



# ZCINTRO - IBM z/OS Connect

## An introduction and overview

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

John Brefach [John.J.Brefach@ibm.com](mailto:John.J.Brefach@ibm.com)

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
- 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

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

© 2017, 2023 IBM Corporation  
Slide 4



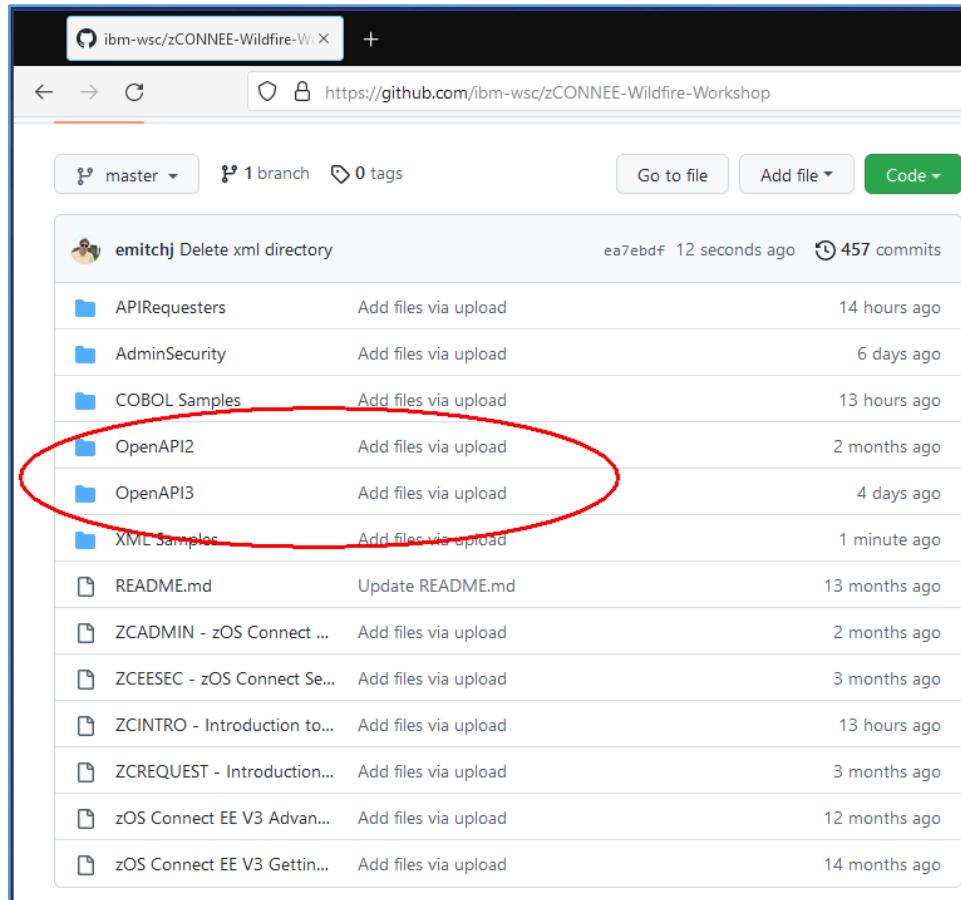
## ZCADMIN – IBM z/OS Connect Administration

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



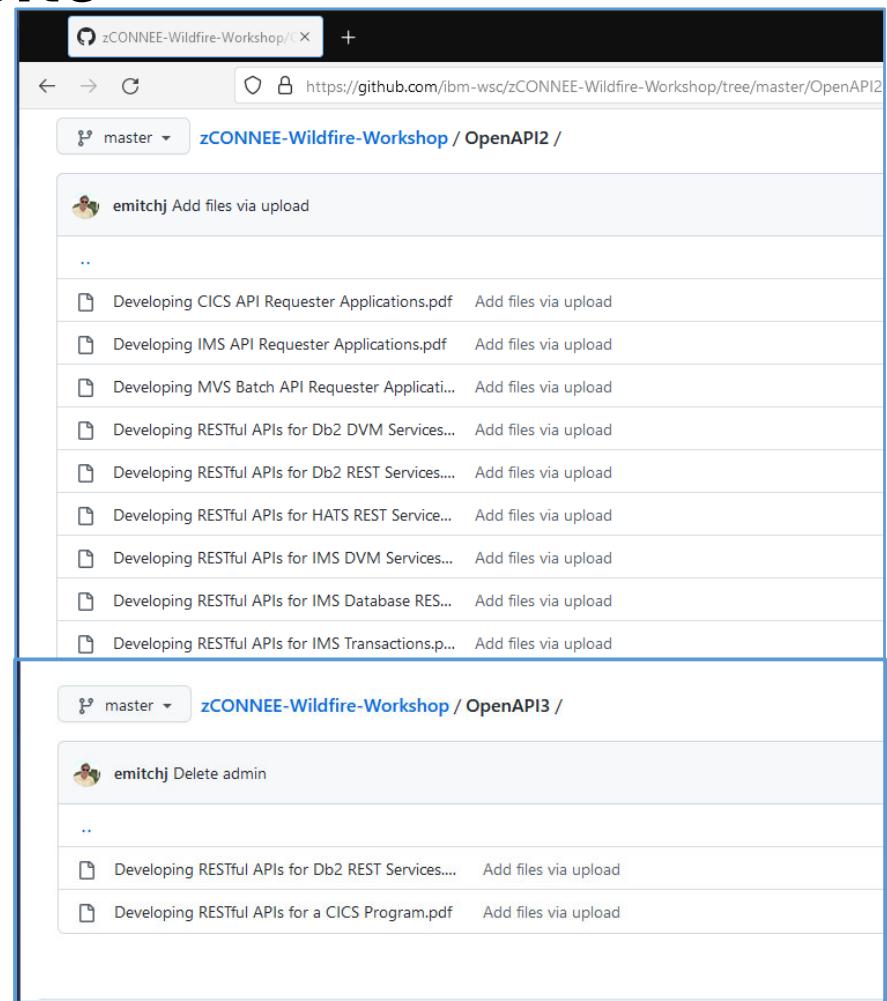
A screenshot of a GitHub repository page. The repository name is 'ibm-wsc/zCONNEE-Wildfire-Workshop'. The commit list shows several entries:

- 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

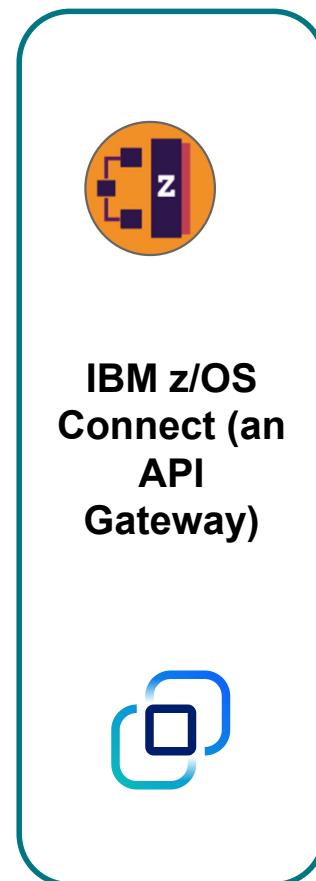
<https://ibm.biz/zCEEWorshopMaterial>



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**

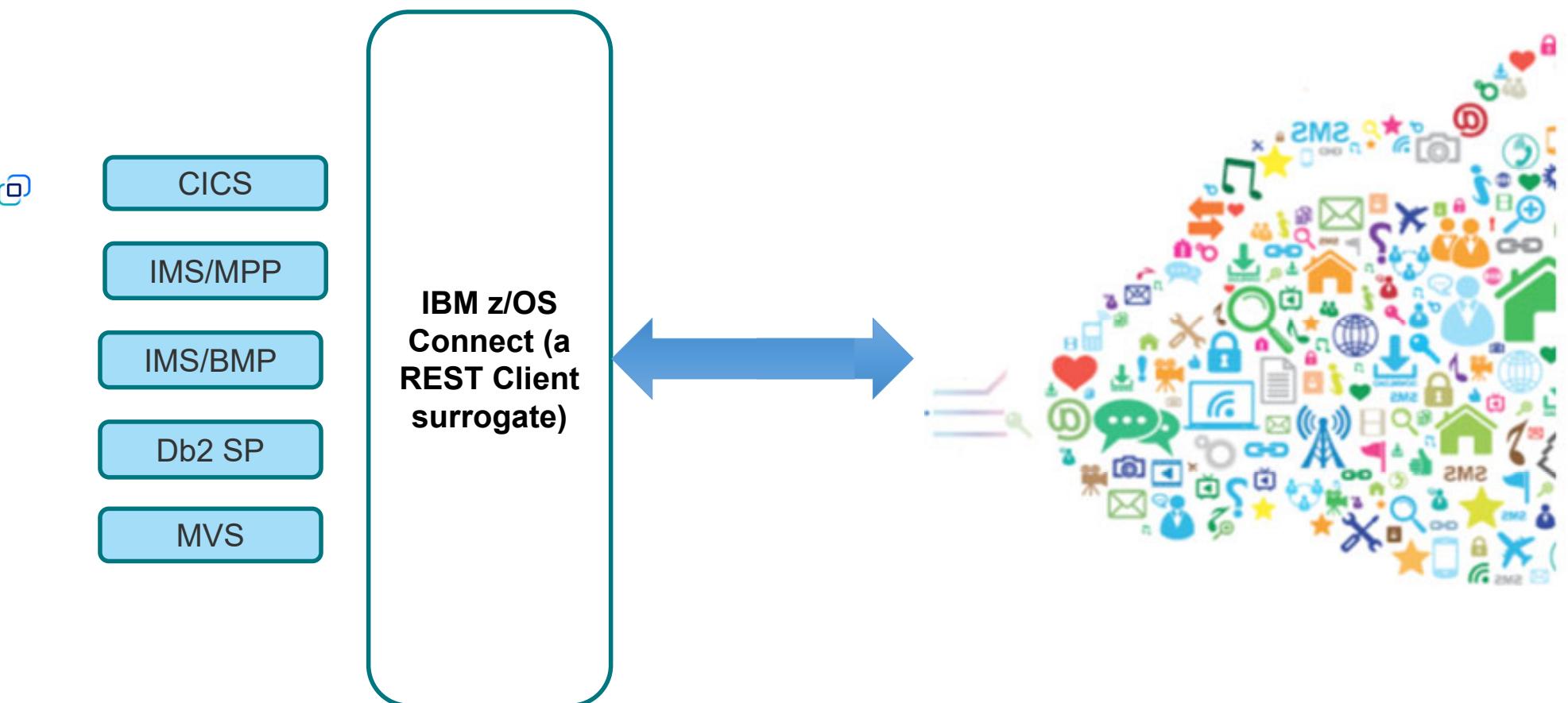


+ HCL and Rocket Software

\*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”?



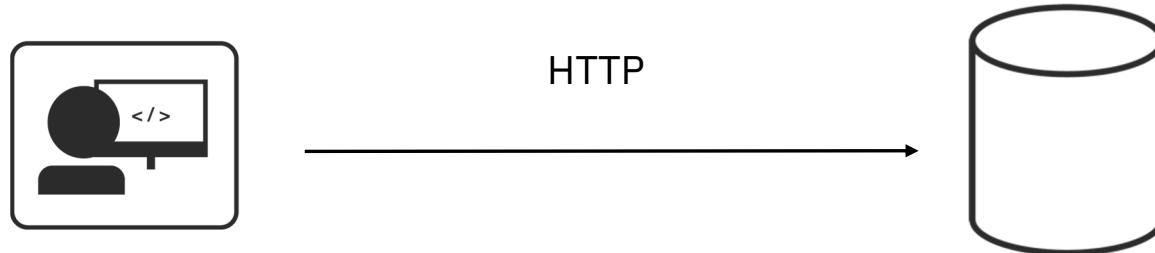
# REST is architectural programming style

**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: To have client code that 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);
    }
}

```

Output:

```

<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/22222
USERID: CICSUMER
CEIBRESP0: 0
CEIBRESP2: 0
name: DR E. GRIFFITHS
employeeNumber: 22222
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": "22222", "amount": "$0022.00", "address": "FRANKFURT, GERMANY", "phoneNumber": "20034151", "date": "26 11 81"}}, "status": "Success", "message": "Employee record retrieved successfully."}

```

**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);
    }
}

```

Output:

```

<terminated> ZceeDb2Get [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: 3000

```

**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));

```

Output:

```

<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:

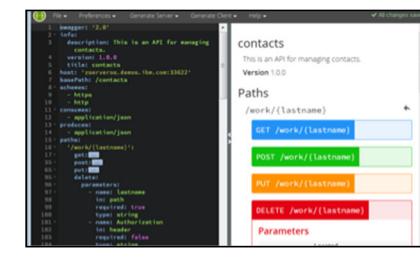
**Code Generation\*** - create stub code to consume APIs from various languages



**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).**

```
{
  "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",
            "schema": {}
          }
        }
      }
    }
  }
}
```

