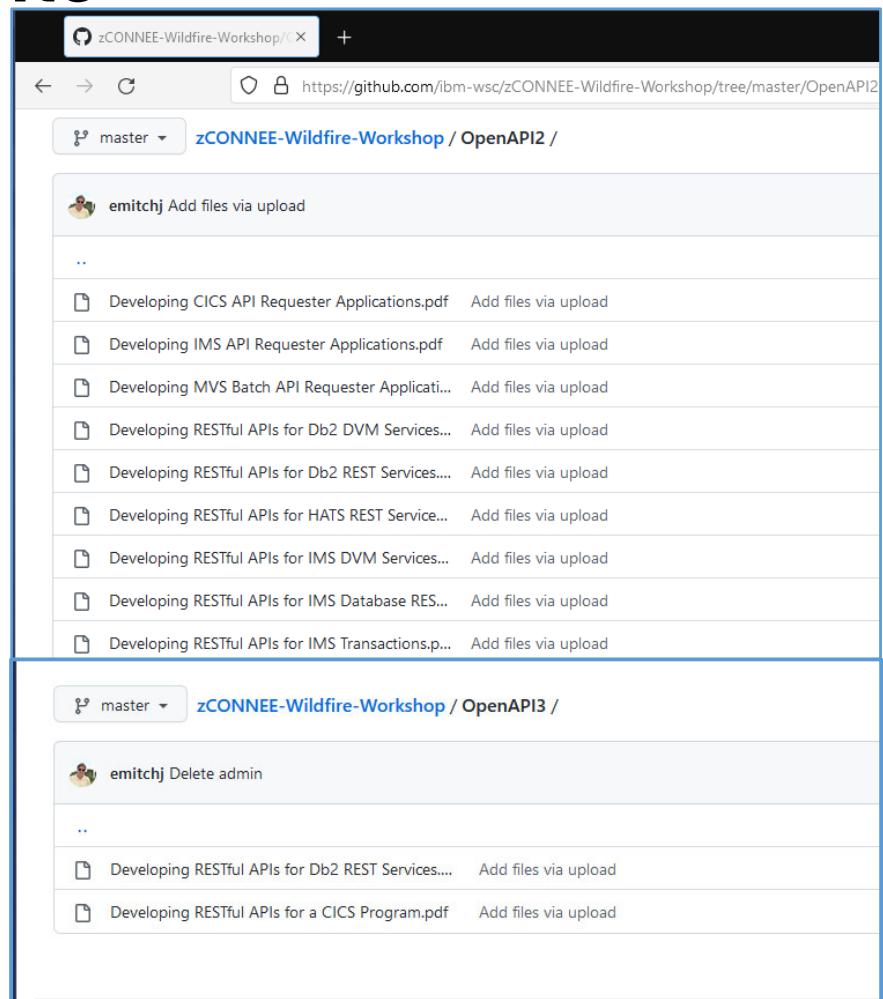
# z/OS Connect Wildfire Github Site



A screenshot of a GitHub repository page. The repository name is 'ibm-wsc/zCONNEE-Wildfire-Workshop'. The commit list shows several entries:

- emitchj Delete xml directory ea7ebdf 12 seconds ago 457 commits
- APIRequesters Add files via upload 14 hours ago
- AdminSecurity Add files via upload 6 days ago
- COBOL Samples Add files via upload 13 hours ago
- OpenAPI2** Add files via upload 2 months ago
- OpenAPI3** Add files via upload 4 days ago
- XML Samples Add files via upload 1 minute ago
- README.md Update README.md 13 months ago
- ZCADMIN - zOS Connect ... Add files via upload 2 months ago
- ZCEESEC - zOS Connect Se... Add files via upload 3 months ago
- ZCINTRO - Introduction to... Add files via upload 13 hours ago
- ZCREQUEST - Introduction... Add files via upload 3 months ago
- zOS Connect EE V3 Advan... Add files via upload 12 months ago
- zOS Connect EE V3 Gettin... Add files via upload 14 months ago

<https://ibm.biz/zCEEWorshopMaterial>



A screenshot of a GitHub repository page for 'zCONNEE-Wildfire-Workshop'. The repository has two main branches: 'OpenAPI2' and 'OpenAPI3'.

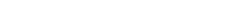
- OpenAPI2 /**
  - emitchj Add files via upload
  - Developing CICS API Requester Applications.pdf
  - Developing IMS API Requester Applications.pdf
  - Developing MVS Batch API Requester Application...
  - Developing RESTful APIs for Db2 DVM Services...
  - Developing RESTful APIs for Db2 REST Services....
  - Developing RESTful APIs for HATS REST Service...
  - Developing RESTful APIs for IMS DVM Services...
  - Developing RESTful APIs for IMS Database RES...
  - Developing RESTful APIs for IMS Transactions.p...
- OpenAPI3 /**
  - emitchj Delete admin
  - Developing RESTful APIs for Db2 REST Services....
  - Developing RESTful APIs for a CICS Program.pdf

mitchj@us.ibm.com

- Contact your IBM representative to schedule access to these exercises

WSC wants your  
feedback!

## What you will see:

**From:** IBM Client Feedback <[ibm@feedback.ibm.com](mailto:ibm@feedback.ibm.com)>   
**Subject:** Got a minute? Two questions on your IBM Z Washington Systems Center experience

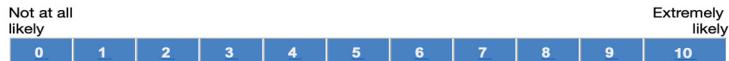


Dear

Thank you for engaging with our team. At IBM Z Washington Systems Center, we make it a priority to listen to our clients and want to continuously improve your experience. So, we would love your candid feedback on how we are doing. Please take a moment to answer two short questions about your experience.

You can begin the survey by answering this question.

## **How likely are you to recommend IBM Z Washington Systems Center to others?**



Sincerely,

IBM Advocacy Team

**\*\*you will NOT receive a new survey if you already responded to an IBM Survey from Medallia in the last **60 days** OR if you haven't responded within the last **30 days**\*\***



Thank you for listening and your questions

**And thank you for completing the Medallia survey**