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

```
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.

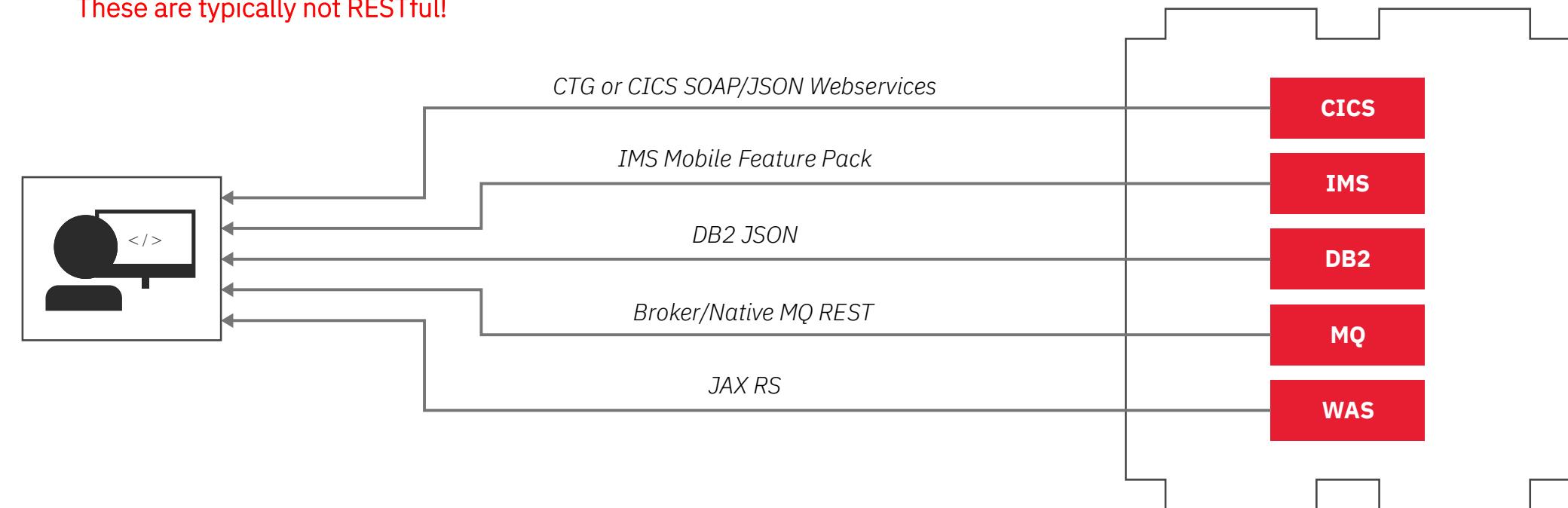


# There was support for REST before z/OS Connect but..

Completely different configuration and management.

Multiple endpoints for developers to call/maintain access to.

These are typically not RESTful!



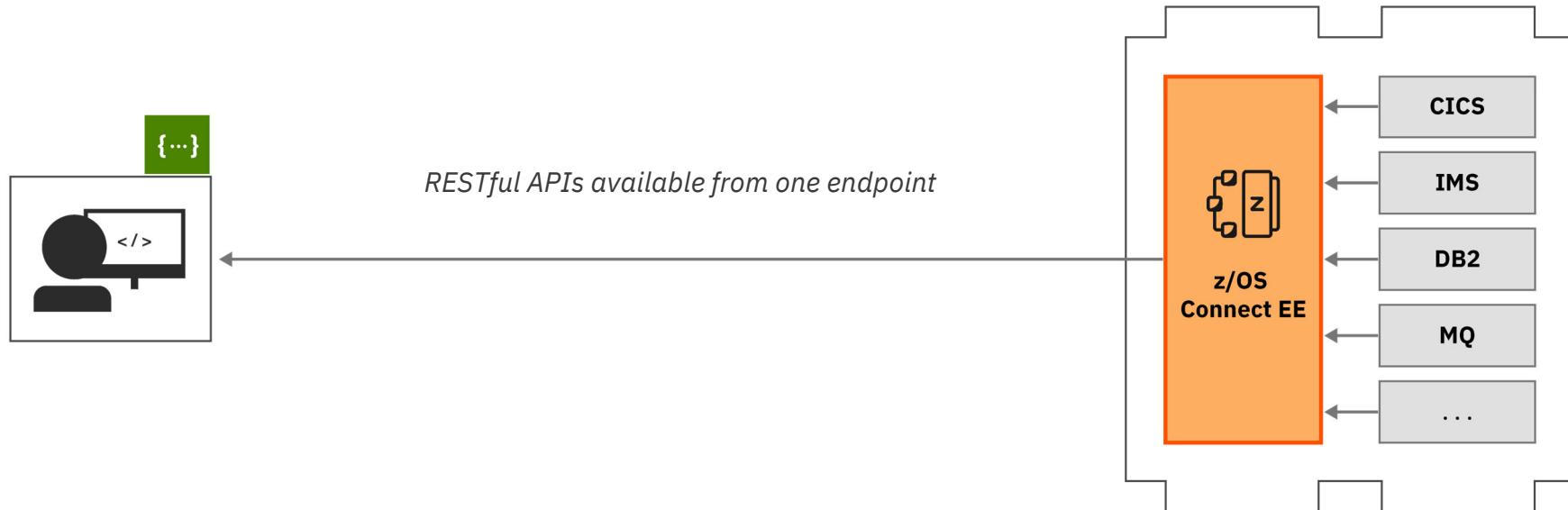


# z/OS Connect provides a single-entry point

- 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.



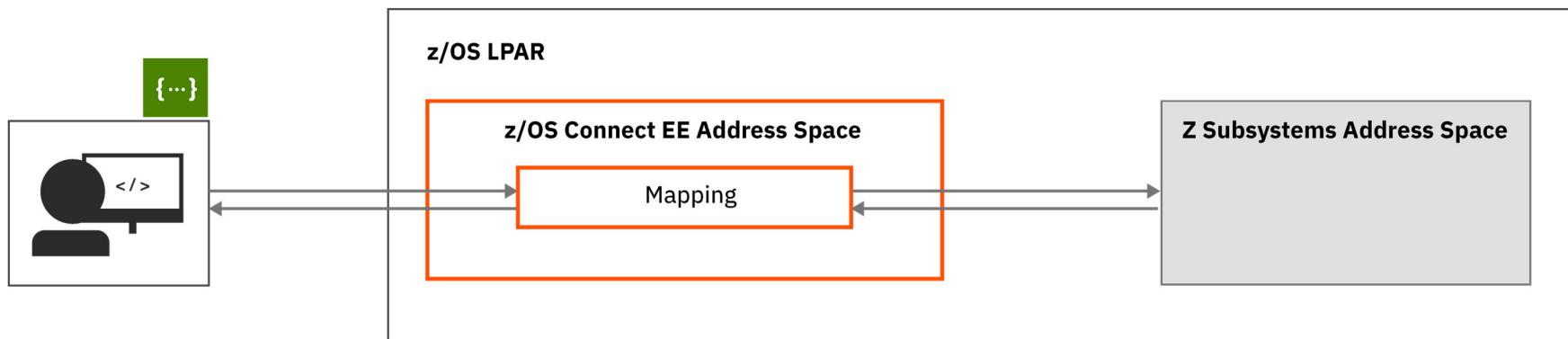


**Other than a RESTful interface,  
what else does z/OS Connect provide?**



# Data mapping/transformation

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

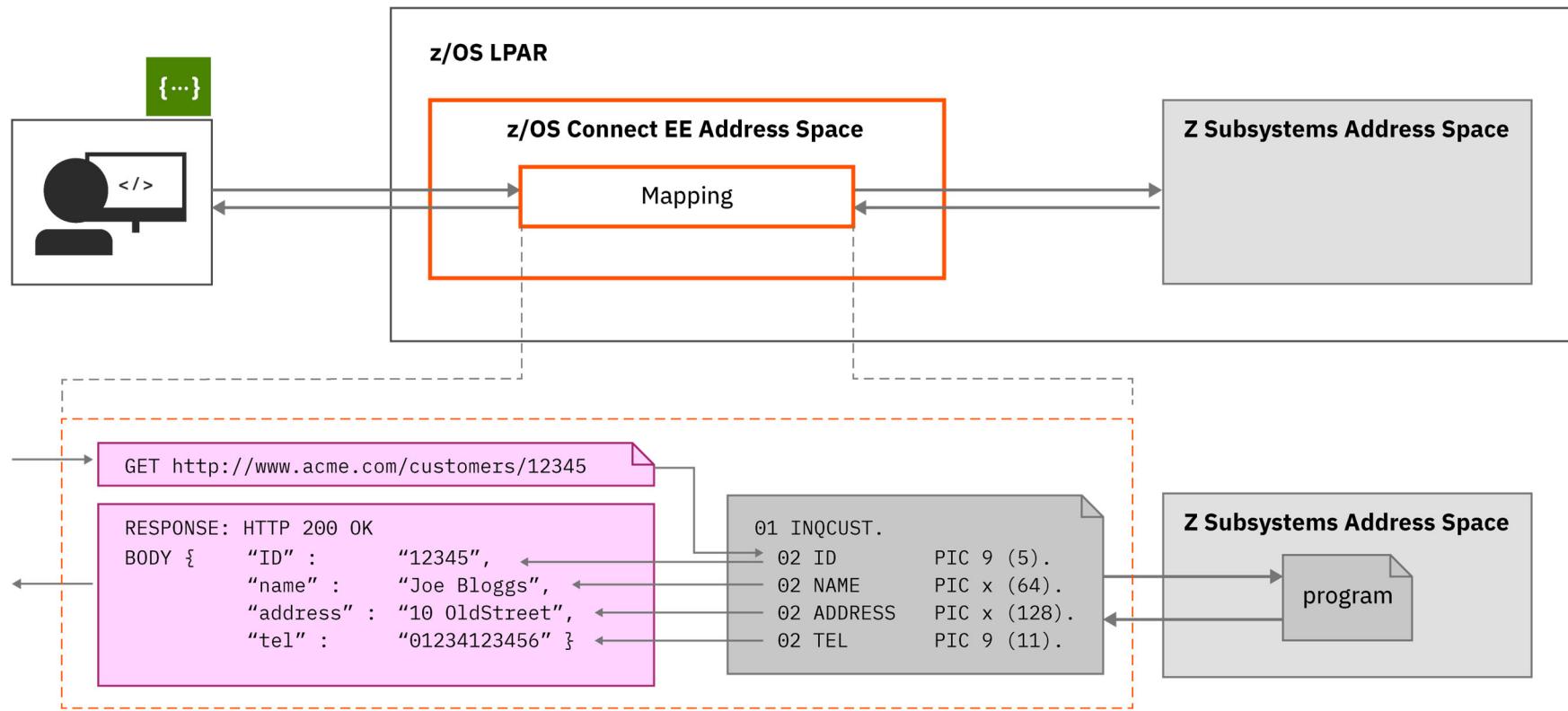


\* Most z/OS subsystems depend on information in a serial data format and do not normally work with JSON request/response messages. Examples of serialized messages are CICS COMMAREAAs and CONTAINERS, IMS or MQ messages, or records stored in sequential or VSAM data sets. Data mapping and transformation 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 methods 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 'Information', 'Security', 'Paths', 'z/OS Assets', 'Components', and various schema definitions. The 'Paths' section is expanded, showing two entries: '/employee' (with POST operation) and '/employee/{employee}' (with GET, PUT, and DELETE operations). On the right, a detailed view of the '/employee' path is shown, listing its summary, description, and operations (POST).

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

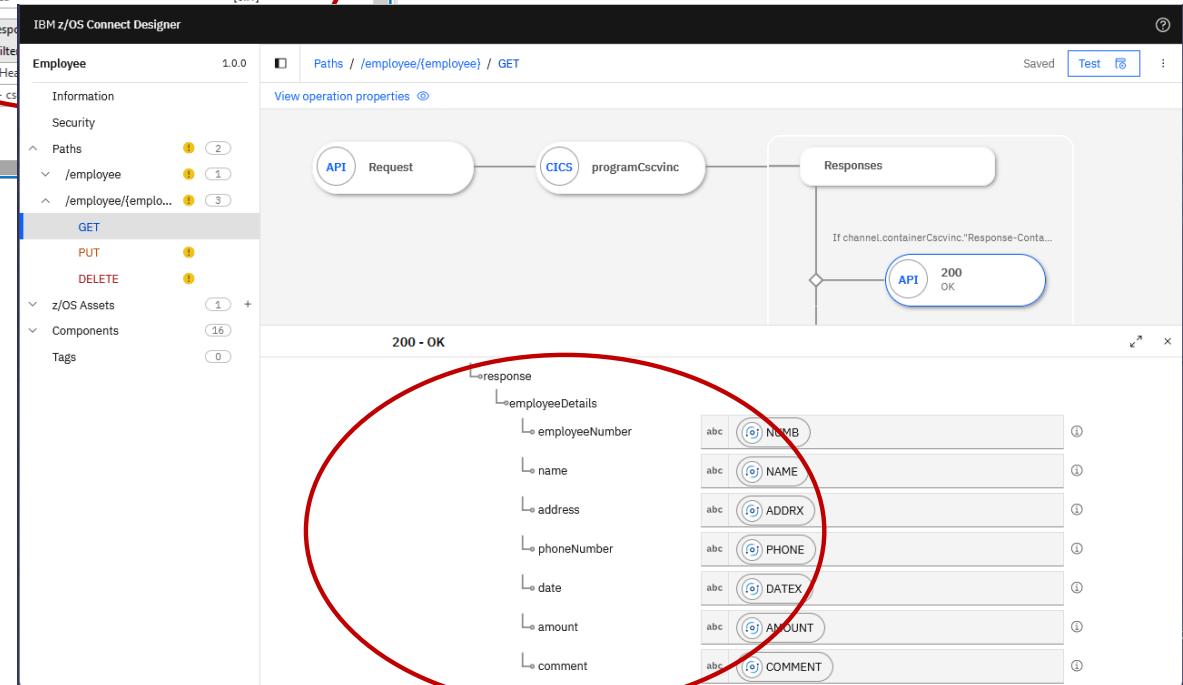
The API specification provides predefined URI Paths and methods.

# 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 a JSON schema for a response message. 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 a similar JSON 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



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.



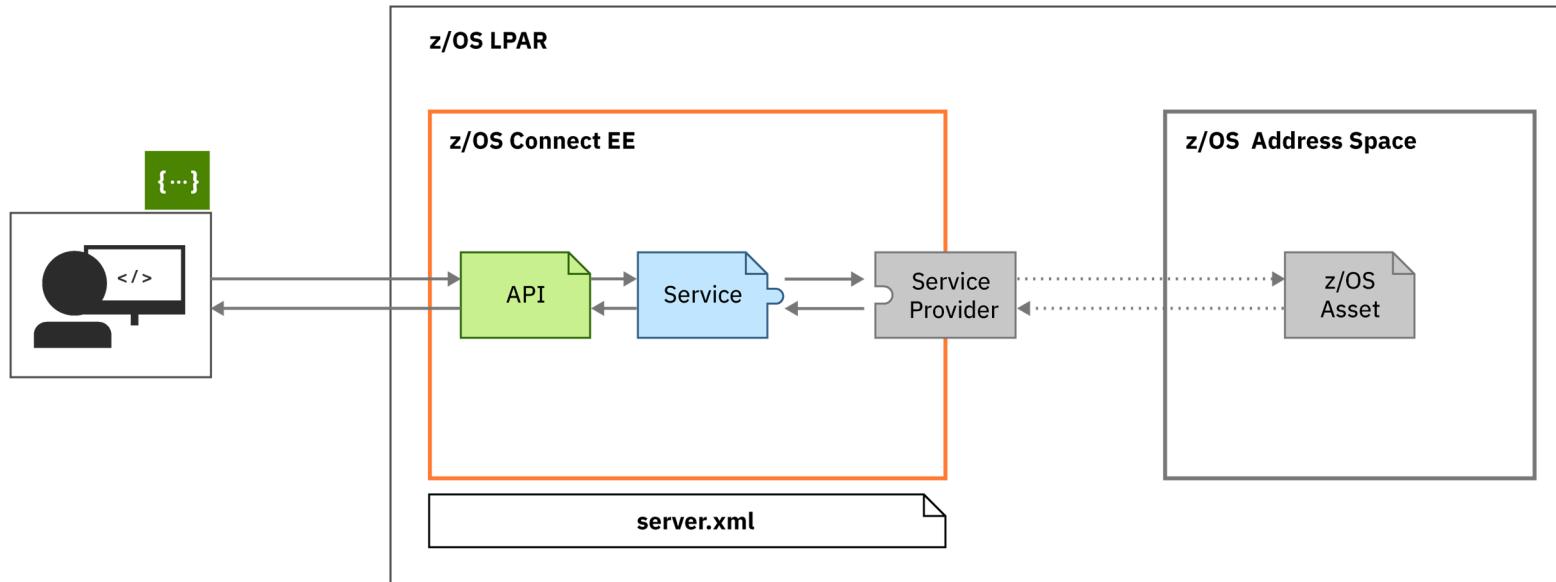
## OpenAPI 2

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)

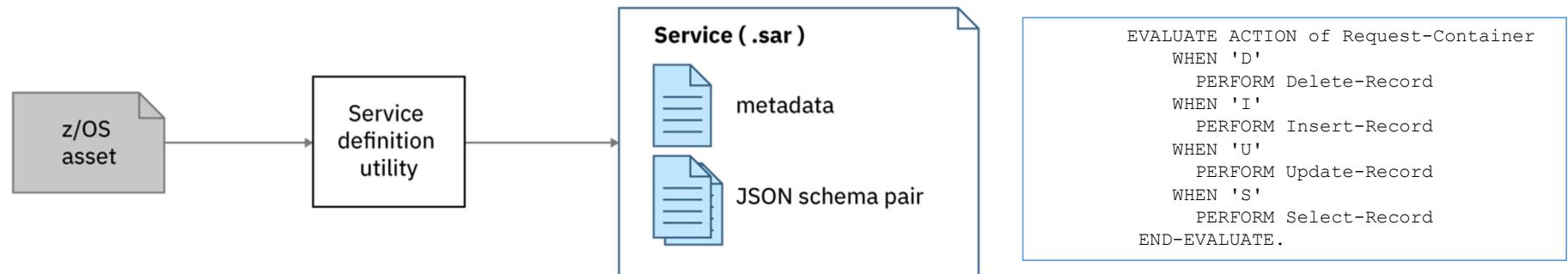


## Describing the interaction (service) with the z/OS resource (OpenAPI 2)

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



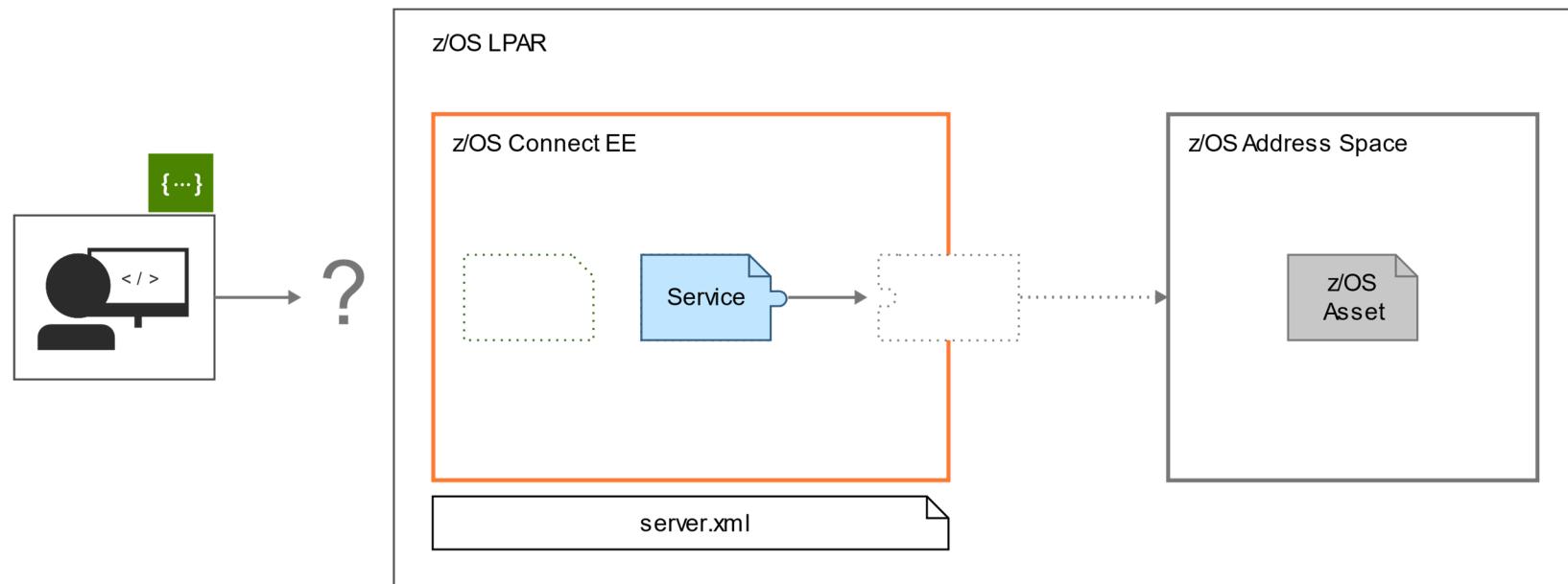
Use a system-appropriate utility to generate a service archive file for the z/OS application

- 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

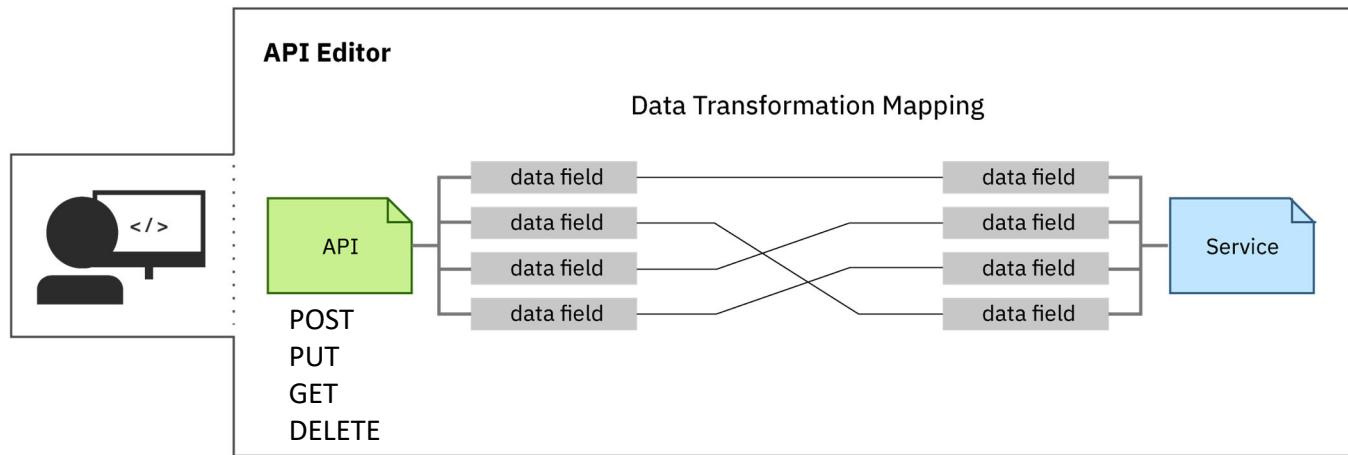


Deploy the service archive file generated in **Step 1** using the right-click deploy in **the API toolkit**.

# 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

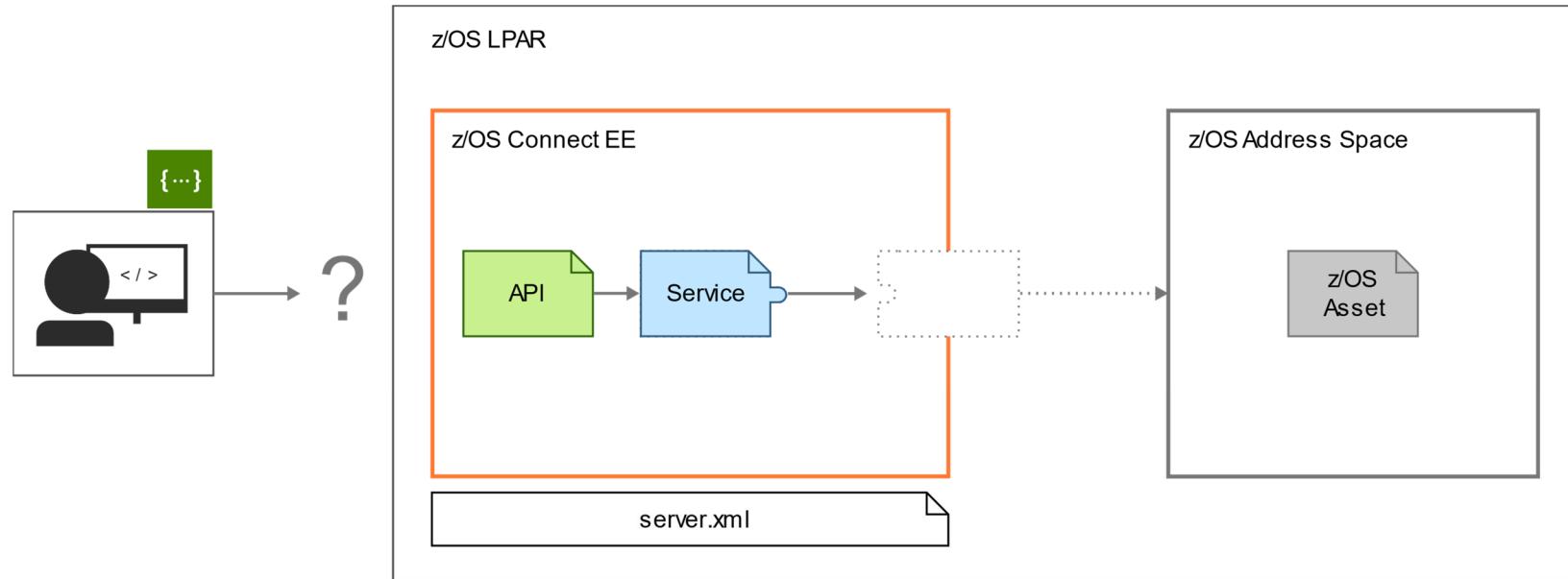


- Import the service archive file into the **API toolkit** and start designing the RESTful API.
- Provides additional data mapping
- Use the editor to describe the API and how it maps to underlying services.



# Deploy the API (Open API 2)

Deploy the API archive file

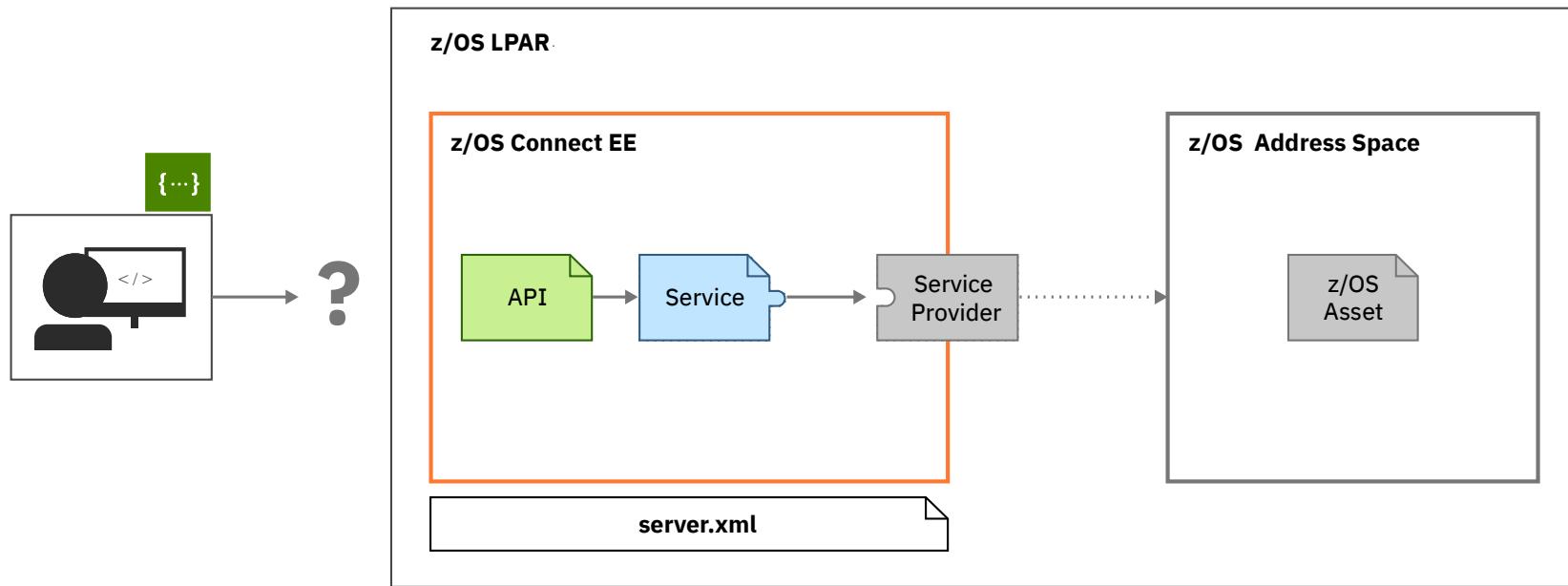


Deploy your API using the right-click deploy in **the API toolkit**



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

Configure the service provider

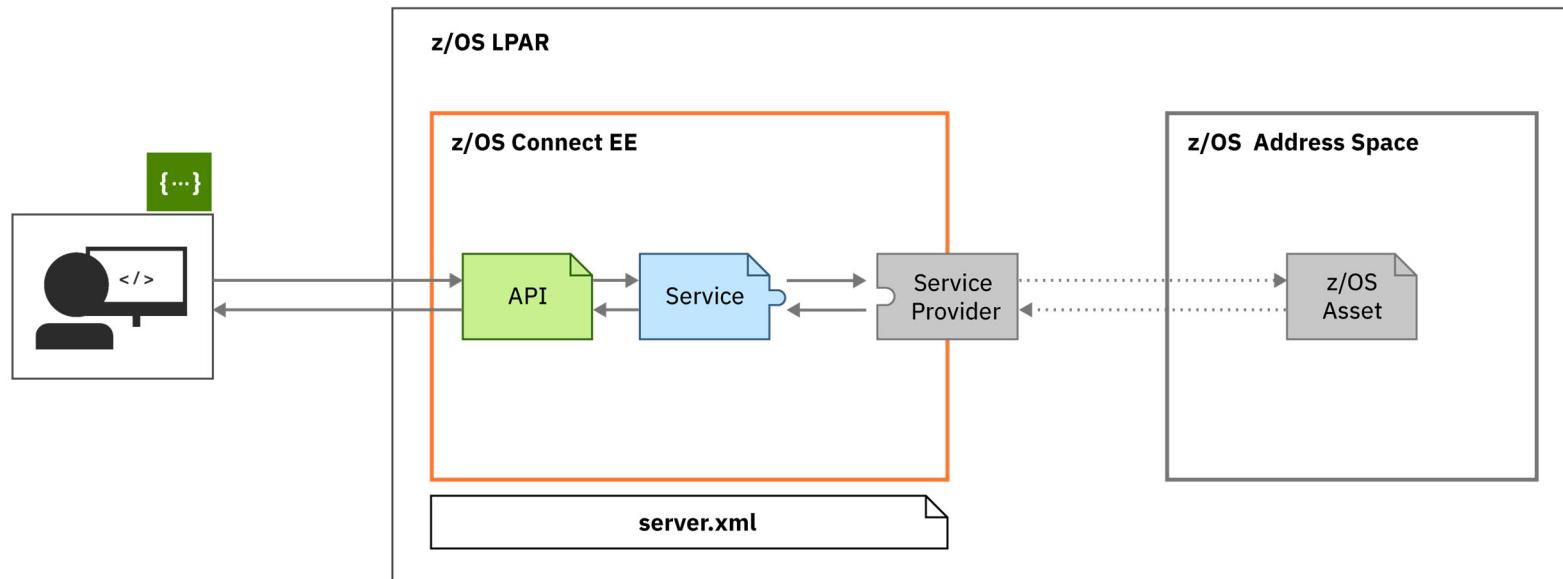


Configure the system-appropriate service provider to connect to your backend system in your **server.xml**.



# 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)

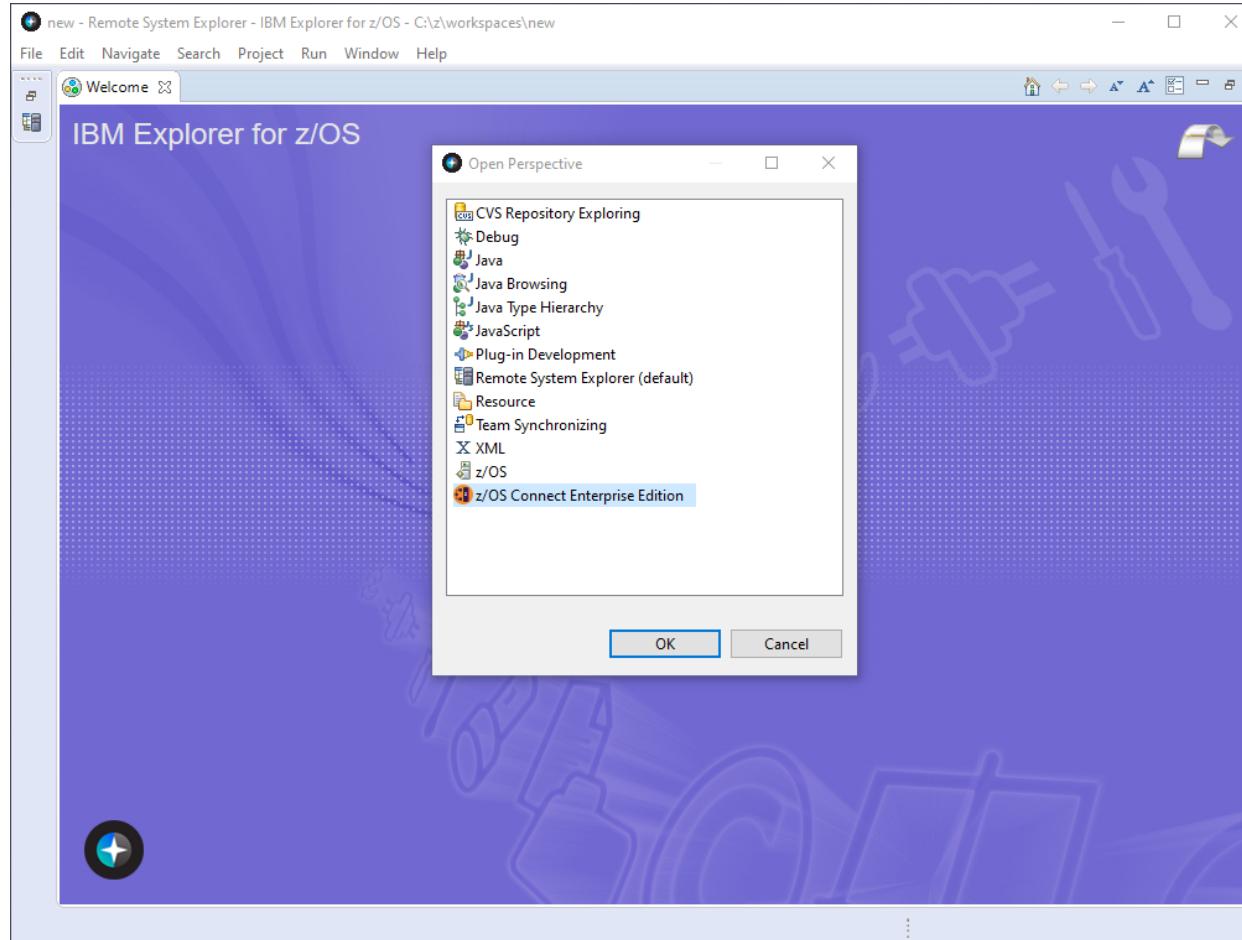


# OpenAPI2

Simple service creation for OpenAPI 2 APIs



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

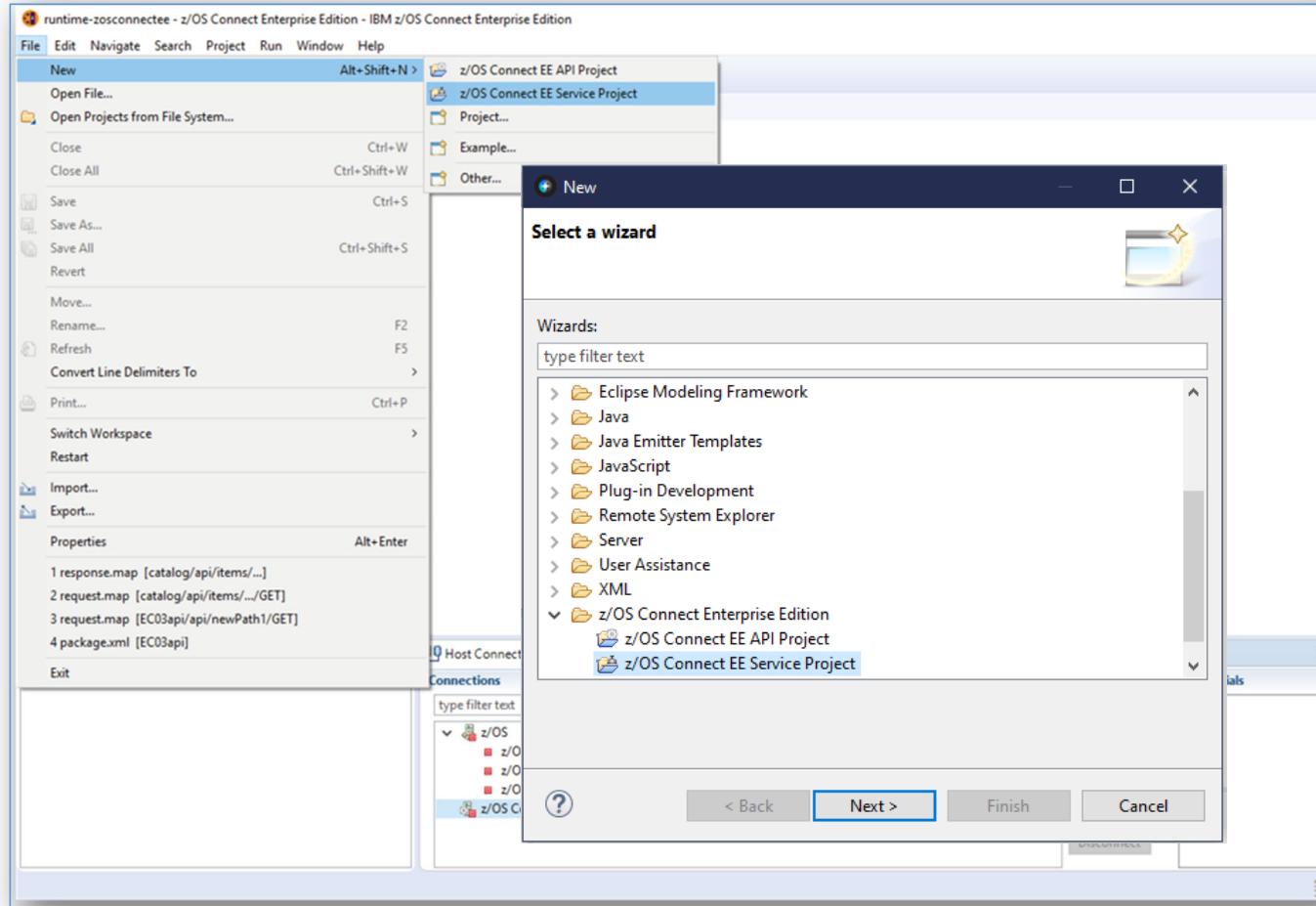


Use the **API toolkit** to create services through Eclipse-based tooling.

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



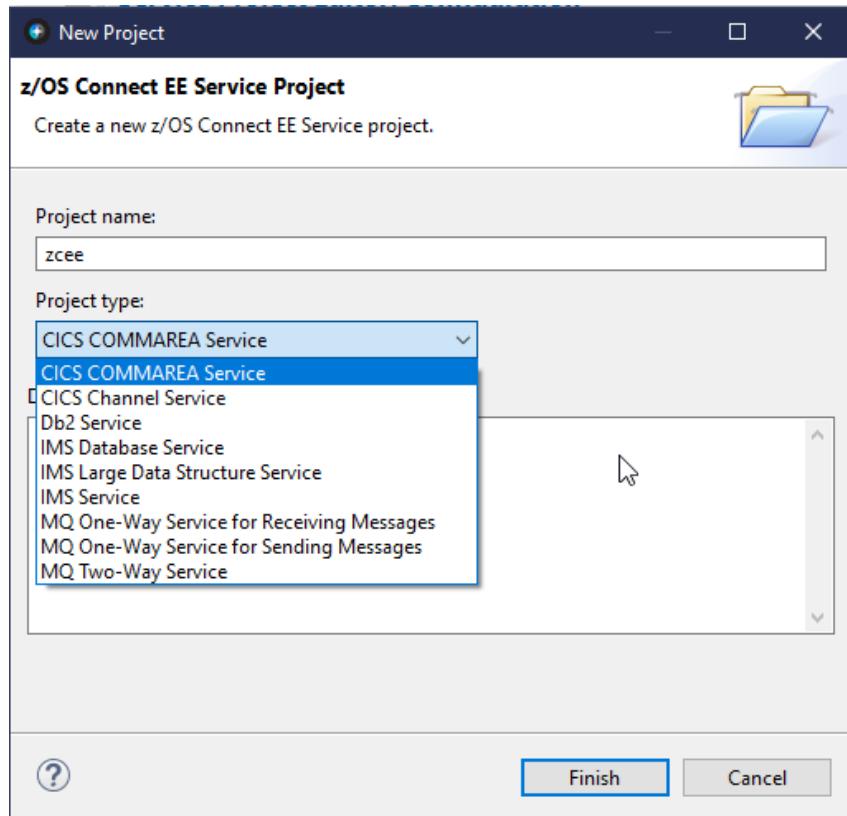
Use the **API toolkit** to create services through Eclipse-based tooling.

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



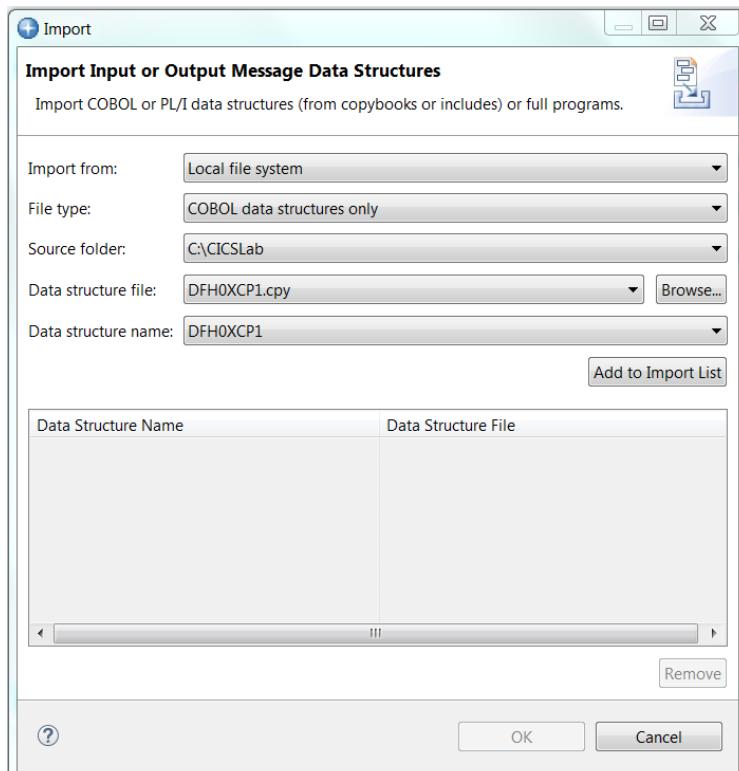
A common interface for service creation, irrespective of back-end subsystem.

- 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



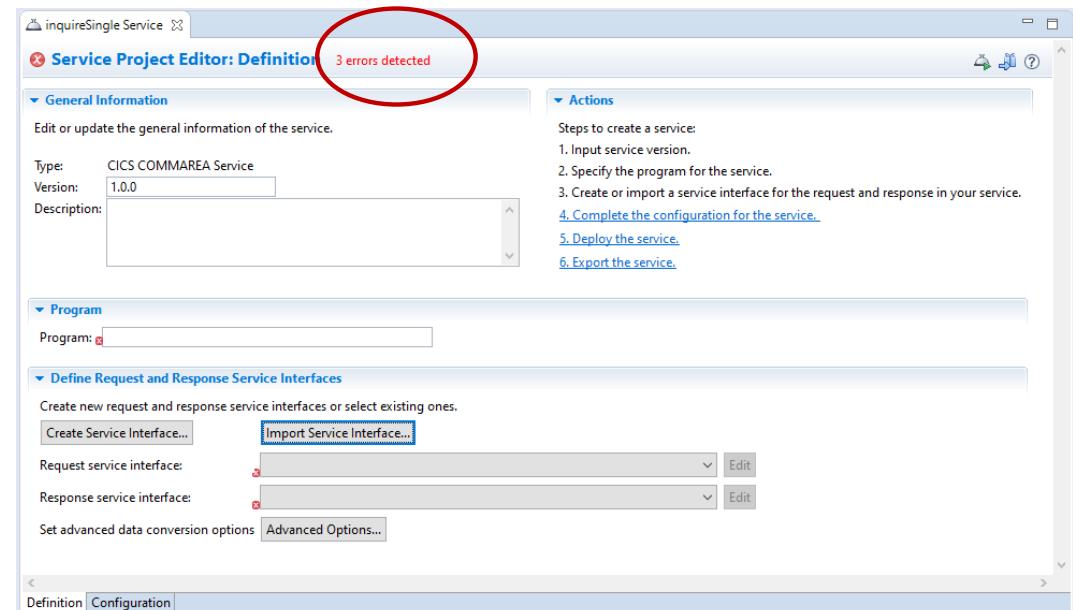
# API toolkit – Creating Services for CICS, IMS TM and MQ

## 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

```
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 – The significance of Transaction ID Usage

**cscvincSelectService Service**

**Service Project Editor: Configuration**

**Required Configuration**

Enter the required configuration for this service.

Coded character set identifier (CCSID): 37

Connection reference: cscvinc

**Optional Configuration**

Enter the optional configuration for this service.

Transaction ID: MUO

Transaction ID usage: EIB\_AND\_MIRROR (selected)

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'  
EIBCPSON = 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  
Connected to remote server/host wg31a using lu/pool TCP00126 and port 23

**EIB\_AND\_MIRROR**

TRANSACTION: MIJO PROGRAM: DFHMIRS DFHDFM STATUS: PROGRAM INITIATION

EIBTIME = 109914  
EIBDATE = 0122050  
EIBTRNID = 'MIJO'  
EIBTRSKN = 8492  
EIBTRMID = '/RBX'  
EIBCPSON = 0  
EIBCALEN = 0  
EIBAID = X'00'  
EIBFN = X'0000'  
EIBRCODE = X'000000000000  
+ EIBDS = '.....'  
EIBREQID = '.....'

TRANSACTION: MIJO PROGRAM: DFHMIRS DFHDFM STATUS: ABOUT TO EXECUTE COMMAND EXEC CSMI LINK PROGRAM  
PROGRAM ('CSCVINC')  
SYNCRETURN CHANNEL ('Channel')  
TRANSID ('MIJO')  
NOHANDLE

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

TRANSACTION: MIJO PROGRAM: CSCVINC TASK: 0008476 APPLID: CICS53Z STATUS: PROGRAM INITIATION

EIBTIME = 181730  
EIBDATE = 0122051  
EIBTRNID = 'MIJO'  
EIBTRSKN = 8837  
EIBTRMID = '/RBX'  
EIBCPSON = 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 :  
PF7 : SCROLL BACK PF8 :  
PF10: PREVIOUS DISPLAY PF11 :  
M1 D  
Connected to remote server/host wg31a using lu/pool TCP00126 and port 23

TRANSACTION: MIJO PROGRAM: CSCVINC TASK: 0008837 APPLID: CICS53Z STATUS: PROGRAM INITIATION

EIBTIME = 182613  
EIBDATE = 0122051  
EIBTRNID = 'MIJO'  
EIBTRSKN = 8949  
EIBTRMID = '/ADB'  
EIBCPSON = 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 C  
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

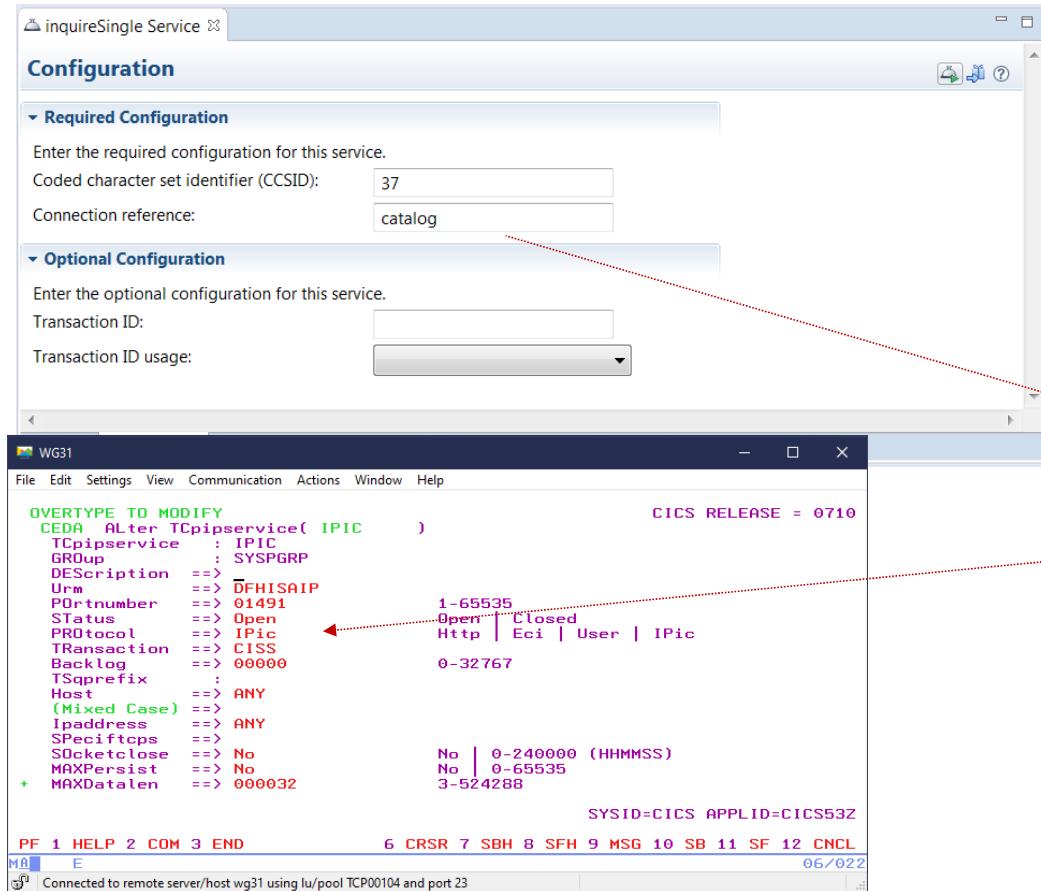
mitchj@us.ibm.com

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

catalog.xml

Design      Source

```
1<server description="CICS IPIC - catalog">
2
3<!-- Enable features -->
4<featureManager>
5  <feature>zosconnect:cicsService-1.0</feature>
6</featureManager>
7
8<zosconnect_cicsIpicConnection id="catalog">
9  host="wg31.washington.ibm.com"
10 port="1491"
11 transid="CSMI"
12 transidUsage="EIB_AND_MIRROR"/>
13
14</server>
15
```

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.

Search:

Fields	Include	Interface Rename	Default Field Value	Data Type	Field Length
ivtnoDisplayRequest					
Segment 1					
INPUT_MSG	<input type="checkbox"/>	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 type="checkbox"/>	IN_COMMAND		CHAR	8
IN_LAST_NAME	<input checked="" 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:

Interaction profile:

Optional Configuration

Enter the optional configuration for this service.

IMS destination override:

Program name:

Definition Configuration

# IMS Connections and Interactions



ivtnoService Service

**Configuration**

**Required Configuration**

Enter the required configuration for this service.

Connection profile: **IMSCONN**

Interaction profile: **IMSINTER**

**Optional Configuration**

Enter the optional configuration for this service.

IMS destination override:

Program name:

Overview Configuration

## Connection Profile

```
<server>
<imsmobile_imsConnection comment="" connectionFactoryRef="CF1"
connectionTimeout="-1" connectionType="IMSCONNECT" id="IMSCONN"/>
<connectionFactory containerAuthDataRef="Connection1_Auth" id="CF1">
    <properties.gmea 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,XTBAREA=100,RACF=Y,RRS=N)
TCPIP=(HOSTNAME=TCPIP,PORTID=(4000,LOCAL),RACFID=JOHNSON,
TIMEOUT=5000)
DATASTORE=(GROUP=OTMAGRP, ID=IVP1, MEMBER=HWSMEM, TMEMBER=OT
MAMEM)
```



# 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 title bar says 'response X'. The main area is titled 'Service Interface Editor' with a help icon. A message at the top says: '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.' Below this is a search bar and a set of toolbar icons. The main content is a table with columns: 'Fields', 'Include', 'Interface Rename', 'Default Field Value', 'Data Type', and 'Field Length'. The 'Fields' column lists service interface definitions. The 'Include' column has checkboxes. The 'Interface Rename' column contains field names. The 'Default Field Value' column is empty. The 'Data Type' column shows data types like ARRAY and CHAR. The 'Field Length' column shows lengths like 0, 10, and 7. A red circle highlights the checkbox for 'result' in the 'Include' column. A red box highlights the entire row for 'result'.

Fields	Include	Interface Rename	Default Field Value	Data Type	Field Length
selectByName_Response					
Segment 1					
Output Columns					
Result [0..*]	<input checked="" type="checkbox"/>	response			
FIRSTNAME	<input checked="" type="checkbox"/>	result		ARRAY	0
ZIPCODE	<input checked="" type="checkbox"/>	firstName		CHAR	10
PHONENR	<input checked="" type="checkbox"/>	zipCode		CHAR	7
A1111111	<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,RACF=N,RRS=N)
TCPIP=(HOSTNAME=TCPIP,PORTID=(4000,LOCAL),RACFID=JOHNSON,TIMEOUT=5000)
DATASTORE=(GROUP=OTMAGRP,ID=IVP1, MEMBER=HWSMEM,T MEMBER=OTMAMEM)
IMSPLEX=(MEMBER=IMS14HWS,T MEMBER=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

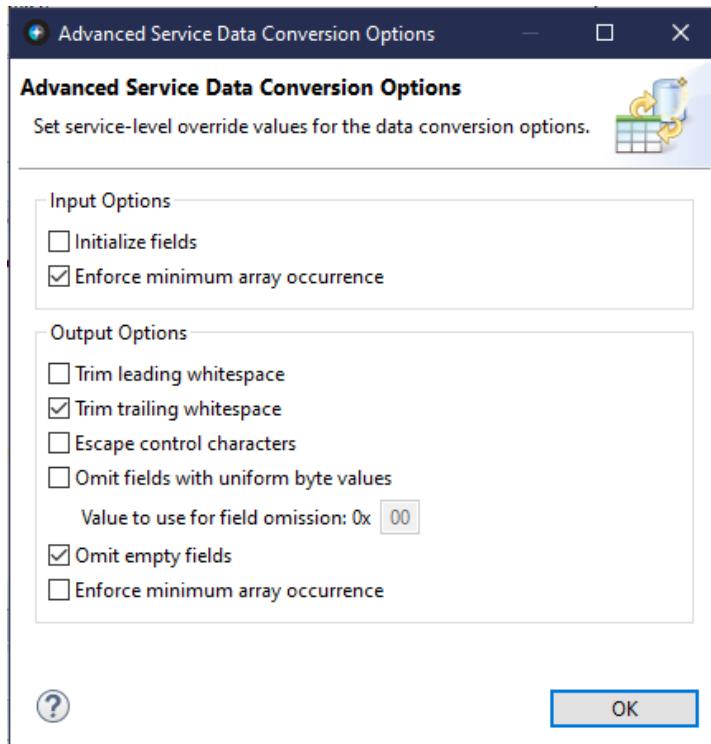
```

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="BROADCAST"/>
15   <queueManager>QMZ1</queueManager>
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.MQZCEE.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 context menu for 'z/OS Connect EE' is displayed, with 'Deploy Service to z/OS Connect EE Server' highlighted. To the right, two dialogs are shown: 'Deploy Service to z/OS Connect EE Server Result' and 'Export Service Package'. The 'Deploy Service to z/OS Connect EE Server Result' dialog shows deployment results for 'z/OS Connect EE Server: wg31:9453' with one service named 'cscvincDeleteSe...' successfully deployed. The 'Export Service Package' dialog allows selecting the export location ('Workspace' is selected) and provides fields for folder ('/Services'), file name ('cscvincDeleteService'), and options like 'Overwrite service package file'.

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

?



## /api\_toolkit/services

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



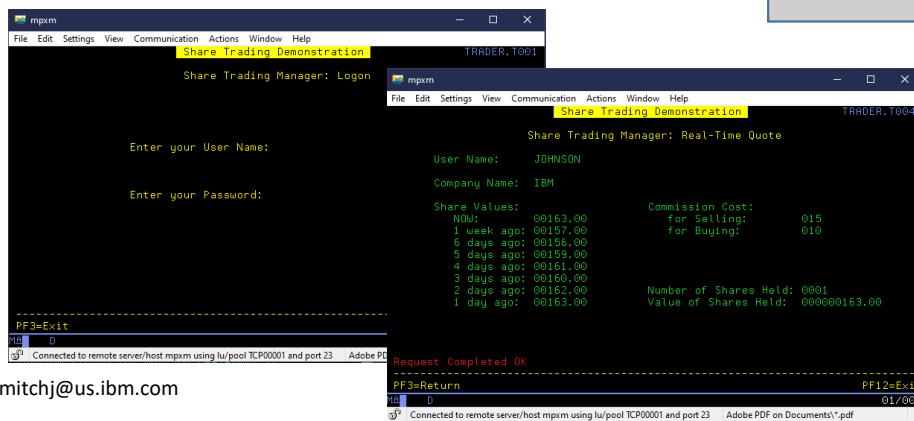
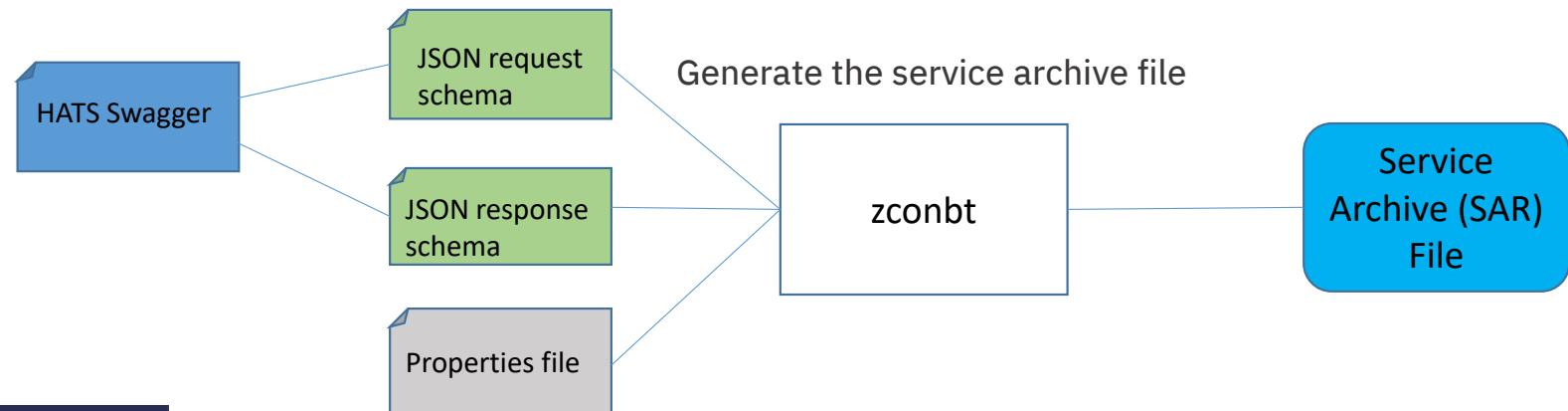
# Use HATS to capture screen flows and input/output fields

The screenshot shows the Rational Application Developer for WebSphere Software interface. On the left, the Project Explorer displays a project structure under 'Trader' with various macros like 'buyShares', 'connect', 'disconnect', 'getCompany', 'getQuoteValue', and 'sellShares'. In the center, the code editor shows the Java file 'BuyShares.java' with code related to user input handling. To the right, the 'Host Terminal' window shows a terminal session with a welcome message and a screen capture of a 'Share Trading Demonstration' interface. A 'Screen Recognition Criteria' dialog is open over the terminal, allowing configuration of recognition rules for specific fields. The bottom right corner of the dialog has a 'Hide Advanced <<' button.



# Command line(zconbt) – REST Services

For HATS REST Services use the z/OS Connect Build toolkit (zconbt)



```
provider=rest
name=getCompany
version=1.0
description=Obtain a list of companies
requestSchemaFile=getCompanyRequest.json
responseSchemaFile=getCompanyResponse.json
verb=POST
uri=/Trader/rest/GetCompany
connectionRef=HatsConn
```

# HATS server XML configuration



```
getCompany.properties - Notepad
File Edit Format View Help
provider=rest
name=getCompany
version=1.0
description=Obtain a list of companies
requestSchemaFile=getCompanyRequest.json
responseSchemaFile=getCompanyResponse.json
verb=POST
uri=/Trader/rest/GetCompany
connectionRef=HatsConn
```

Server Config

hats.xml

Read only Close

Design Source

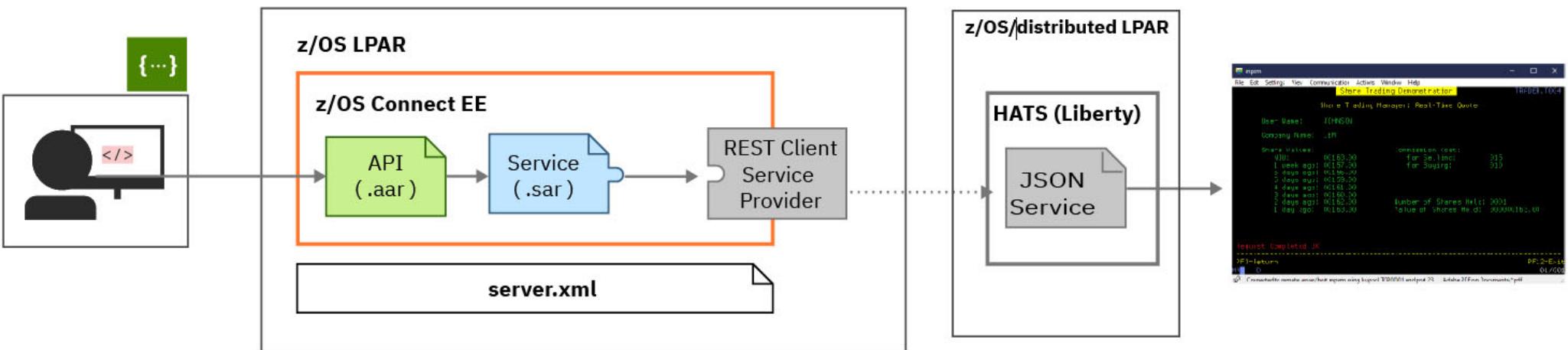
```
<server description="HATS">
  <zosconnect_zosConnectServiceRestClientConnection id="HatsConn"
    host="wg31.washington.ibm.com"
    port="29080" />
</server>
```

## HATS Liberty server.xml

```
<!-- To access this server from a remote client, add a host attribute to the following element, e.g. host="*" -->
<httpEndpoint id="defaultHttpEndpoint"
  httpPort="29080" host="*"
  httpsPort="29443" />
```



# Using HATS to access a 3270 application

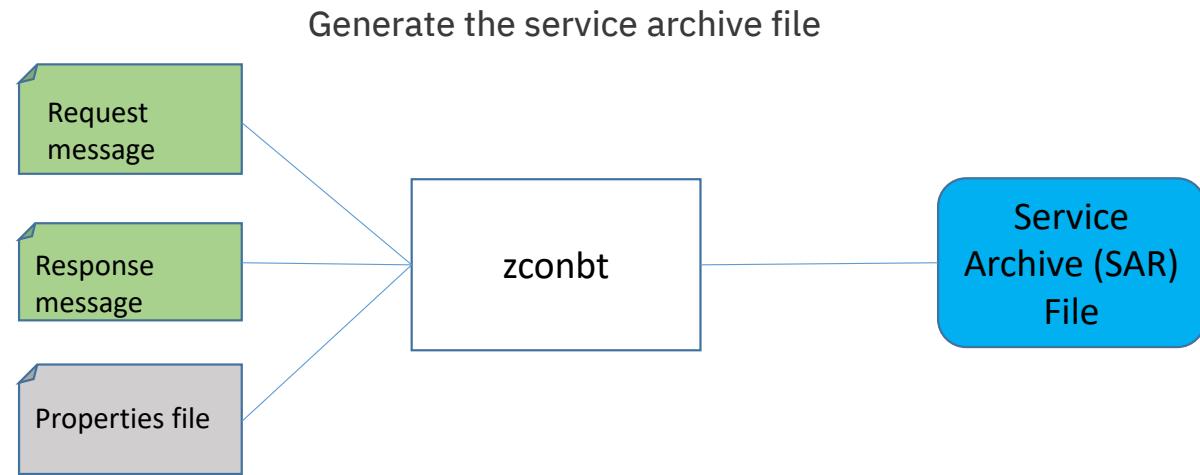




## Command line(zconbt) – MVS Batch

For batch WOLA services use the z/OS Connect Build toolkit (zconbt)

```
name=Filea
version=1.0
provider=wola
description=COBOL batch program
language=COBOL
program=ATSFFILEA
register=FILEAZCON
connectionRef=wolaCF
requestStructure=fileareq.cpy
responseStructure=filearsp.cpy
```



**WebSphere Optimized Local Adapter** – a protocol for cross memory communications between address spaces

# MVS batch server XML



```
filea.properties - Notepad
File Edit Format View Help
name=Filea
version=1.0
provider=wola
description=Test COBOL batch program
language=COBOL
program=ATSFFILEA
register=FILEAZCON
connectionRef=wolaCF
requestStructure=./fileareq.cpy
responseStructure=./filearsp.cpy
```

Server Config

wola.xml

Design      Source

```
<server description="WOLA">
  <featureManager>
    <feature>zosLocalAdapters-1.0</feature>
  </featureManager>
  <zosLocalAdapters wolaGroup="ZCEESRVR">
    wolaName2="ZCEESRVR"
    wolaName3="ZCEESRVR"/>
  <connectionFactory id="wolaCF">
    jndiName="eis/ola">
      <properties.ola/>
    </connectionFactory>
  </zosLocalAdapters>
</server>
```

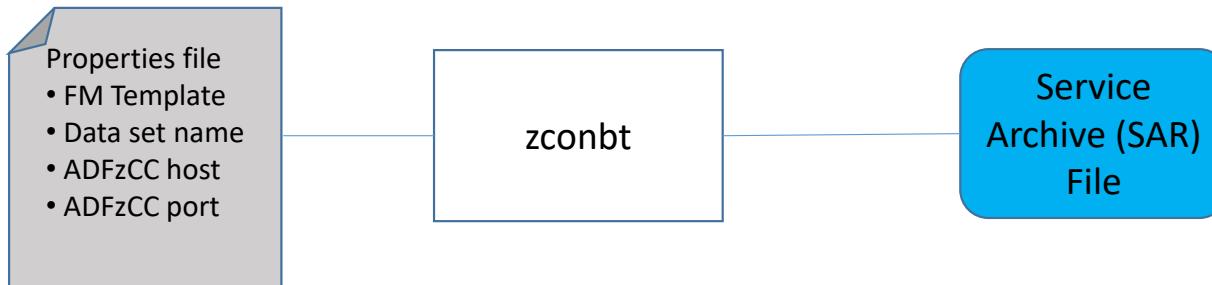
```
* SET THE VALUES FOR USE WITH WOLA REGISTRATION
  MOVE 'FILEAZCON'          TO REG-REGNAME.
  MOVE 'ZCEESRVR'           TO REG-DAEMONGRP.
  MOVE 'ZCEESRVR'           TO REG-NODE.
  MOVE 'ZCEESRVR'           TO REG-SVRNAME.
  MOVE 'ATSFFILEA'          TO SVC-SERVICE-NAME.
  INSPECT REG-DAEMONGRP CONVERTING ' ' to LOW-VALUES.
* Register to a Local Liberty server
  CALL 'BBOA1REG' USING
    REG-DAEMONGRP,REG-NODE,REG-SVRNAME,REG-REGNAME,REG-MINCONN,REG-MAXCONN,REG-FLAGS,RSP-RC,RSP-RSN.
```



## Command line(zconbt) – File Manager

For File Manager Services use the z/OS Connect Build toolkit (zconbt)

Generate the service archive file



```
WG31 - 3270
File Edit Settings View Communication Actions Window Help
Process Options Help
File Manager Copybook and Template Utility functions
Command ==>
1 Workbench Create, edit or update single templates
2 Print View or print copybooks or templates
3 Update Convert existing template(s) into template(s)
4 Update Update template(s)
5 Import Import template(s)
6 Export Export template(s)
7 Repository Repository utilities menu
04/015
Connected to remote server/host wg31 using lu/pool TCP00102 and port 23
```

```
name=filea
provider=filemanager
host=wg31.washington.ibm.com
version=1.0
port=2800
file=USER1.ZCEE.FILEA
template=USER1 TEMPLATE(FILEA)
connid=default
userid=USER1
passwd=USER1
```

# File Manager server XML



```
filea.properties - Notepad
File Edit Format View Help
name=filea
provider=filemanager
host=wg31.washington.ibm.com
version=1.0
port=2800
file=USER1.ZCEE.FILEA
template=USER1.ZCEE.TEMPLATE(FILEA)
connid=default
userid=USER1
passwd=USER1
```

Server Config

filemgr.xml

Read only Close

Design Source

```
<?xml version="1.0" encoding="UTF-8"?>
<server description="new server">
  <!-- Enable features -->
  <featureManager>
    <feature>filemanager:fmProvider-2.0</feature>
  </featureManager>
  <FileManager_Connection id="default"
    runport="2800"
    max_timeout="1800"/>
</server>
```

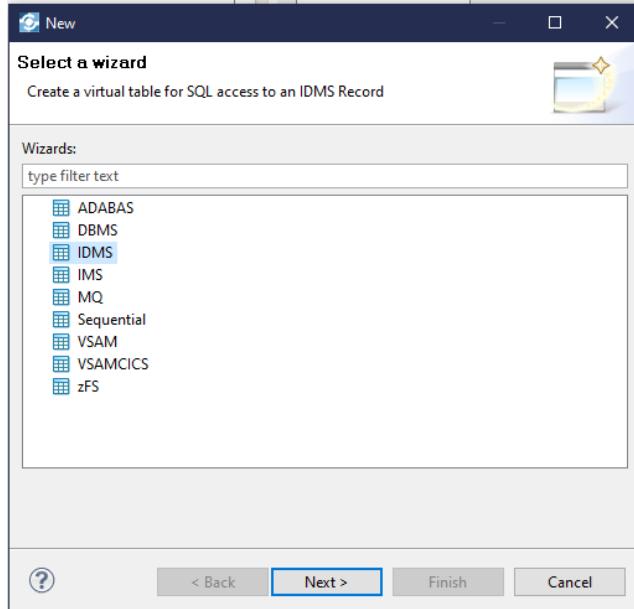
SYS1.PROCLIB(IPVSRV1)

```
//IPVSRV1 PROC PORT=2800,FAMILY='AF_INET',TRACE=N
//      SET ENV=''
//RUN      EXEC PGM=IPVSRV,REGION=40M,
//      PARM='(&ENV/&PORT &FAMILY &TRACE')
// SET IPV=SYSP.ADFZ.JCL          <== Update HLQ
//STEPLIB  DD DISP=SHR,DSN=ADFZ.SIPVMODA      <== ADFzCC APF LIBRARY
//SYSPRINT DD SYSOUT=*
//IPVTRACE DD SYSOUT=*
//STDOUT   DD SYSOUT=*
///* Server wide, then participating product configurations
//CONFIG   DD DISP=SHR,DSN=&IPV.(IPVCFG)
```

# DVM Studio



## DVM uses the DVM Studio



The screenshot shows the DVM Studio interface with the following details:

- SQL Editor:** Displays generated SQL code:

```
-- IEEE LOCATION: wg31.washington.ibm.com:1200/SUB/Data/AVZS/V1LU
-- Remarks: VSAMCICS - USER1.OFFICE.SUPPLIES
SELECT WS_ITEM_REF, WS_DESCRIPTION, WS_DEPARTMENT, WS_COST, WS_IN_
WS_ON_ORDER
FROM EXMPCAT LIMIT 1000;

INSERT INTO EXMPCAT WHERE (WS_ITEM_REF, WS_DESCRIPTION, WS_DEPARTM
WS_ON_ORDER) VALUES('9997','Mitch Johnson','010','002.90','0106'
INSERT INTO EXMPCAT WHERE (WS_ITEM_REF, WS_DESCRIPTION, WS_DEPARTM
WS_ON_ORDER) VALUES('9998','Mitch Johnson','010','002.90','0106'
INSERT INTO EXMPCAT WHERE (WS_ITEM_REF, WS_DESCRIPTION, WS_DEPARTM
WS_ON_ORDER) VALUES('9999','Mitch Johnson','010','002.90','0106'

UPDATE EXMPCAT SET WS_COST = '003.90' WHERE WS_ITEM_REF = 9999

MPCAT WHERE WS_ITEM_REF = 9999
```
- Toolbars and Menus:** Standard Windows-style toolbar and menu bar.
- Left Sidebar:** Data Sources, Navigator, Edit SQL, and Set Current Server.
- Central Panel:** Shows the host (wg31.washington.ibm.com), port (1200), server (AVZS), and version (01.01.00.0).
- Bottom Panel:** Active Connections, Server Trace, Console, and a table preview showing columns WS\_DESCRIPTION, WS\_DEPARTMENT, and WS\_C.

# **z/OS Resources accessible from DVM using SQL\***



Data Source	SELECT	INSERT	UPDATE	DELETE	CALL Statement
CA IDMS	Yes	No	No	No	N/A
CICS COMMAREA	N/A	N/A	N/A	N/A	Yes
DB2 for z/OS	Yes	Yes	Yes	Yes	Yes
Db2 Direct	Yes	No	No	No	N/A
Db2 other platforms	Yes	Yes	Yes	Yes	Yes
IMS DBCTL	Yes	Yes	Yes	Yes	N/A
IMS Direct	Yes	No	No	No	N/A
IMS OTMA	N/A	N/A	N/A	N/A	Yes
MQ	Yes	No	No	No	N/A
Natural	N/A	N/A	N/A	N/A	Yes
SMF	Yes	No	No	No	N/A
Sequential data set	Yes	Yes	No	No	N/A
VSAM data set*	Yes	Yes	Yes	Yes	N/A
z/OS Syslog	Yes	No	No	No	N/A
OMVS files	Yes	No	No	No	Yes

\*Administering IBM DVM Manager, SC27-9303



# DVM server XML

Server Config

dvs.xml

Read only Close

Design Source

```
<?xml version="1.0" encoding="UTF-8"?>
<server description="new server">
<!-- Enable features -->
<featureManager>
<feature>usr:dvsProvider</feature>
<feature>zosLocalAdapters-1.0</feature>
</featureManager>
<!-- Adapter Details with WOLA Group Name (ZCEEDVM) -->
<zosLocalAdapters wolaName3="NAME3"
wolaName2="NAME2"
wolaGroup="ZCEEDVM"/>
<!-- DVS Service Details with Register Name (ZCEEDVM) -->
<zosconnect_zosConnectService invokeURI="/dvs"
serviceDescription=""
serviceRef="dvsService"
serviceName="dvsService"
id="zosConnectDvsService"/>
<usr_dvsService invokeURI="/dvs"
serviceName="DVSS1"
registerName="ZCEEDVM"
connectionFactoryRef="wolaCF"
id="dvsService"/>
<connectionFactory jndiName="eis/ola" id="wolaCF">
<properties.ola/>
</connectionFactory>
<zosconnect_zosConnectService serviceRef="svc1"
serviceAsyncRequestTimeout="600s"
serviceName="dvs1" id="sdef1"/>
<zosconnect_localAdaptersConnectService
connectionWaitTimeout="7200"
connectionFactoryRef="wolaCF"
serviceName="DVSS1"
registerName="ZCEEDVM"
id="svc1"/>
</server>
```

**DVS . AVZS . SAVZEXEC (AVZSIN00)**

```
/*
 * Enable z/OS Connect interface facility
 */
if DoThis then
  do
    /*
     * The following parameter enables the z/OS Connect interface
     * facility.
    */
    "MODIFY PARM NAME(ZCONNECT)           VALUE(YES)"
    "MODIFY PARM NAME(NETWORKBUFFERSIZE)   VALUE(96K)"
    /*
     * The "DEFINE ZCPATH" command(s) can be used to define
     * paths to z/OS Connect regions to handle requests.
     * Use a separate "DEFINE ZCPATH" command to define each
     * path required (Note that a single path can handle
     * several different requests)
     * refer to the documentation for details about the parameters,
     * and information about optional parameters.
    */
    "DEFINE ZCPATH",
    "  NAME(ZCEE)                      ",
    "  RNAME(ZCEEDVM)                  ",
    "  WNAME(ZCEEDVM)                  ",
    ""
  end
```



## 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*



# Deploying Service Archive files outside of the Eclipse tooling

- Use SAR as request message and use HTTP POST
- Use URI path /zosConnect/services
- Postman or cURL

The screenshot shows the Postman interface. In the top navigation bar, there are tabs for File, Edit, View, Help, New, Import, Runner, and My Workspace. A sign-in button is also present. Below the navigation, there are three active requests: a GET request to https://mpx3.washington.ibm.com, a POST request to https://mpx3.washington.ibm.com, and a POST request to https://wg31.washington.ibm.com. The main area is titled "Untitled Request" and shows a POST method directed at https://wg31.washington.ibm.com:9453/zosConnect/services. The "Body" tab is selected, showing a file named "selectEmployee.sar". The file content is displayed as a JSON object:

```
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
{
  "zosConnect": {
    "serviceName": "selectEmployee",
    "serviceDescription": "Select a row from USER1.EMPLOYEE",
    "serviceProvider": "restclient-1.0",
    "serviceURL": "https://wg31.washington.ibm.com:9453/zosConnect/services/selectEmployee",
    "dataFormProvider": "DATA_UNAVAILABLE",
    "serviceStatus": "Started"
  },
  "selectEmployee": {
    "receiveTimeout": 60000,
    "port": "2446",
    "host": "wg31.washington.ibm.com",
    "basicAuthConfigId": "dsn2Auth",
    "id": "Db2Conn",
    "httpMethod": "POST",
    "connectionTimeout": 30000,
    "uri": "/services/selectEmployee"
  }
}
```

Command:

```
curl --data-binary @selectEmployee.sar
--header "Content-Type: application/zip"
https://mpxm:9453/zosConnect/services
```

Results:

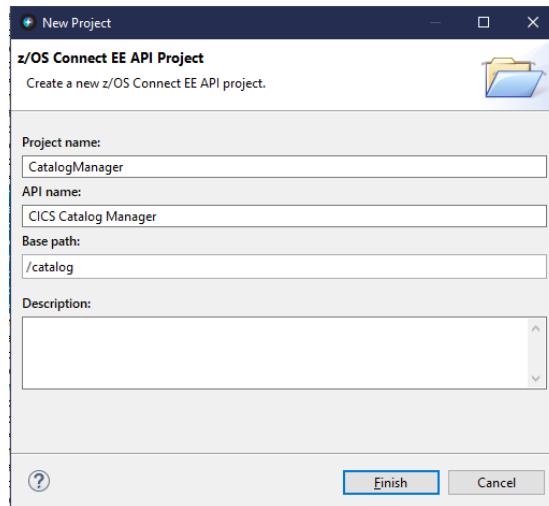
```
{"zosConnect": {"serviceName": "selectEmployee", "serviceDescription": "Select a row from USER1.EMPLOYEE", "serviceProvider": "IBM_ZOS_CONNECT_SERVICE_REST_CLIENT-1.0", "serviceURL": "https://mpxm:9453/zosConnect/services/selectEmployee", "dataFormProvider": "DATA_UNAVAILABLE", "serviceStatus": "Started"}, "selectEmployee": {"receiveTimeout": 0, "port": null, "host": null, "httpMethod": "POST", "connectionTimeout": 0, "uri": "/services/selectEmployee"}}
```



## **/api\_toolkit/api\_editor**

Quick and easy **API mapping.**

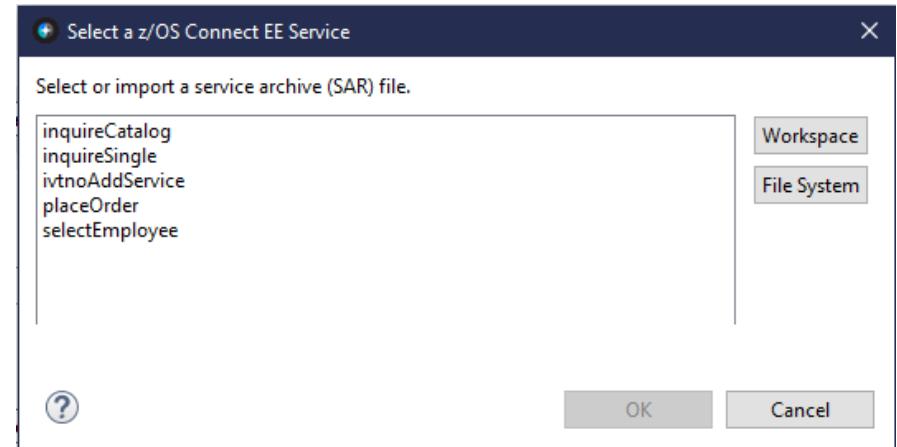
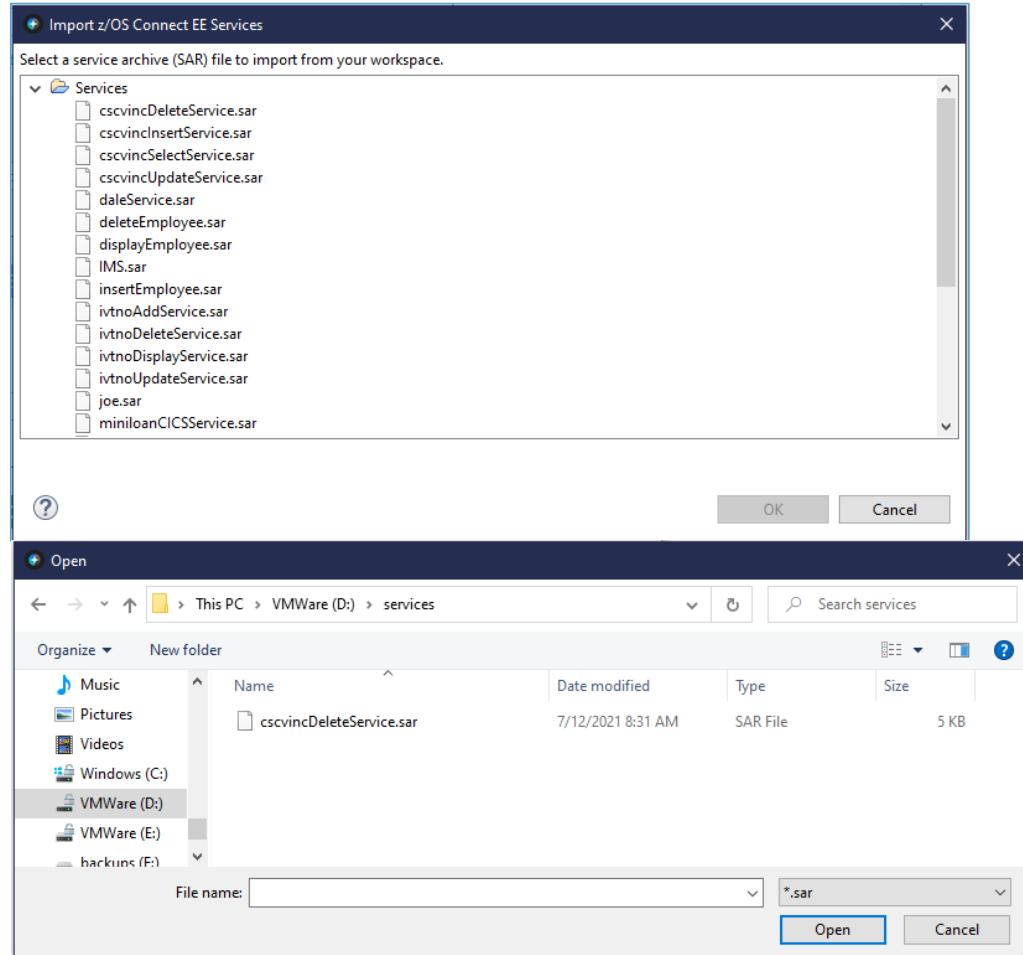
# API toolkit – API Editor



The screenshot shows the API Editor for the 'CICS Catalog Manager API'. The 'Describe your API' section includes fields for Name (CICS Catalog Manager), Description, Base path (/catalog), and Version (1.0.0). The 'Contact Information' section is collapsed. The main area displays a path entry ('/newPath1') and a list of four methods: POST, GET, PUT, and DELETE. Each method has a corresponding input field, a 'Service...' button, a 'Mapping...' button, and up/down arrow buttons for reordering.



# Importing the service archives files



mitchj@us.ibm.com



# 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.

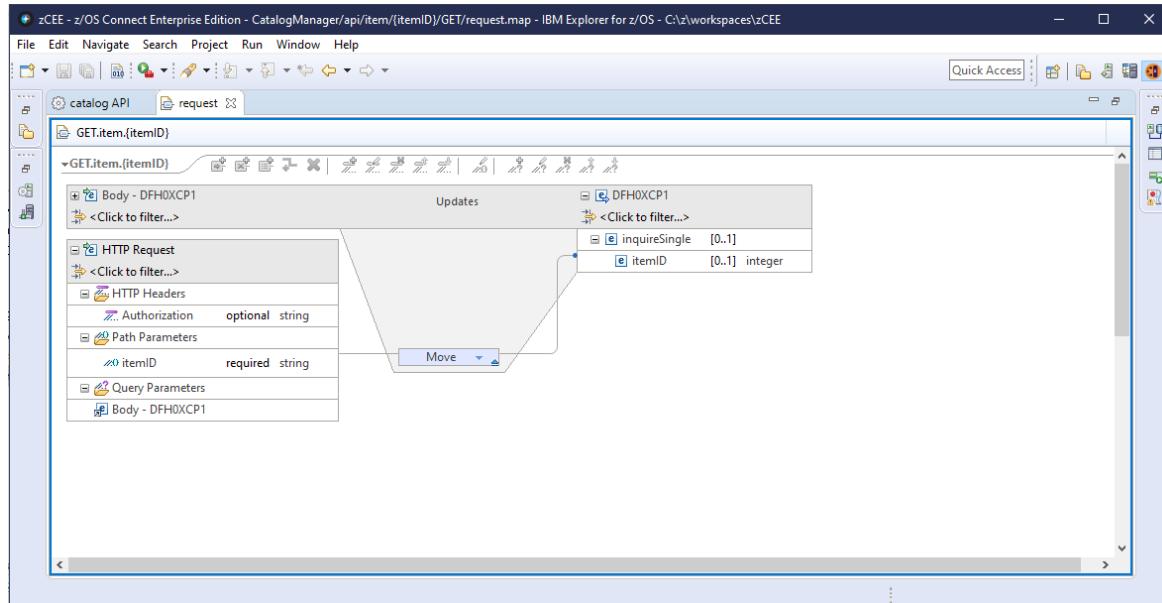
Once you define your API, you can map backend services to each request.

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.

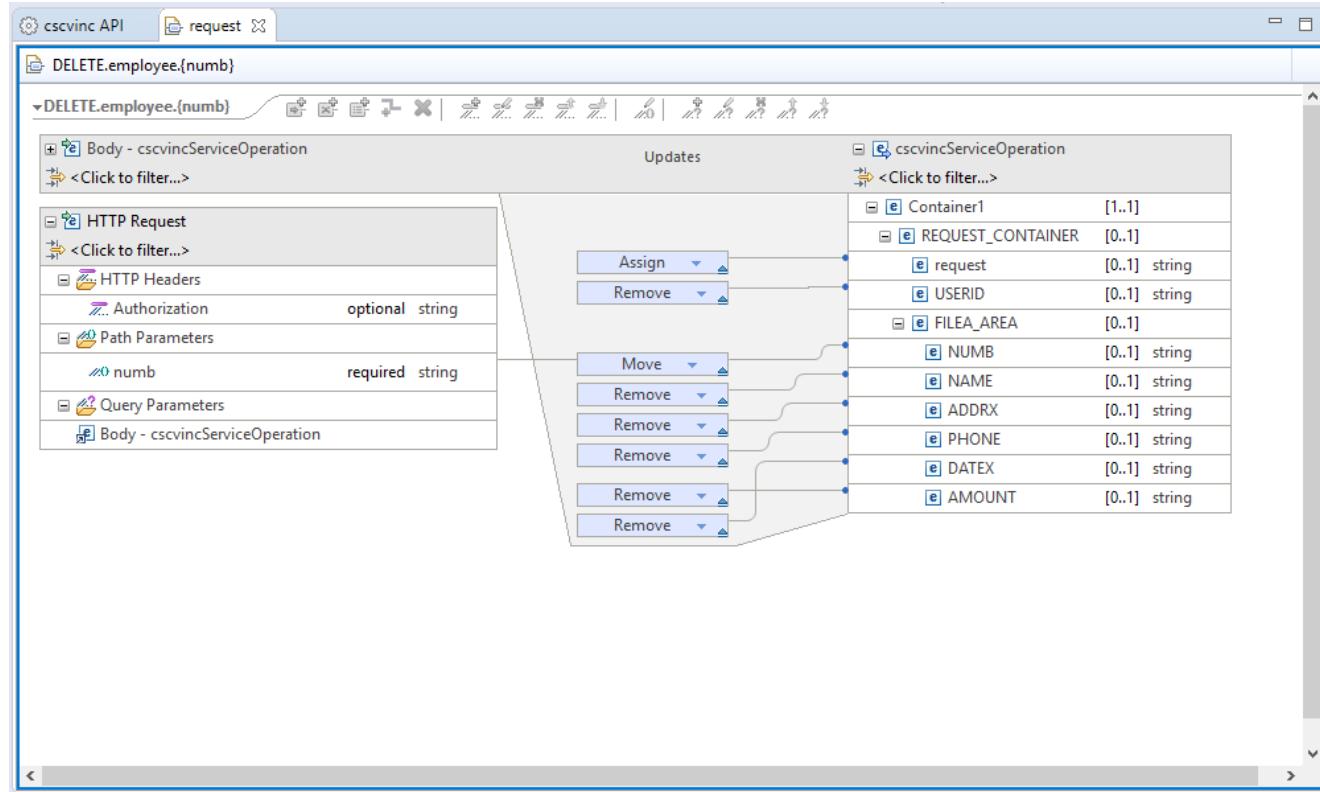
Assign static values to fields, useful for Op codes.

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



# API toolkit – API Editor

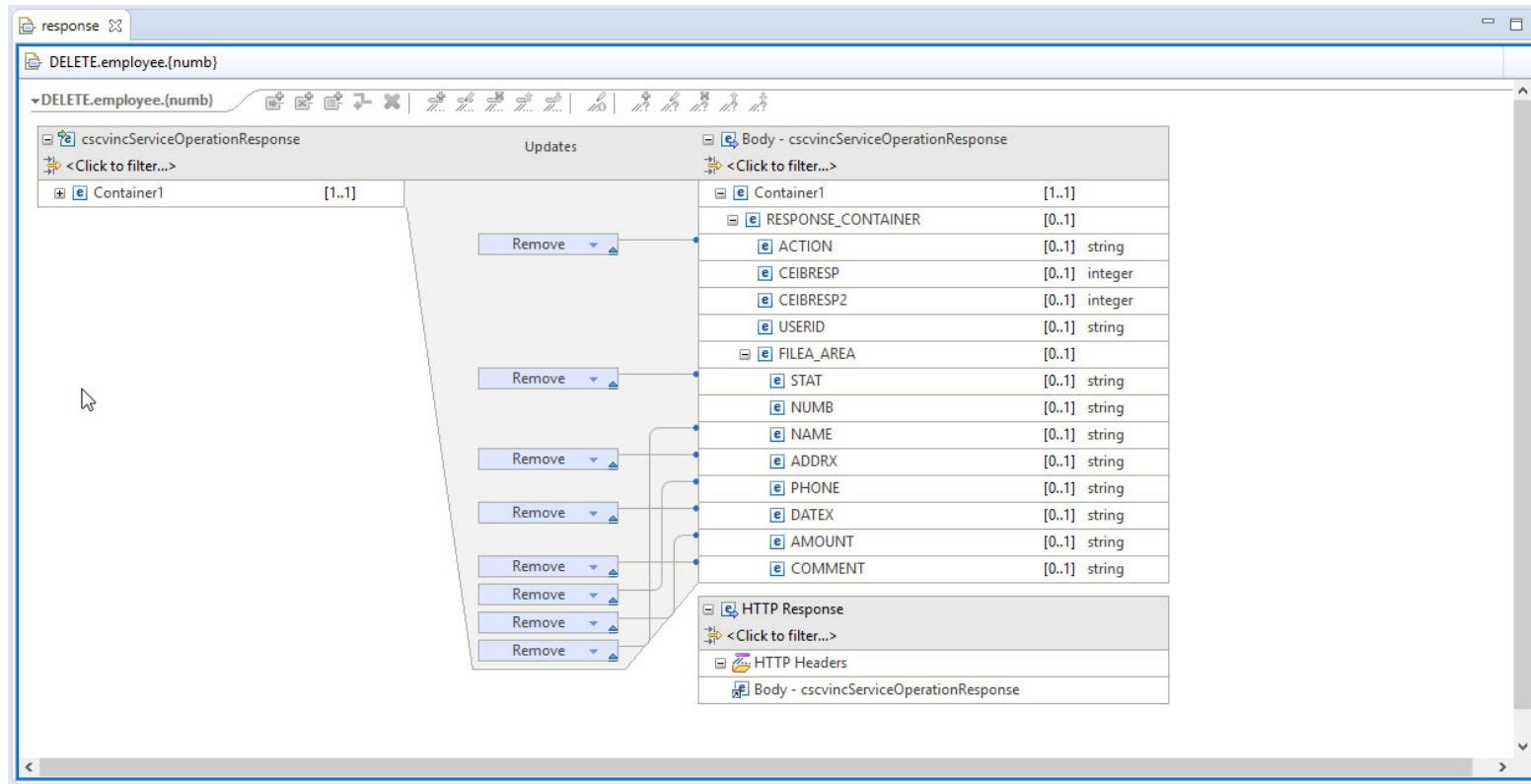
API mapping: Remove or assign values to the fields exposed by service developer





# API toolkit – API Editor

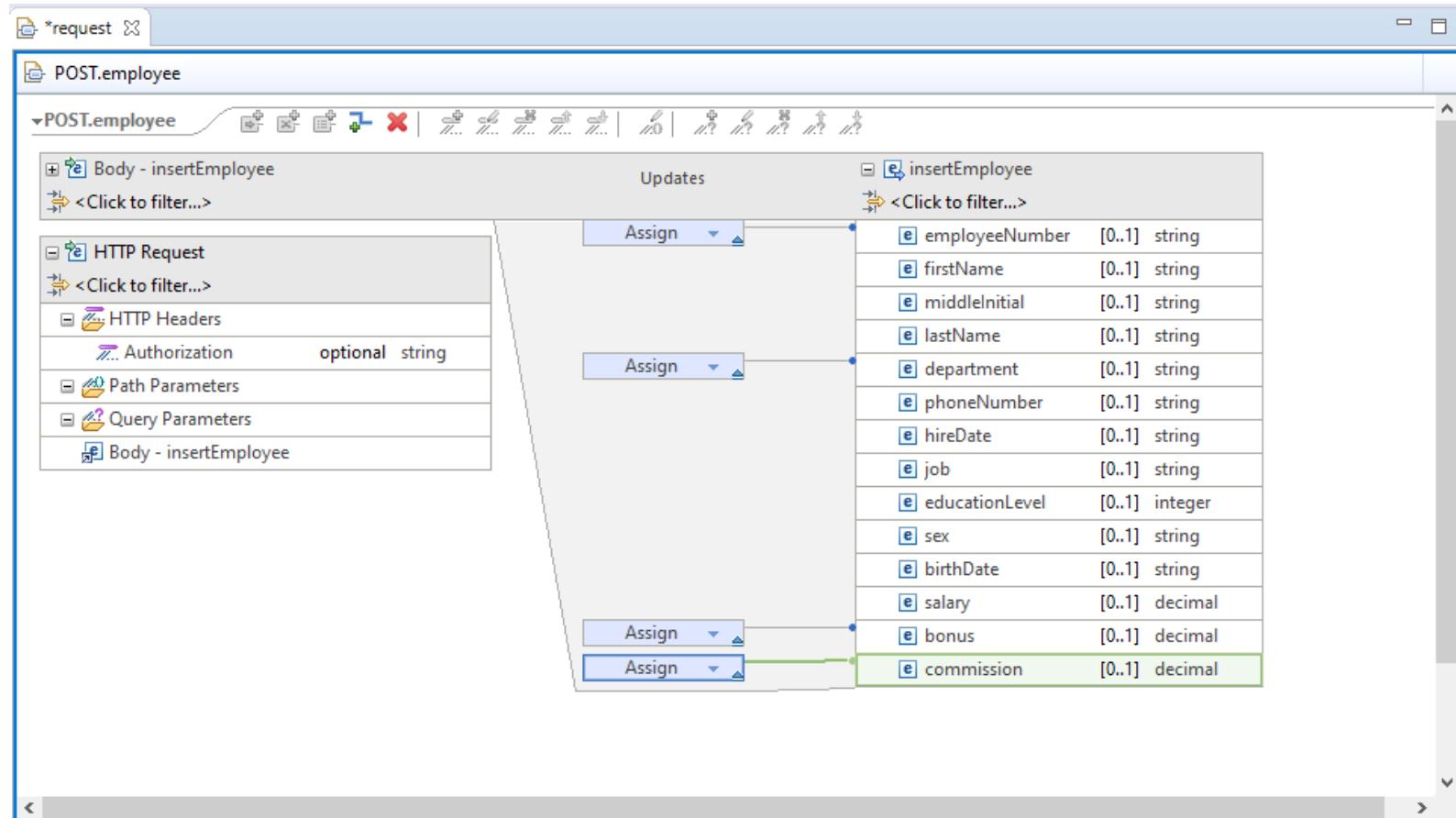
API mapping: Allows the API Developer to remove fields from the response to tailor the API





# API toolkit – API Editor and Db2 REST service

API mapping: Remove/Assign values columns exposed in Db2 REST service





# 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. It lists two headers: "Authorization" (optional string) and "ibm-mq-md-correlID" (optional string). 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, the API path is set to `/employee/{employee}`. The main panel displays the `Methods (4)` section for a `GET` operation named `cscvincSelectService`. Under the `Responses (2)` section, two responses are defined: `404 Not Found` and `200 OK`. A modal window titled `Edit Response 404` is open, showing the response code as `404 - Not Found` and the description as `Not Found`. Below this, rules are defined to indicate when to use this response code:

Rule	Condition	Value	Operator	Value	Logic
Rule 1	<code>se/Container1/RESPONSE_CONTAINER/CEIBRESP</code>	<code>=</code>	<code>13</code>	<code>AND</code>	
Rule 2	<code>:/Container1/RESPONSE_CONTAINER/CEIBRESP2</code>	<code>=</code>	<code>80</code>	<code>OR</code>	

At the bottom of the modal, the 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.

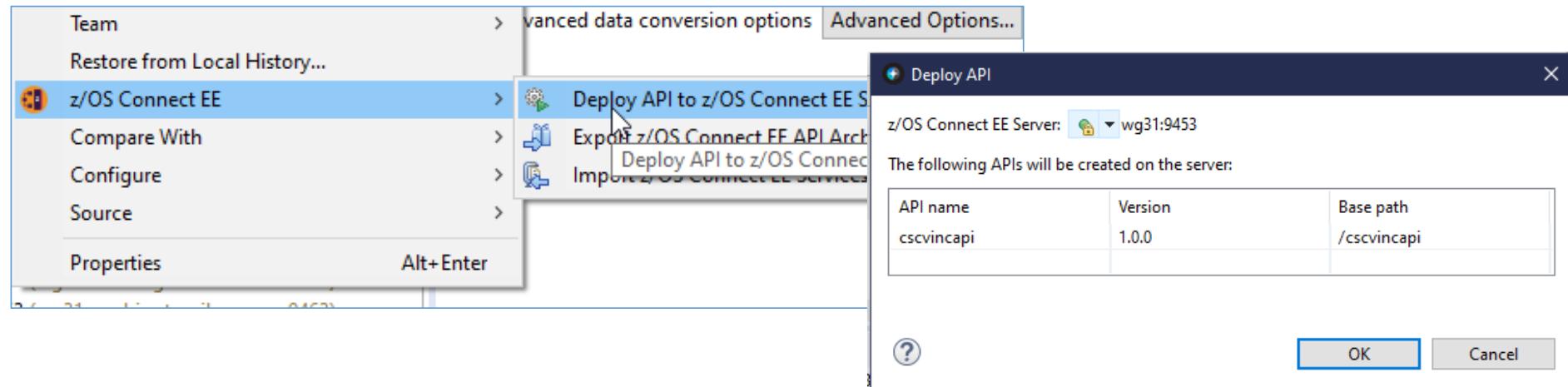
You can define rules based on fields in the service's return interface to tell z/OS Connect EE which response code to return



# 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}.



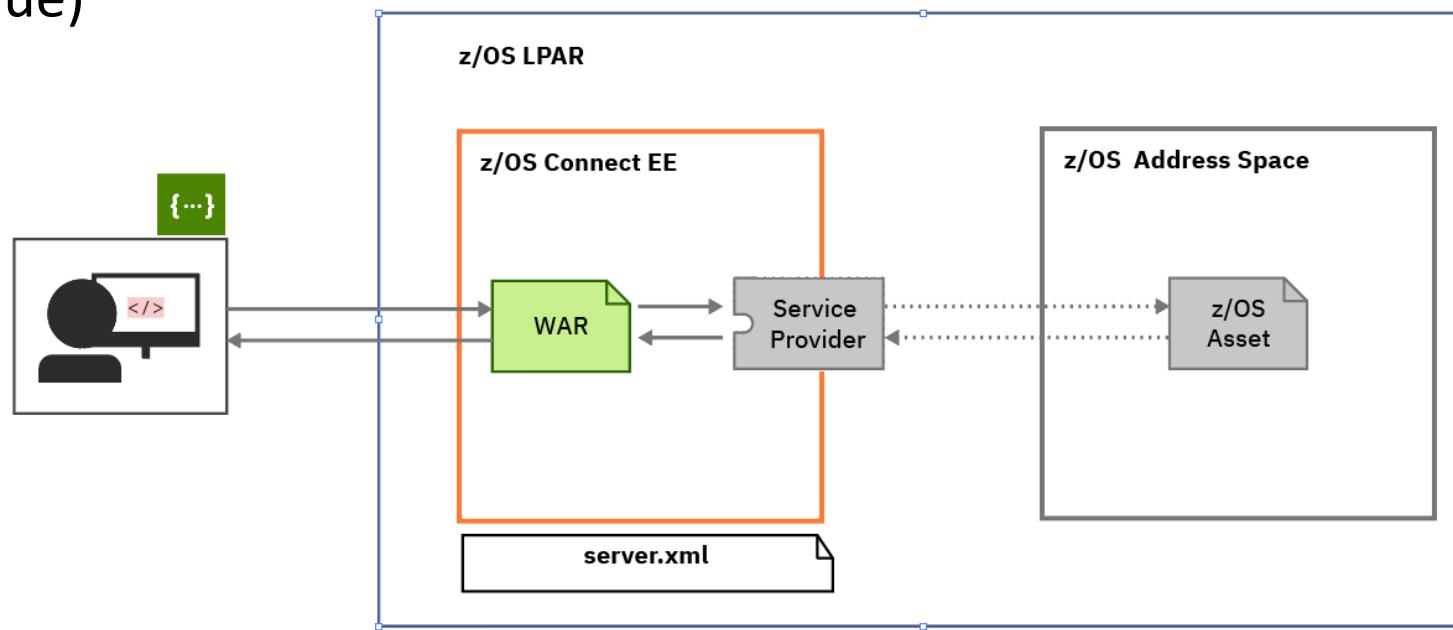
## **/zosConnect/designer**

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)



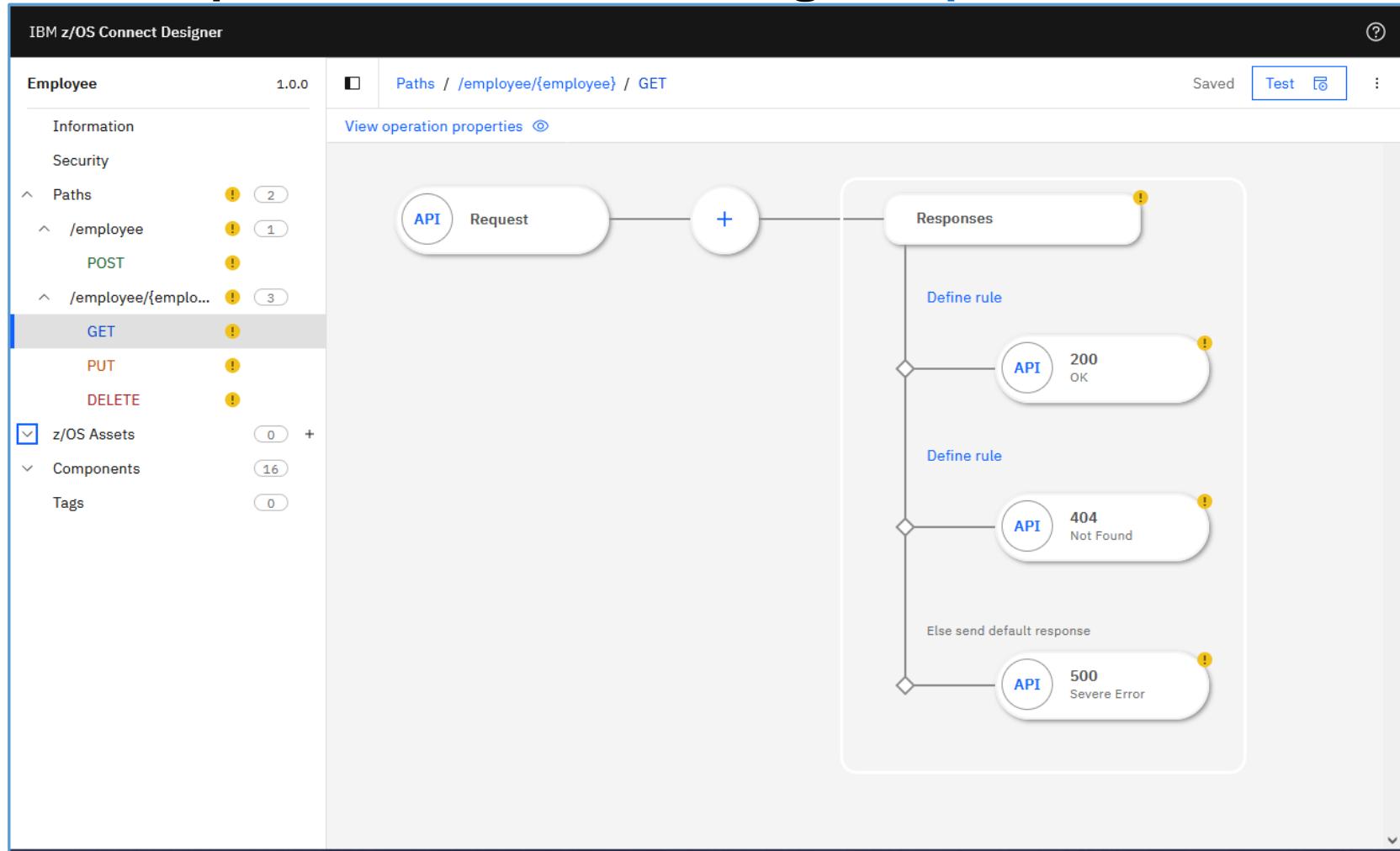
## Accessing a z/OS asset starts with a description of an API (OpenAPI 3)

```
File Edit View
"/employee/{employee}":
  get:
    tags:
      - Employee
    operationId: getEmployeeSelectService
    x-ibm-zcon-roles-allowed:
      - Staff
    parameters:
      - name: Authorization
        in: header
        required: false
        schema:
          type: string
      - name: employee
        in: path
        required: true
        schema:
          type: string
          maxLength: 6
    responses:
      "200":
        description: OK
        content:
          application/json:
            schema:
              $ref: "#/components/schemas/getEmployeeSelectService_response_200"
      "404":
        description: Not Found
        content:
          application/json:
            schema:
              $ref: "#/components/schemas/getEmployeeSelectService_response_404"
      "500":
        description: Severe Error
        content:
          application/json:
            schema:
              $ref: "#/components/schemas/getEmployeeSelectService_response_500"
  put:
    tags:
      - Employee
```

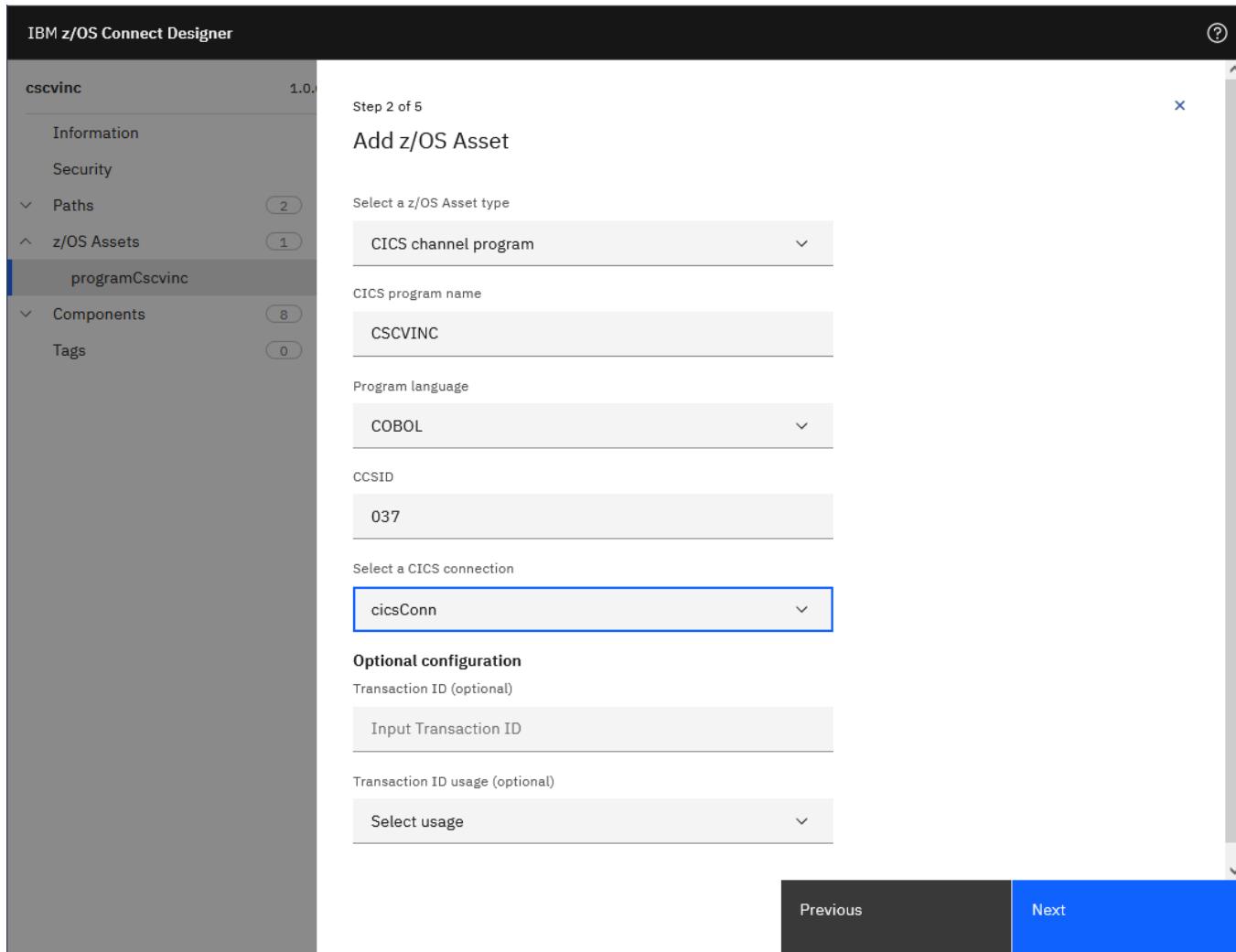
```
File Edit View
getEmployeeSelectService_response_200:
  type: object
  properties:
    summary:
      $ref: '#/components/schemas/getEmployeeSelectService_response_200_message'
    detail:
      $ref: '#/components/schemas/getEmployeeSelectService_response_200_detail'
getEmployeeSelectService_response_200_message:
  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
```



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



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





# Or by accessing an 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 by accessing a Db2 REST service (OpenAPI 3)

IBM z/OS Connect Designer

EmployeesApi 1.1

Information Security

Paths /employees/{id} 2 3

GET PUT DELETE

/employees 2

GET POST

z/OS Assets addEmployee deleteEmployee getEmployee getEmployees selectByRole updateEmployee 6 + <

Components 16

Tags 0

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

The screenshot shows the 'Add z/OS Asset' dialog in the IBM z/OS Connect Designer. The left sidebar lists the 'EmployeesApi' project with various endpoints and assets. The main dialog is titled 'Step 3 of 4' and 'Add z/OS Asset'. It prompts for a 'Db2 connection' (set to 'db2Conn') and provides fields to 'Import from Db2 service manager' (with a placeholder 'e.g. SYSIBMSERVICE'), 'Db2 native REST service name' (placeholder 'e.g. myService'), and 'Db2 native REST service version (optional)' (placeholder 'e.g. V1'). Below these are sections for 'Import Db2 native REST service request schema' and 'Import Db2 native REST service response schema', each with a 'Drag and drop or select a file' area and a note about JSON schema support. A 'Specify a URL' field contains 'http://github.com/example/api-docs' and an 'Import file' button. Navigation buttons 'Previous' and 'Next' are at the bottom.



# 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

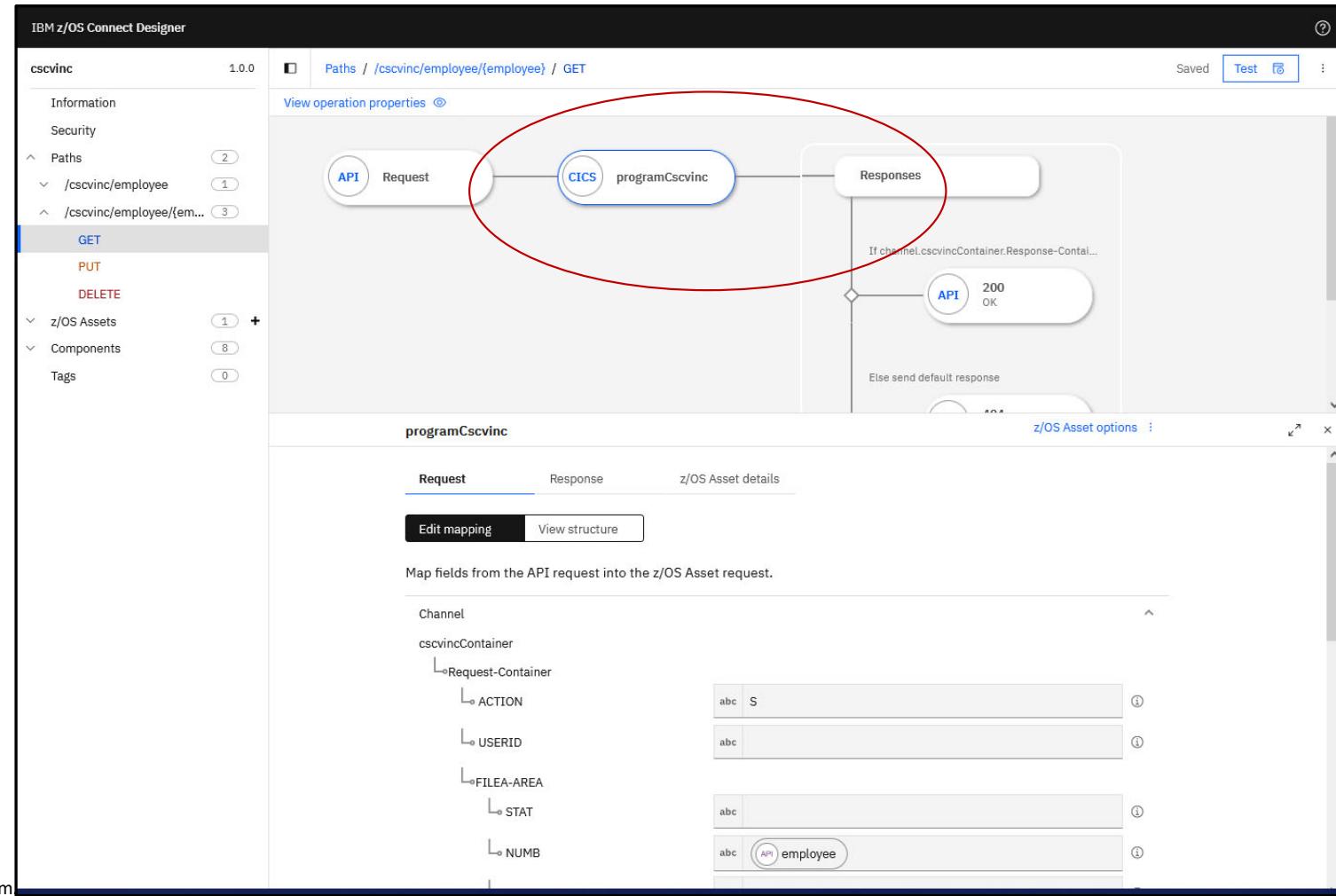
Service name	Version	Collection ID	Path	Description	Status
addEmployee	V1	SYSIBMSERVICE	/services/SYSIBMSERVI...	Add the details of an ind...	Available
deleteEmployee	V1	SYSIBMSERVICE	/services/SYSIBMSERVI...	Remove the details of a...	Available
getEmployee	V1	SYSIBMSERVICE	/services/SYSIBMSERVI...	Get 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

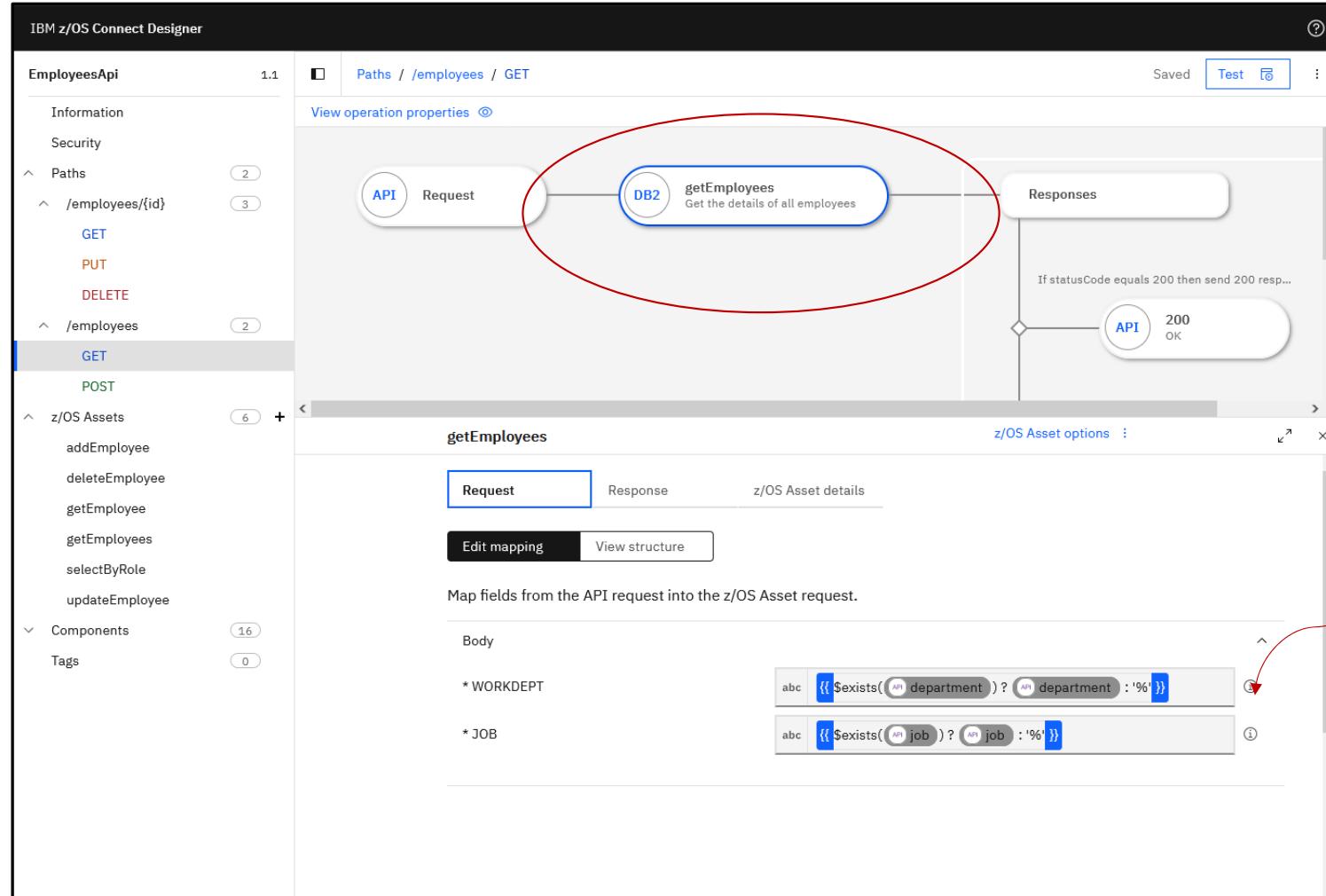
Previous Import

1 ▾ of 2 pages

# Map the method and request message with the resource's message (OpenAPI<sup>®</sup>)

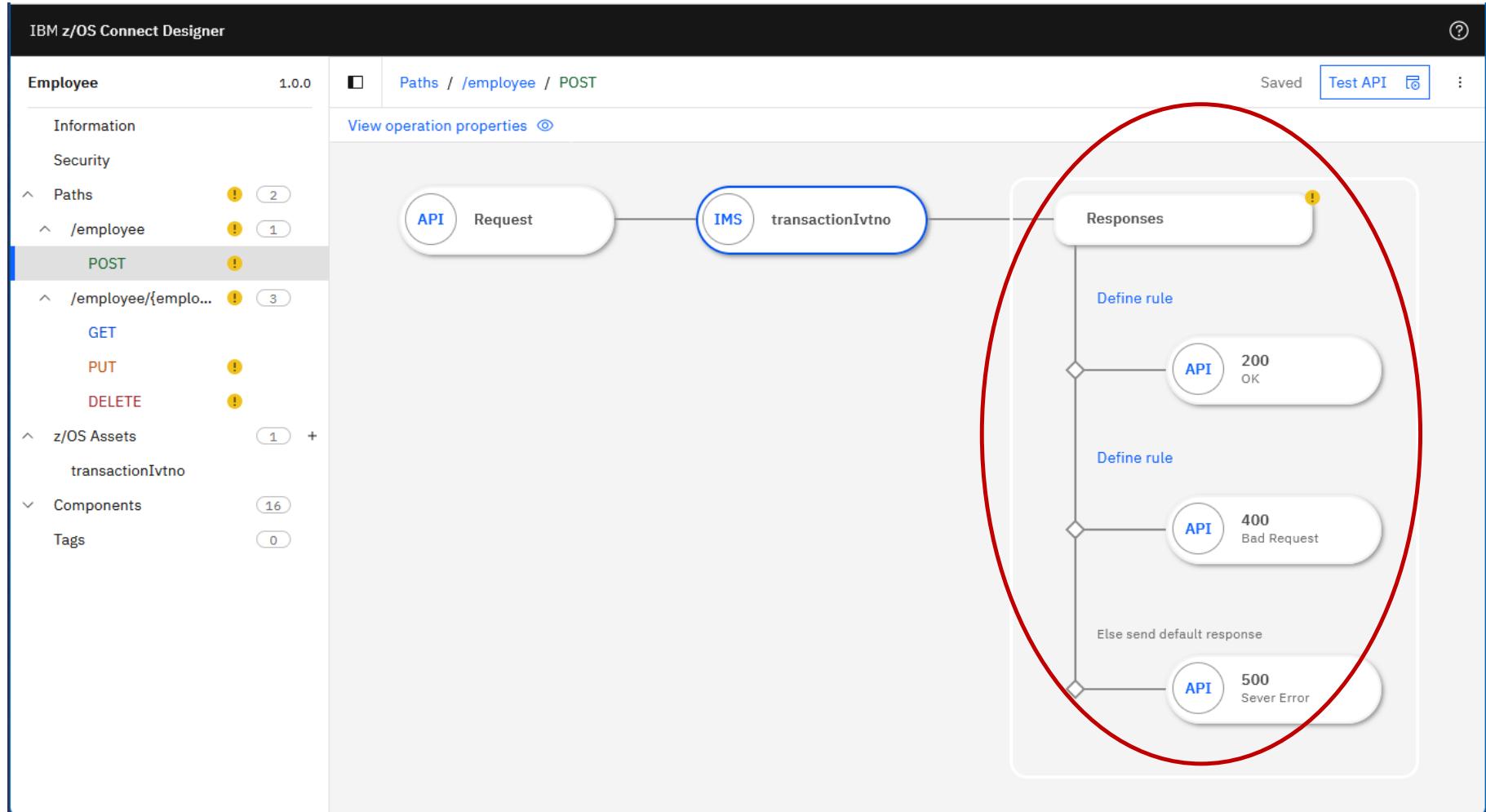


# Map the method and the response message with a Db2 REST service (OpenAPI 3)





# Map the resource's results to the specification's response (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)

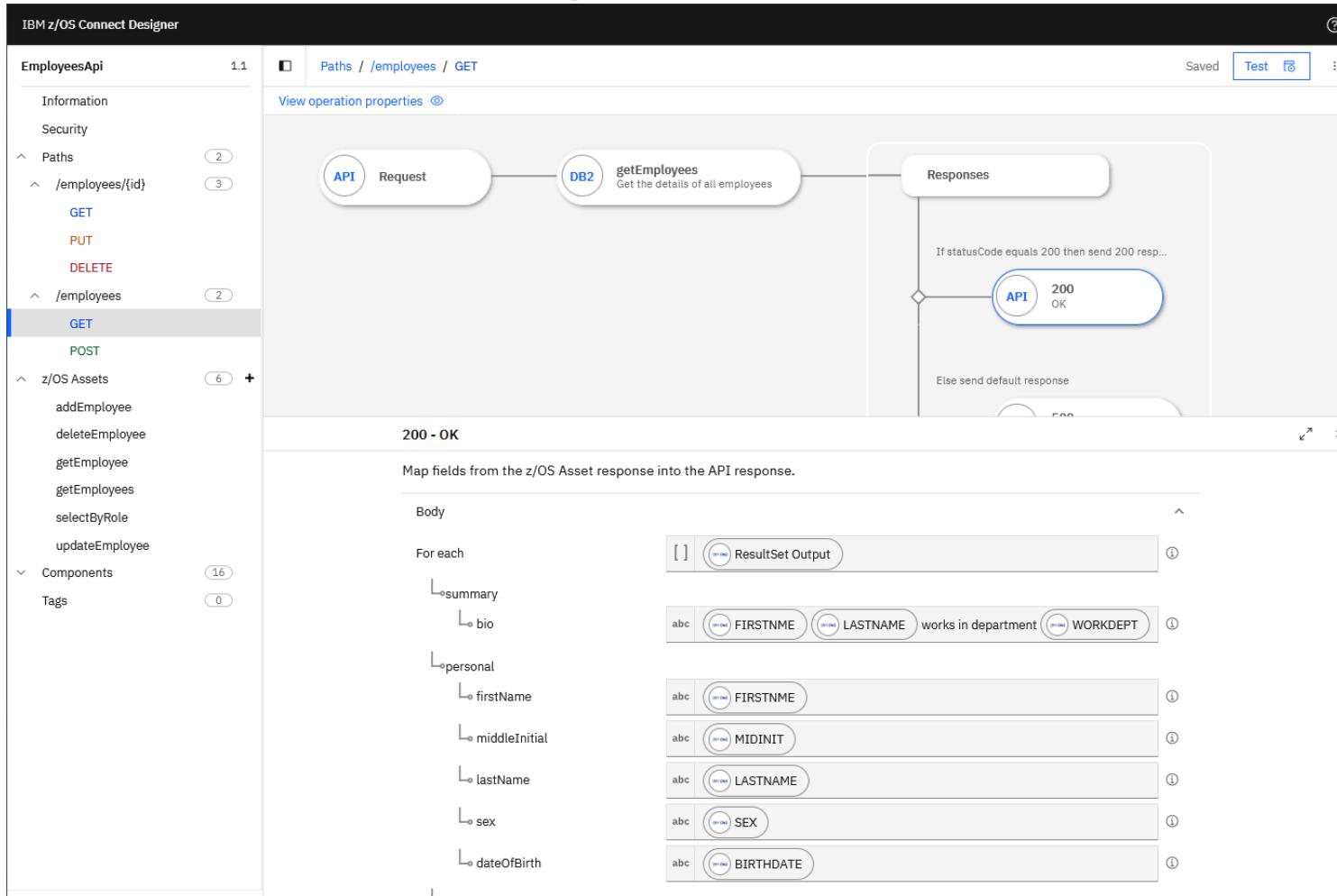
abc (COMMENT)



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 named **getEmployees**, which retrieves "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 [ ] **ResultSet Output**
  - 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.

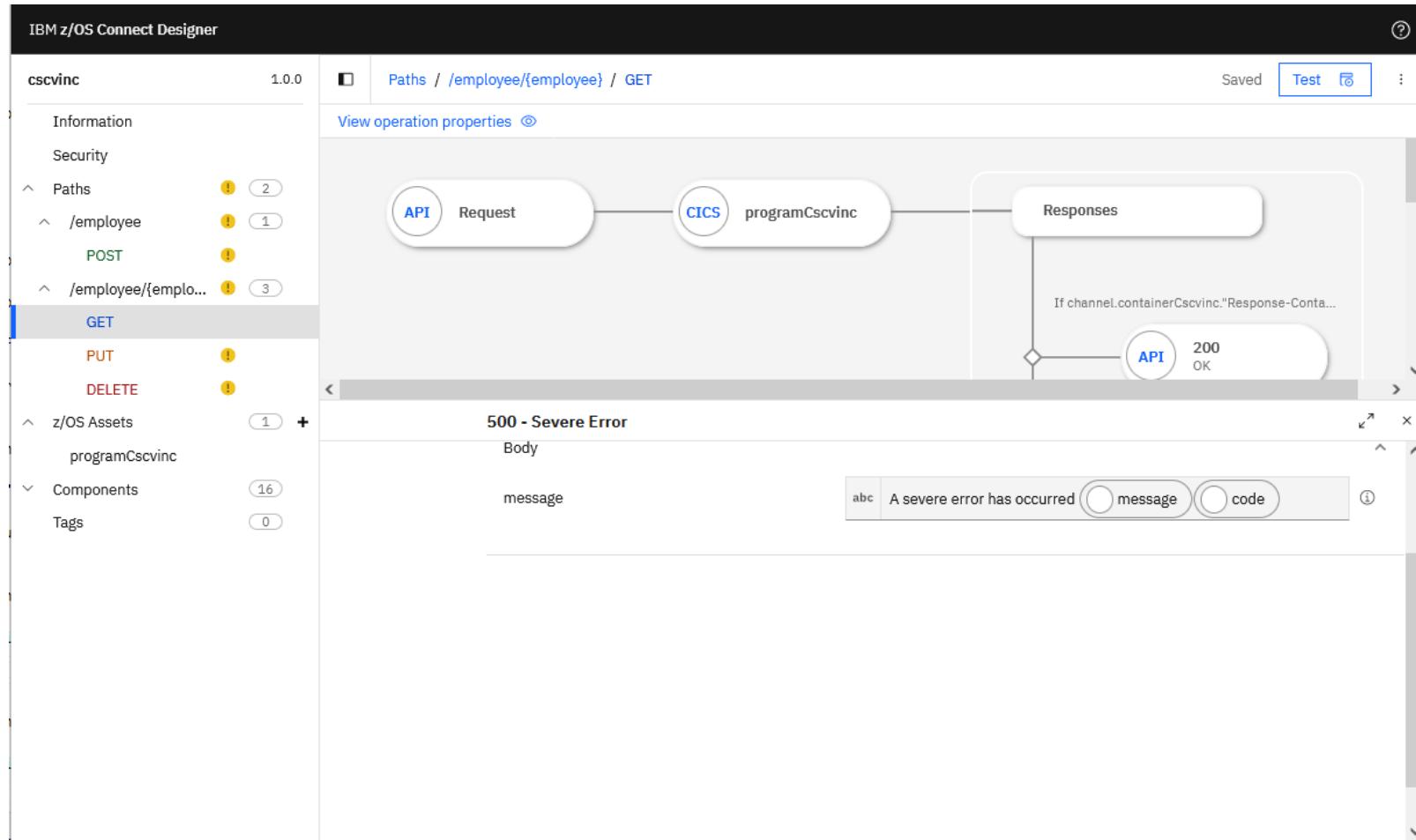
Body

message abc Employee not found ⓘ

```
graph TD; Start(( )) --> If{If "Update Count" equals 0}; If --> 200[200 Updated]; If --> Else{Else send default response}; Else --> 404[404 Not Found]; Else --> 500[500 Internal Server Error];
```

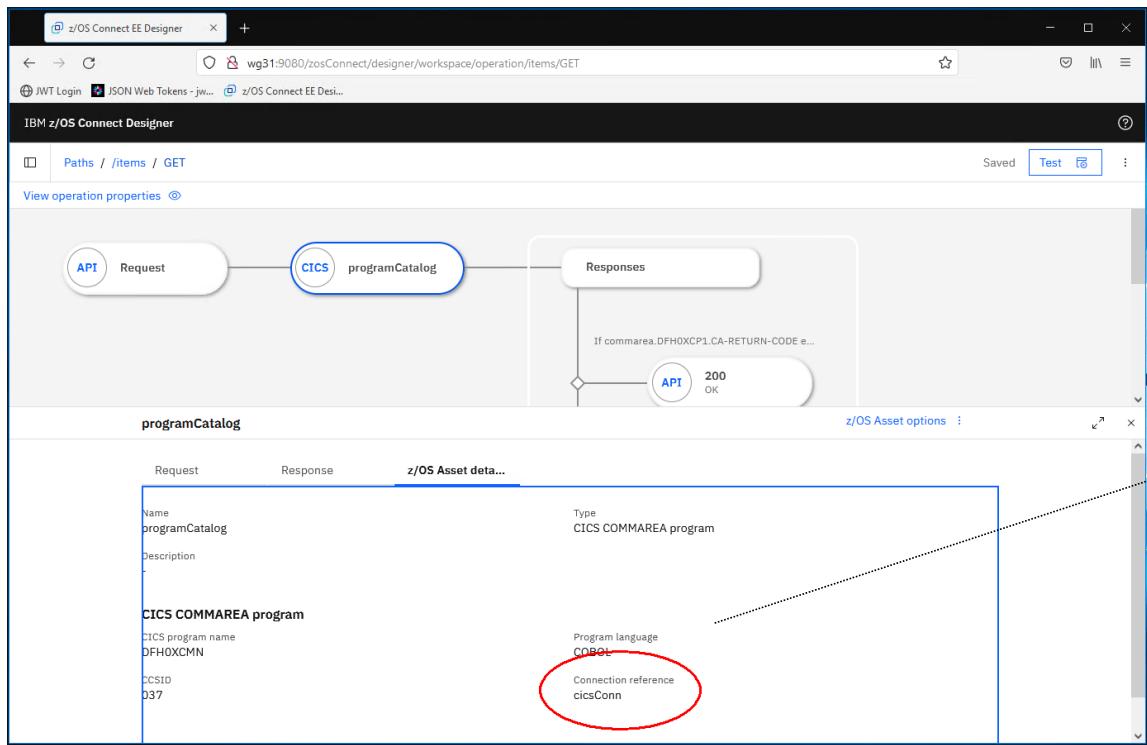


# z/OS Connect Designer for OpenAPI 3 (500)





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



The screenshot shows the 'Server Config' interface with the 'cics.xml' configuration file open. It has tabs for 'Design' and 'Source'. The 'Source' tab displays the XML code for the server configuration.

```
<server description="CICS IPIC connections">
<!-- Enable features -->
<featureManager>
<feature>zosconnect:cics-1.0</feature>
</featureManager>
<zosconnect_cicsIpicConnection id="cicsConn" host="${CICS_HOST}"
port="${CICS_PORT}" />
</server>
```

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. The 'z/OS Asset options' tab is selected, showing details for an 'IMS transaction' named 'transactionIvtno'. The 'Connection profile' dropdown is set to 'imsConn', which is highlighted with a red oval. On the right is the 'Server Config' window for 'ims.xml'. It shows the XML code for a server configuration:

```
1<?xml version="1.0" encoding="UTF-8"?>
2<server description="zosconnect_imsConnection" >
3<zosconnect_imsConnection id="imsConn" >
4    connectionFactoryRef="imsConnectionFactory"
5    imsDatastoreName="${IMS_DATASTORE}"/>
6<connectionFactory id="imsConnectionFactory" >
7    containerAuthDataRef="IMSCredentials">
8        <properties.gmoa hostName="${IMS_HOST}" portNumber="${IMS_PORT}" />
9</connectionFactory>
10<authData id="IMSCredentials" user="${IMS_USER}" >
11    password="${IMS_PASSWORD}" />
12</authData>
13</server>
14|
```

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 an API request to a 'getEmployee-V1' endpoint, which then triggers a 'Responses' block. This block contains logic to check if a specific return code is returned, leading to an 'OK' response (HTTP 200). Below the diagram, the 'getEmployee-V1' service details are shown, including its name, type (Db2 native REST service), version (V1), and connection reference ('db2Conn'). A red arrow points from this connection reference to the 'zosconnect\_db2Connection' element in the db2.xml configuration file.

The screenshot shows the 'Server Config' interface with the 'db2.xml' configuration file open. The 'Source' tab displays the XML code for the server definition. A red arrow points from the 'zosconnect\_db2Connection' element in the XML to a callout box containing the text: 'Define connections to Db2 using variables defined in bootstrap.properties file'. The XML code includes elements like <server>, <featureManager>, <zosconnect\_credential>, and <zosconnect\_db2Connection>.

```
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.



# Provide remote access to z/OS Connect OPENAPI 2 archives files

Name	Last Modified	Size	Description
<a href="#">apis</a>	Fri Feb 19 13:46:13 GMT 2021	-	Directory
<a href="#">services</a>	Sat Feb 20 20:54:41 GMT 2021	-	Directory
<a href="#">apiRequesters</a>	Wed Feb 07 17:59:04 GMT 2018	-	Directory
<a href="#">rules</a>	Tue Jan 26 20:34:05 GMT 2021	-	Directory

```
<webApplication  
    id="resources-location" name="resources"  
    location="${server.config.dir}/resources/zosconnect">  
    <web-ext context-root="/resources/zosConnect"  
        enable-file-serving="true"  
        enable-directory-browsing="true">  
        <file-serving-attribute name="extendedDocumentRoot"  
            value="${server.config.dir}/resources/zosconnect"/>  
    </web-ext>  
</webApplication>
```

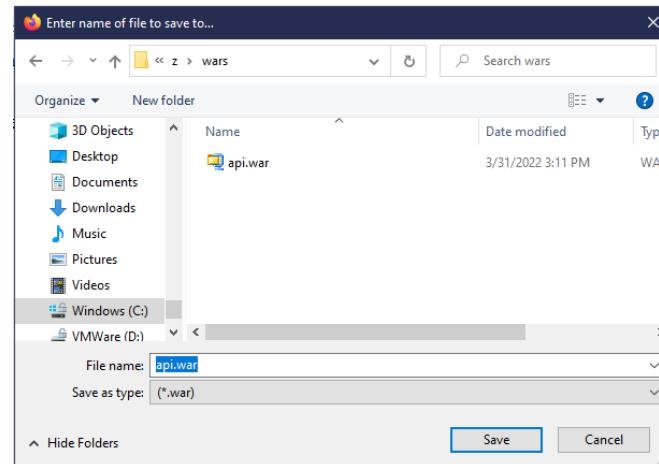
Name	Last Modified	Size	Description
<a href="#">cscvincDeleteService.sar</a>	Thu Feb 18 18:02:19 GMT 2021	4362	File
<a href="#">cscvincInsertService.sar</a>	Thu Feb 18 18:02:19 GMT 2021	4491	File
<a href="#">cscvincSelectService.sar</a>	Thu Feb 18 18:02:19 GMT 2021	4590	File

# Provide remote access to z/OS Connect Designer OPENAPI 3 web archives files

The screenshot shows a web browser window with the URL <https://localhost:9445/dropins/>. The title bar says "Index of /dropins/". The page content is titled "Index of /dropins/" and contains a table with two rows:

Name	Last Modified	Size	Description
<a href="#">EmployeesApi.war</a>	Tue May 03 22:36:07 UTC 2022	26394	File
<a href="#">api.war</a>	Wed May 04 12:33:45 UTC 2022	15227	File

```
<webApplication id="resources-location" name="resources"
location="/opt/ibm/wlp/usr/servers/defaultServer/dropins">
<web-ext context-root="dropins"
enable-file-serving="true" enable-directory-browsing="true">
<file-servering-attribute name="extendDocumentRoot"
value="/opt/ibm/wlp/usr/servers/defaultServer/dropins" />
</web-ext>
</webApplication> >
```





## 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*.



# z/OS Server Issues and Considerations – API Deployment

<https://www.ibm.com/docs/en/zos-connect/zos-connect/3.0?topic=server-devops-overview>

Graduate plug-in and its requirements, see [Using graduate with z/OS Connect](#).

**Important:** In order for multiple API project .war files to function when deployed to a single z/OS Connect native server, you must ensure that your OpenAPI specification includes a contextRoot defined within the servers section. The contextRoot attribute specifies the entry point of the deployed application. For example, to use a context root of /myContextRoot in the API's OpenAPI definition server's section:

```
openapi: 3.0.0
...
servers:
  url: "https://localhost:9443/myContextRoot"
...
```

This definition must match the contextRoot attribute value of the webApplication element in your server.xml file. An example of the configuration might be:

```
<webApplication location="${server.config.dir}/apps/api.war" name="EmployeesApi" contextRoot="/myContextRoot"/>
```

If the server entry in the server section of the OpenAPI definition includes a contextRoot value, then this value must be specified in the contextRoot attribute of the corresponding webApplication element, even when only a single API is deployed to the z/OS Connect server.

Each API deployed to the same IBM z/OS Connect server requires a unique context root.

4. Copy the server configuration.  
Copy the server configuration files from the API project /scr/main/liberty/config directory into the \${server.config.dir}/configDropins/overrides directory of the server or another directory via FTP. For more information, see [Overview of IBM z/OS Connect Server configuration](#)

5. Deploy the generated API files to the z/OS Connect native server  
Deploy API .war files into a permanent USS directory. An example directory, such as the one provided by the z/OS Connect native server template, is \${server.config.dir}/apps. API files can also be deployed to other directories, such as in separately mounted zFS file systems.

For each deployed API define a webApplication element in the configuration file. An example element is included in the supplied openApi3 server template. An example element might be the following:

```
<webApplication id="My API" location="${server.config.dir}/apps/api.war" name="MyAPI"/>
```



# z/OS Server Issues and Considerations – Context Root

<https://www.ibm.com/docs/en/zos-connect/zos-connect/3.0?topic=image-devops-overview>

## The drop-ins directory

The *drop-ins* directory, `/config/dropins` is a special directory that is supported by WebSphere® Application Server for Liberty. It allows `.war` files to be deployed and dynamically loaded into the running IBM z/OS Connect with no additional definitions that are required in the configuration file.

By default, z/OS Connect Designer deploys the API `.war` file to this directory. Using the same directory in your API container image simplifies the creation of that image because the configuration remains the same.

## A directory other than drop-ins

This is required in any of the following situations:

- The API's OpenAPI definition server's section contains server entry that includes a context root value, which is not just `/`.
- Multiple APIs are to be deployed to the same IBM z/OS Connect container. Because the API `.war` file will be generated with a context root of `/`, and multiple API `.war` files in the same server must have unique context root values.

You need to include a context root value (not `/`) in the API's OpenAPI definition server's section, for example to use a context root of `/MyCompany`:

```
openapi: 3.0.0
...
servers:
  url: https://localhost:9443/MyCompany
  ...
```

- Requests to start an API require authentication only, without authorization, so the authorization roles need to be mapped to the WebSphere Application Server for Liberty special subject `ALL_AUTHENTICATED_USERS`. For more information, see [How to define authorization roles](#).

If you choose not to use the drop-ins directory, you must alter the configuration that is used in z/OS Connect Designer during the creation of the API container image.

- **Required to define applications and add context root**

```
<webApplication id="cics" contextRoot="/cics" name="cicsAPI"
  location="${server.config.dir}apps/cscvinc.war"/>
<webApplication id="db2" contextRoot="/db2" name="db2API"
  location="${server.config.dir}apps/employees.war"/>
```



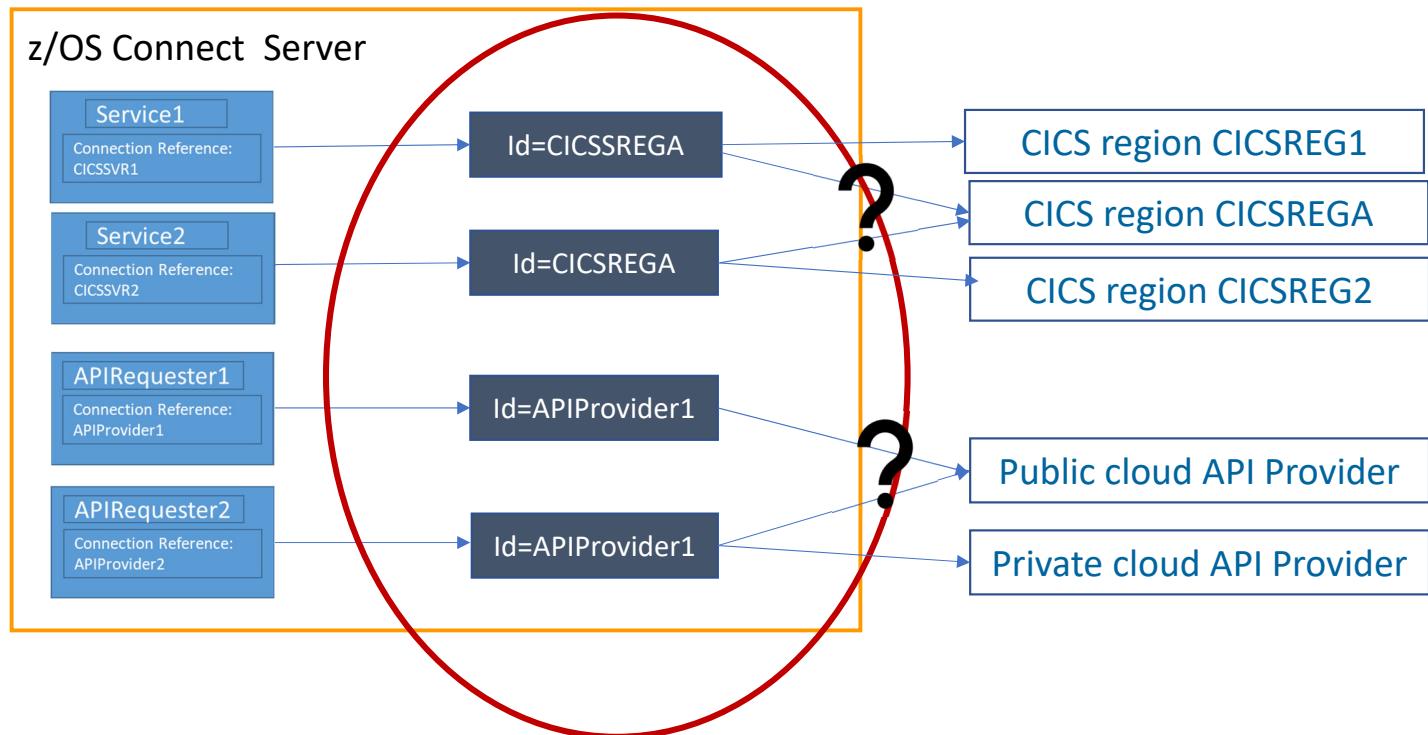
## Connection Reference Consideration

Carefully consider the names used for connections



# Use naming conventions for connection references

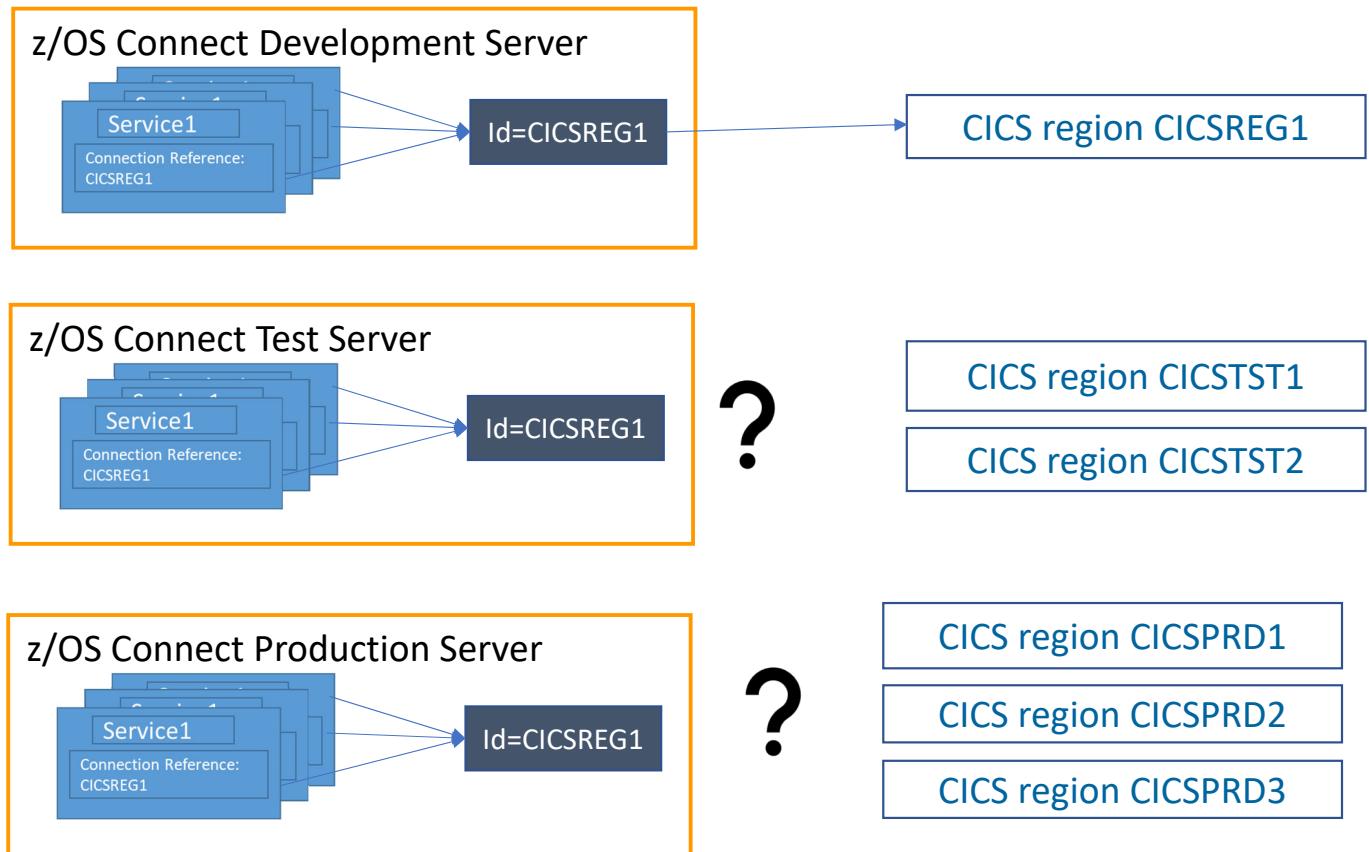
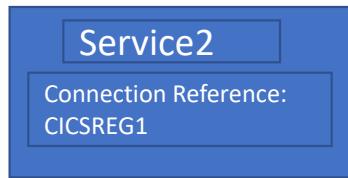
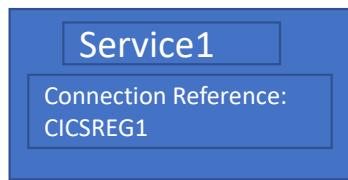
Use application meaningful names or an extendable convention for connection reference names





## 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 shows a list of requests. One request is selected, showing a GET method and the URL https://mpz3.washington.ibm.com:9443/cscvinc/employee/111111. The request tab is expanded, showing 'Params' (which is currently active), Authorization, Headers (10), Body, Pre-request Script, Tests, and Settings. Under 'Params', there's a table for 'Query Params' with columns for KEY, VALUE, DESCRIPTION, and Bulk Edit. The Body tab is selected, displaying the JSON response:

```
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": "511212003",  
12                 "date": "26 11 81",  
13                 "amount": "00011 00"  
14             }  
15         }  
16     }  
17 }
```

Below the response, there are tabs for Body, Cookies (1), Headers (8), and Test Results. The status bar at the bottom shows Status: 200 OK, Time: 205 ms, Size: 899 B, and a Save Response button. The bottom navigation bar includes 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

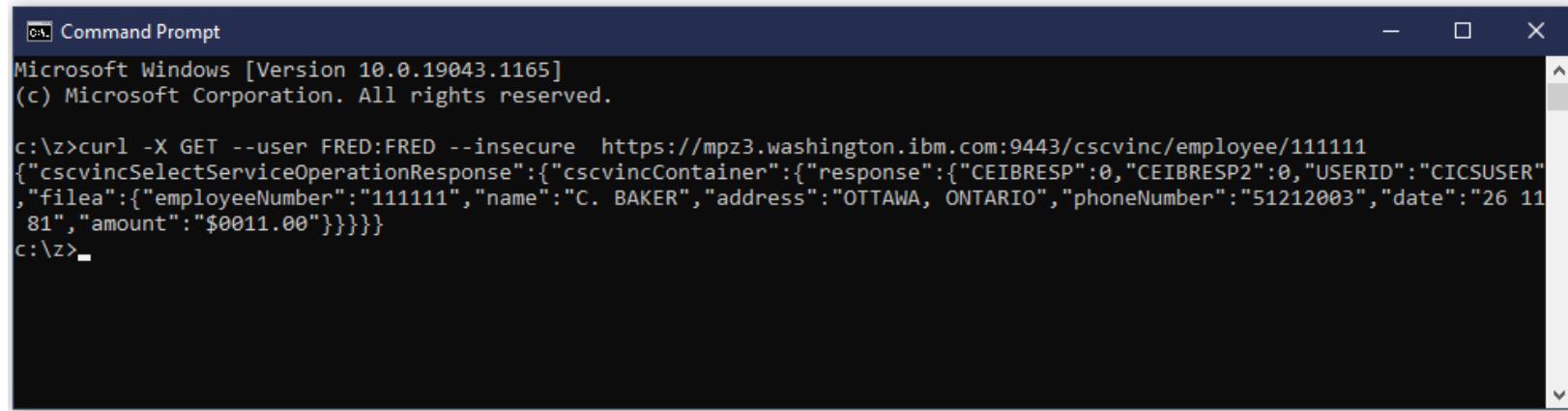
Discover REST APIs available within Liberty

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 or in scripts, bat or command 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  
curl -X GET -user user1:user1 --insecure  
https://%TARGET%:9483/zosConnect/cscvinc/employee/000100  
echo ''
```

Stop an API

Delete or remove the API from the server

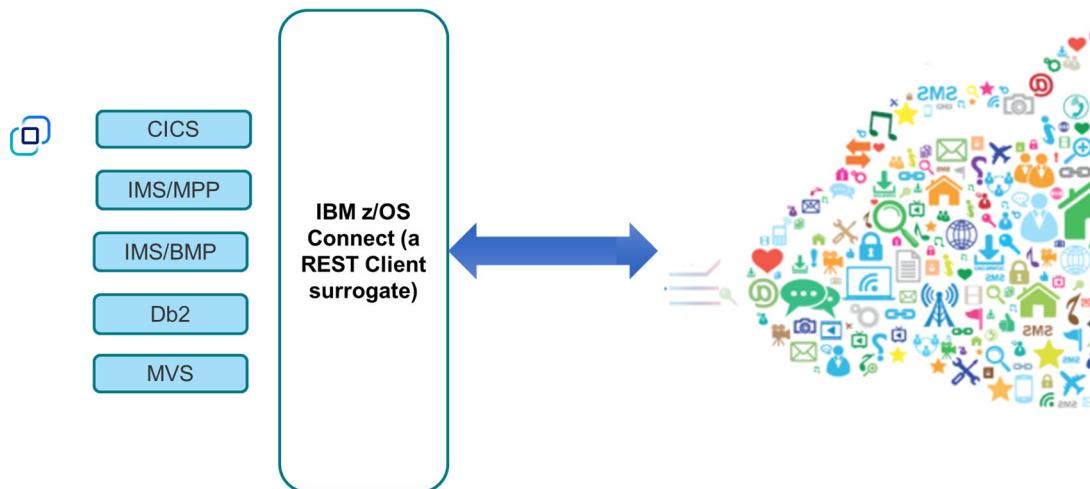
Deploy the API to the server

Execute the API



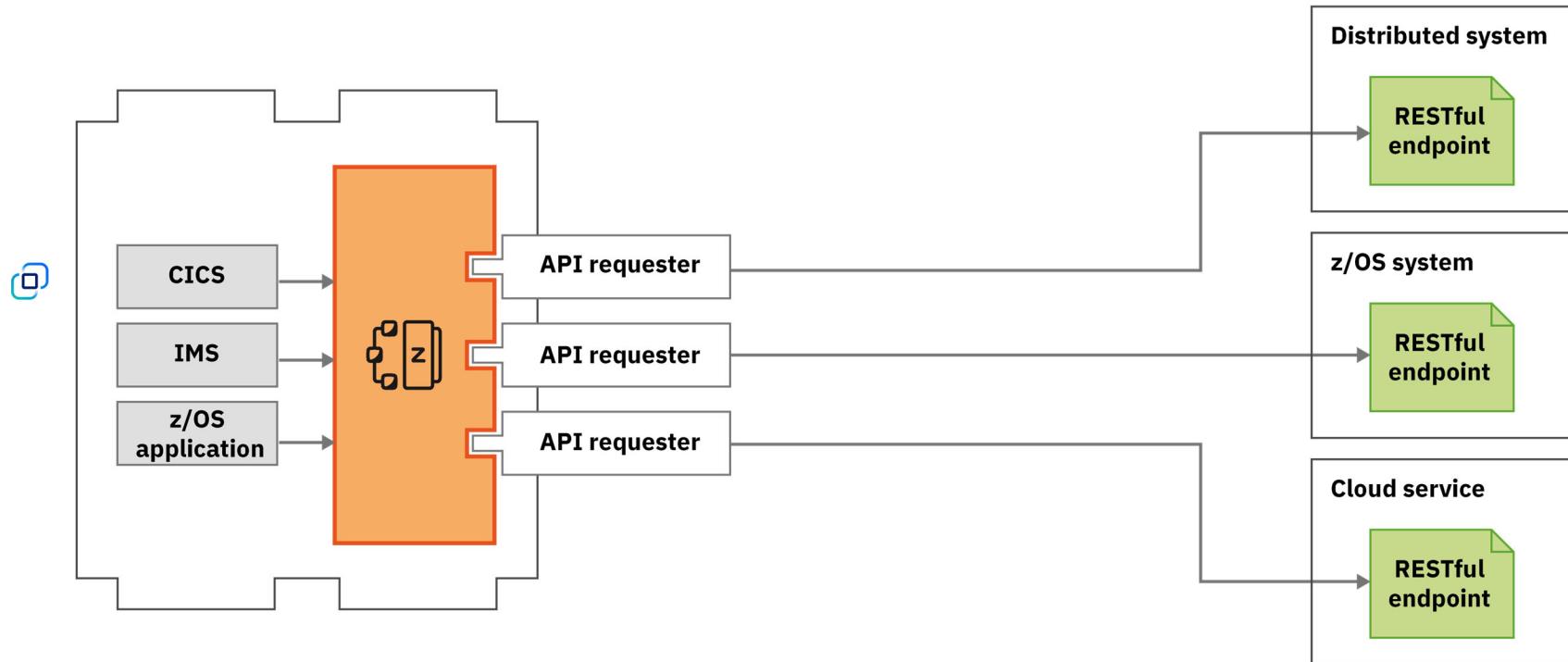
## /api\_toolkit/apiRequesters

Quick and easy API mapping.



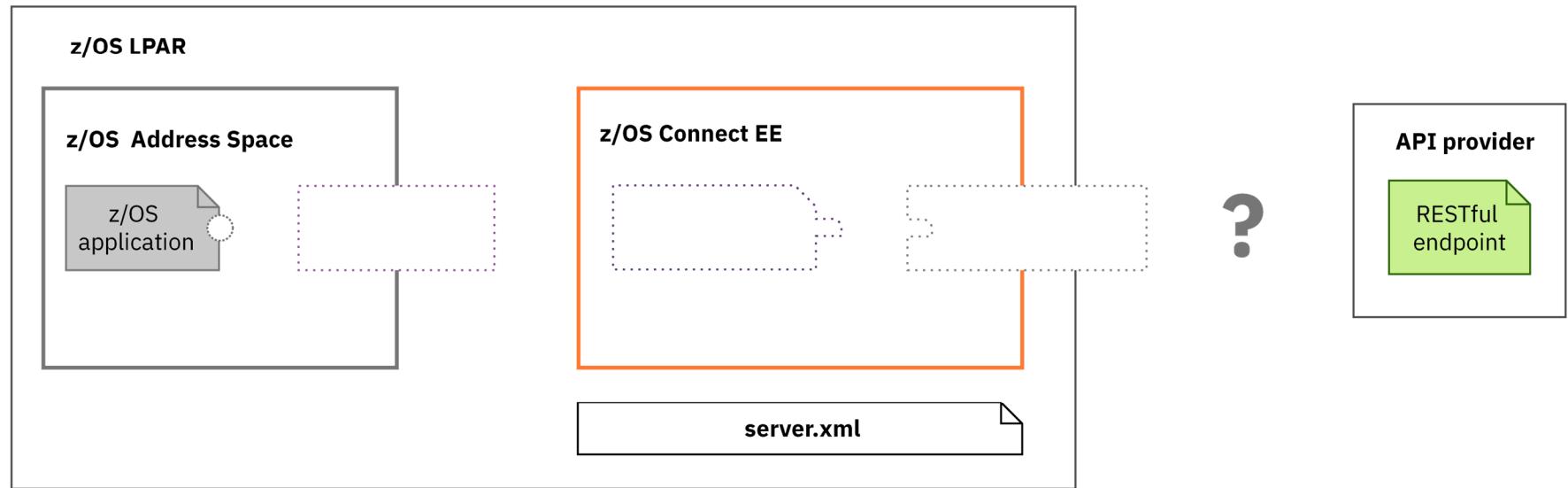


# Use API requester to call external APIs from z/OS assets





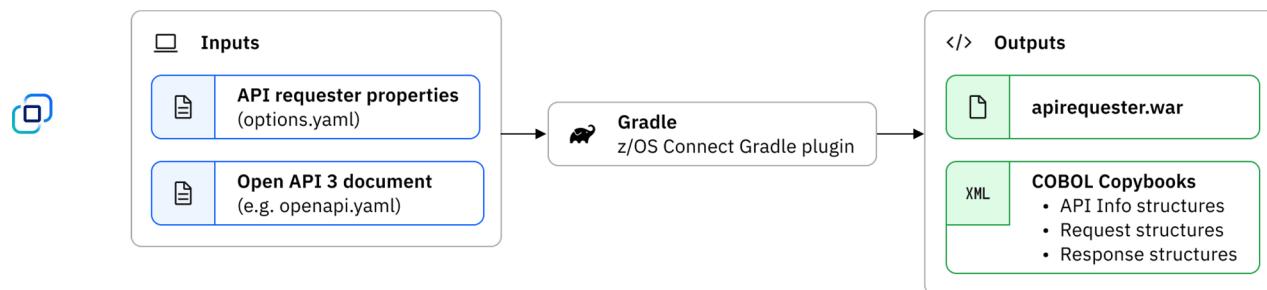
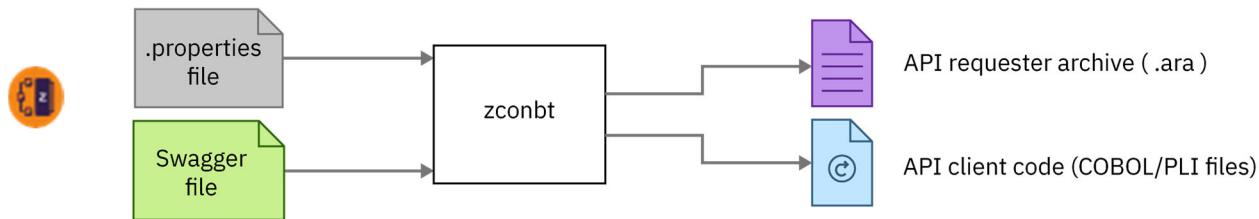
# Calling an external API





# Calling an external API starts with the specification document

- Use the API's specification and generate COBOL copy books for the API requester archive file for an OpenAPI2 API or a web archive file for an OpenAPI3 API

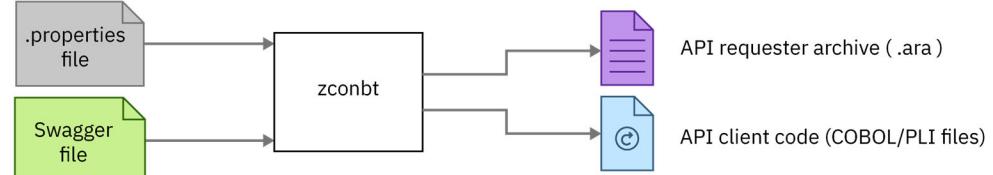




# For an OpenAPI2 API, use the zconbt tooling

The screenshot shows a JSON editor window displaying an OpenAPI2 schema. The schema defines an endpoint for retrieving an employee by ID, with parameters for operation ID and path. The JSON structure includes definitions for the employee entity and various response codes (200, 404).

```
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>



# Generates an API requester archive and COBOL copybooks

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.



# For an OpenAPI3 API, use Gradle

The screenshot shows a code editor window titled "PhonebookApi.yaml". The file content is an OpenAPI3 specification:

```
openapi: 3.0.0
info:
  title: Phonebook
  description: Manage phonebook contacts through an API for IMS.
  version: '2.0'
  license:
    name: Apache-2.0
    url: https://opensource.org/licenses/Apache-2.0
servers:
- url: /
security:
- BasicAuth: []
- BearerAuth: []
paths:
  /phonebook/contacts:
    post:
      tags:
        - Contacts
      summary: Add a contact to the phonebook
      description: Uses the phonebook IMS Transaction z/OS asset
      operationId: phonebookContactsPost
      requestBody:
        description: The contact to add to the phonebook.
        required: true
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/Contact'
      responses:
        '201':
          description: Created
          content:
            application/json:
              schema:
                $ref: '#/components/schemas/NormalResponse'
        '409':
          description: Conflict
          content:
            application/json:
              schema:
                ...

```

Ln 1, Col 1 | 100% | Unix (LF) | UTF-8



## Gradle plug-in properties and options<sup>#</sup>

```
apiKeyMaxLength=255
characterVarying=NO
defaultRequestMediaType="application/json"
defaultResponseMediaType="application/json"
language=COBOL
operations=GetEmployee,UpdateEmployee
connectionRef=cscvincAPI
requesterPrefix=csc
```

<sup>#</sup>Additional property file attributes, e.g., `defaultCharacterMaxLength`, `inlineMaxOccursLimit`, etc. are described at **The API requester Gradle plug-in properties and options** article at URL <https://www.ibm.com/docs/en/zos-connect/zos-connect/3.0?topic=requester-gradle-plug-in-properties-options>



# 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.
```



# COBOL working storage implications

Specification properties are usually not constrained, this can lead to excessive working storage consumption#

A screenshot of a JSON editor showing a complex schema. The schema includes fields like communicationPreferences (array), memberCodeableConcept (object), and member-contacts-request (object). Several fields are circled in red, including communicationPreferences, memberCodeableConcept, umi, firstName, lastName, and birthDate.

```
/C:/z/apiRequester/ATS/ATSContactPreferences.json
{
  "maxItems": 10,
  "communicationPreferences": {
    "items": [
      {
        "$ref": "#/definitions/member-communication-preferences"
      }
    ],
    "type": "array"
  },
  "memberCodeableConcept": {
    "description": "Multiple Member codes",
    "items": [
      {
        "$ref": "#/definitions/member-codeable-concept"
      }
    ],
    "type": "array"
  },
  "type": "object"
},
  "member-contacts-request": {
    "title": "Member Contacts Request",
    "description": "Read-only request data to search for member contact information.",
    "properties": {
      "umi": {
        "description": "Unique Member Id. This value is at a contract level. All members under one contract have the same UMI.",
        "example": "12222444001",
        "type": "string"
      },
      "firstName": {
        "description": "Member first name or given name.",
        "example": "Arthur",
        "type": "string"
      },
      "lastName": {
        "description": "Member last name or family name.",
        "example": "Smith",
        "type": "string"
      },
      "birthDate": {
        "description": "Member date of birth in the format mm/dd/yyyy.",
        "example": "12/19/2019",
        "type": "string"
      }
    }
  }
}
```

mitchj@us.ibm.com

A screenshot of a Notepad window showing COBOL source code. The code defines several fields with the same data type: PIC S9(9) COMP-5 SYNC. These fields are circled in red, including memberContactsResponse, pin, firstName, middleName, and lastName.

```
ATS01P01 - Notepad
File Edit Format View Help
* ++++++
06 RespBody.

09 memberContactsResponse-num      PIC S9(9) COMP-5 SYNC.

09 memberContactsResponse OCCURS 255.

12 umi-num                         PIC S9(9) COMP-5 SYNC.

12 umi.
  15 umi2-length
  15 umi2
  PIC S9999 COMP-5
  PIC X(255).

12 pin-num                          PIC S9(9) COMP-5 SYNC.

12 pin.
  15 pin2-length
  15 pin2
  PIC S9999 COMP-5
  PIC X(255).

12 firstName-num                    PIC S9(9) COMP-5 SYNC.

12 firstName.
  15 firstName2-length
  15 firstName2
  PIC S9999 COMP-5
  PIC X(255).

12 middleName-num                  PIC S9(9) COMP-5 SYNC.

12 middleName.
  15 middleName2-length
  15 middleName2
  PIC S9999 COMP-5
  PIC X(255).

12 lastName-num                    PIC S9(9) COMP-5 SYNC.

12 lastName.
  15 lastName2-length
  15 lastName2
  PIC S9999 COMP-5
  PIC X(255).

Ln 761, Col 2 | 100% Windows (CRLF) | UTF-8
```

#Addressed in OpenAPI3 tooling

© 2018, 2023 IBM Corporation

Page 129



# There are API Requester generation properties are available

Use these generation properties to set default array size and string field sizes

- **defaultArrayMaxItems** - Specify the maximum array boundary to apply when no maximum occurrence information (maxItems) is implied in the Swagger. The value of this parameter can be a positive integer in the range 1 - 32767. By default, **defaultArrayMaxItems** is set to **255**.
- **inlineMaxOccursLimit** - Specifies the size limit for an array before it is upgraded to a data area. If you specify inlineMaxOccursLimit=5, and the array has three elements, it remains an array. If the array has seven elements, it is transformed into a data area instead. See HOST API.
- **defaultCharacterMaxLength** - Specify the default array length of character data in characters for mappings where no length is implied in the JSON schema document. When **characterVarying** is set to YES, the value of this parameter can be a positive integer in the range of 1 to 32767. When **characterVarying** is set to NO or NULL the value of this parameter can be a positive integer in the range of 1 to 16777214 By default, **defaultCharacterMaxLength** is set to **255**.
- **characterVarying** - Specifies how variable-length character data is mapped to the language structure.
  - NO - Variable-length character data is mapped as fixed-length strings.
  - NULL - Variable-length character data is mapped to null-terminated strings (defaultCharacterMaxLength + 1)
  - YES - Variable-length character data is mapped to a CHAR VARYING data type in PL/I. In COBOL variable-length character data is mapped to an equivalent representation that consists of two related elements: the **data-length** and the **data**. By default, **characterVarying** is set to **YES**.

12 firstName-num	PIC S9(9) COMP-5	SYNC.
12 firstName.		
15 firstName2-length	PIC S9999 COMP-5	

12 firstName-num	PIC S9(9) COMP-5	SYNC.
12 firstName	PIC X(31).	

```
MOVE 0 to ws-length
MOVE LENGTH OF firstName2 to firstName2-length.
INSPECT FUNCTION REVERSE (firstName2)
      TALLYING ws-length FOR ALL SPACES.
SUBTRACT ws-length FROM firstName2-length.
```

```
*-----*
 * Add null termination character to strings
 *-----*
 STRING firstName delimited by size
       X'00' delimited by size into _firstName.
 STRING ws-length delimited by size into _ws-length.
```



# The number of occurrences of an entry can be ambiguous

In this case 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. Review the generated COBOL and provide the number of occurrences of these variables

```
BROWSE    USER1.ZCEE.SOURCE(PUTAPI) - 01.05      Line 0000000062 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.

04 / 015
```

Connected to remote server/host wg31z using lu/pool TCP00108 and port 23



# Alternatively, consider adding constraints to the properties

Use the *maxItems* and *maxLength* attributes to set realistic maximum array and field sizes

```
JSON Raw Data Headers
Save Copy Collapse All Expand All Filter JSON
  type: "array"
    communicationPreferences:
      items:
        $ref: "#/definitions/member-communication-preferences"
        type: "array"
        maxItems: 10
        memberCodeableConcept:
          description: "Multiple member codes"
        items:
          $ref: "#/definitions/member-codeable-concept"
          type: "object"
        type: "object"
      member-contacts-request:
        title: "Member Contacts Request"
        description: "Read-only request data to search for member contact information."
        properties:
          umi:
            description: "Unique Member Id. This value is at a contract level. All members under one contract have the same UMI."
            example: "112222444001"
            type: "string"
            maxLength: 12
          firstName:
            description: "Member first name or given name."
            example: "Arthur"
            type: "string"
            maxLength: 30
          lastName:
            description: "Member last name or family name."
            example: "Smith"
            type: "string"
            maxLength: 30
```

```
* Comments for field 'filler':
* This is a filler entry to ensure the correct padding for a
* structure. These slack bytes do not contain any application
* data.
*      15 filler          PIC X(3).
*
*
* ++++++
06 RespBody.

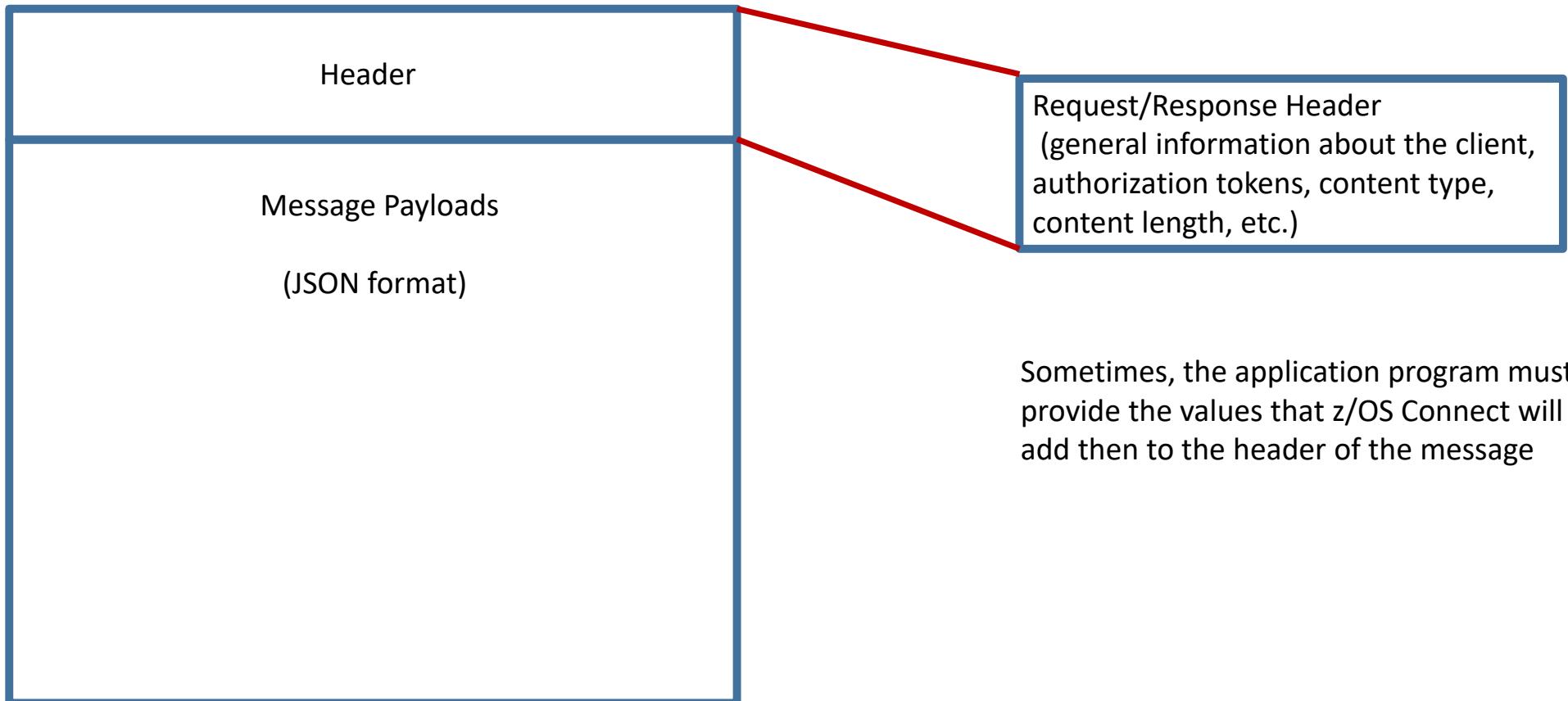
09 memberContactsResponse-num    PIC S9(9) COMP-5 SYNC.

09 memberContactsResponse OCCURS 10.

12 umi-num                      PIC S9(9) COMP-5           SYNC.
12 umi.
15 umi2-length
15 umi2
12 pin-num                       PIC S9(9) COMP-5           SYNC.
12 pin.
15 pin2-length
15 pin2
12 firstName-num                  PIC S9(9) COMP-5           SYNC.
12 firstName.
15 firstName2-length
15 firstName2
12 middleName-num                PIC S9(9) COMP-5           SYNC.
12 middleName.
15 middleName2-length
15 middleName2
```



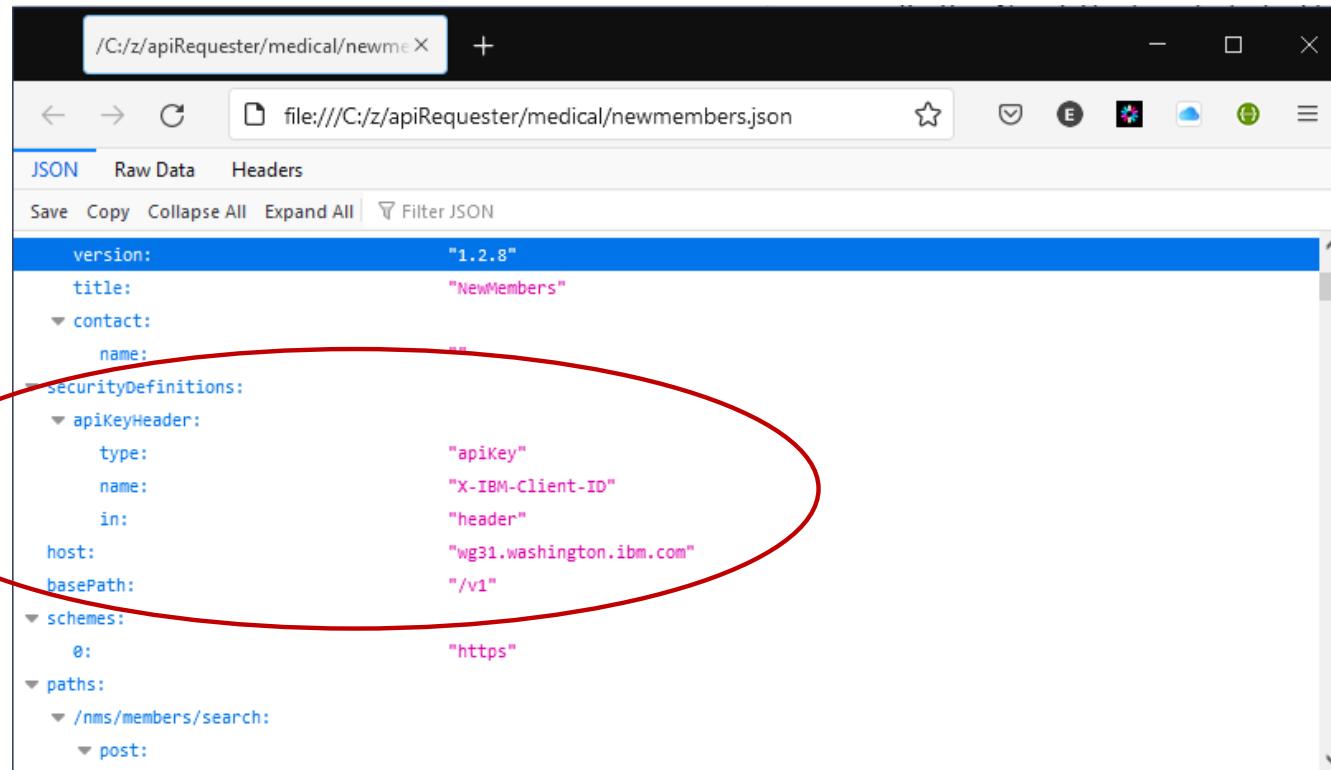
# Request and Response Message Layout





# For security, an API key(aka password) may be required

The details can be provided in the specification document as shown below or . . .



A screenshot of a JSON editor window titled "file:///C:/z/apiRequester/medical/newmembers.json". The window shows a JSON object with several fields. A red oval highlights the "securityDefinitions" field, which contains a "apiKeyHeader" object. This object has "type": "apiKey", "name": "X-IBM-Client-ID", and "in": "header". Below it is a "host" field with the value "wg31.washington.ibm.com" and a "basePath" field with the value "/v1". The "securityDefinitions" field also contains a "schemes" object with a single entry "0": "https". At the bottom, there is a "paths" object with a single entry for the path "/nms/members/search", which has a "post" method.

```
version: "1.2.8"
title: "NewMembers"
contact:
  name: ""
securityDefinitions:
  apiKeyHeader:
    type: "apiKey"
    name: "X-IBM-Client-ID"
    in: "header"
  host: "wg31.washington.ibm.com"
  basePath: "/v1"
  schemes:
    0: "https"
paths:
  /nms/members/search:
    post:
```

## Via a HTTP header

GET /something HTTP/1.1

**X-API-Key: abcdef12345**

## Or via a query parameter

GET /something?api\_key=abcdef12345

# The application provides the API key to the request header



Either way, the generated request copy book includes a ReqHeaders structure which can be used to provide values for header properties

request.cpy - Notepad

```
*      12 dob2-length          PIC S9999 COMP-5
* SYNC.
*      12 dob2                PIC X(255).
*
*
* ++++++
06 ReqHeaders.
09 X-IBM-Client-ID-length    PIC S9999 COMP-5 SYNC.
09 X-IBM-Client-ID           PIC X(255).
09 X-HZN-ClientName-length   PIC S9999 COMP-5 SYNC.
09 X-HZN-ClientName          PIC X(255).
09 X-HZN-ClientSubmitDateTime PIC S9(15) COMP-3.

09 X-HZN-ClientTransaction-num PIC S9(9) COMP-5 SYNC.

09 X-HZN-ClientTransactionId.
12 X-HZN-ClientTransact-length PIC S9999 COMP-5
SYNC.
12 X-HZN-ClientTransactionId2 PIC X(255).

09 X-HZN-ClientSessionId-num  PIC S9(9) COMP-5 SYNC.

09 X-HZN-ClientSessionId.
12 X-HZN-ClientSessionId-length PIC S9999 COMP-5
SYNC.
12 X-HZN-ClientSessionId2    PIC X(255).

09 X-HZN-UserRole-num        PIC S9(9) COMP-5 SYNC.

09 X-HZN-UserRole.
12 X-HZN-UserRole2-length    PIC S9999 COMP-5
SYNC.
12 X-HZN-UserRole2          PIC X(255).

09 X-HZN-UserAssociationI-num PIC S9(9) COMP-5 SYNC.

09 X-HZN-UserAssociationI-length PIC S9(9) COMP-5 SYNC.
```

mpz3

```
EDIT      USER1.ZCEE.SOURCE(GETAPIEN) - 01.01
Command ==> *-----
000081   *-----
000082   * Common code
000083   *-----
000084   * initialize working storage variables
000085   *      INITIALIZE GET-REQUEST.
000086   *      INITIALIZE GET-RESPONSE.
000087   MOVE "abcdef12345" to X-IBM-Client-ID
000088   MOVE 11 to X-IBM-Client-ID-length
000089
000090
000091
000092
000093
000094   MOVE employee of PARM-DATA TO employee IN GET-REQUEST.
000095   MOVE LENGTH of employee in GET-REQUEST to
000096   employee-length IN GET-REQUEST.
000097
000098
000099
000100
000101
000102
000103   * Use pointer and length to specify the location of
000104   * request and response segment.
000105   * This procedure is general and necessary.
000106   SET BAQ-REQUEST-PTR TO ADDRESS OF GET-REQUEST.
000107   MOVE LENGTH OF GET-REQUEST TO BAQ-REQUEST-LEN.
000108   SET BAQ-RESPONSE-PTR TO ADDRESS OF GET-RESPONSE.
000109   MOVE LENGTH OF GET-RESPONSE TO BAQ-RESPONSE-LEN.
```



## Or by including generation properties related to API keys

Use these generation properties to add API key information to the request message when not defined in specification document

**apiKeyMaxLength** - Specify the maximum length of the values set for API keys. The value of this parameter can be a positive integer in the range 1 - 32767. By default, **apiKeyMaxLength** is set to 255.

**apiKeyParmNameInHeader** - Specify the name of an API key that is sent as a request header. The value of this parameter can be set in a comma separated list of a combination of client ID and client secret. For example, you can set **apiKeyParmNameInHeader**=header-IBM-Client-ID, header-IBM-Client-secret when a client ID and a client secret are used to protect an API.

**apiKeyParmNameInQuery** - Specify the name of an API key that is sent in a query string. The value of this parameter can be set in a comma separated list of a combination of client ID and client secret. For example, you can set **apiKeyParmNameInQuery**=query-IBM-Client-ID, query-IBM-Client-secret when a client ID and a client secret are used to protect an API.

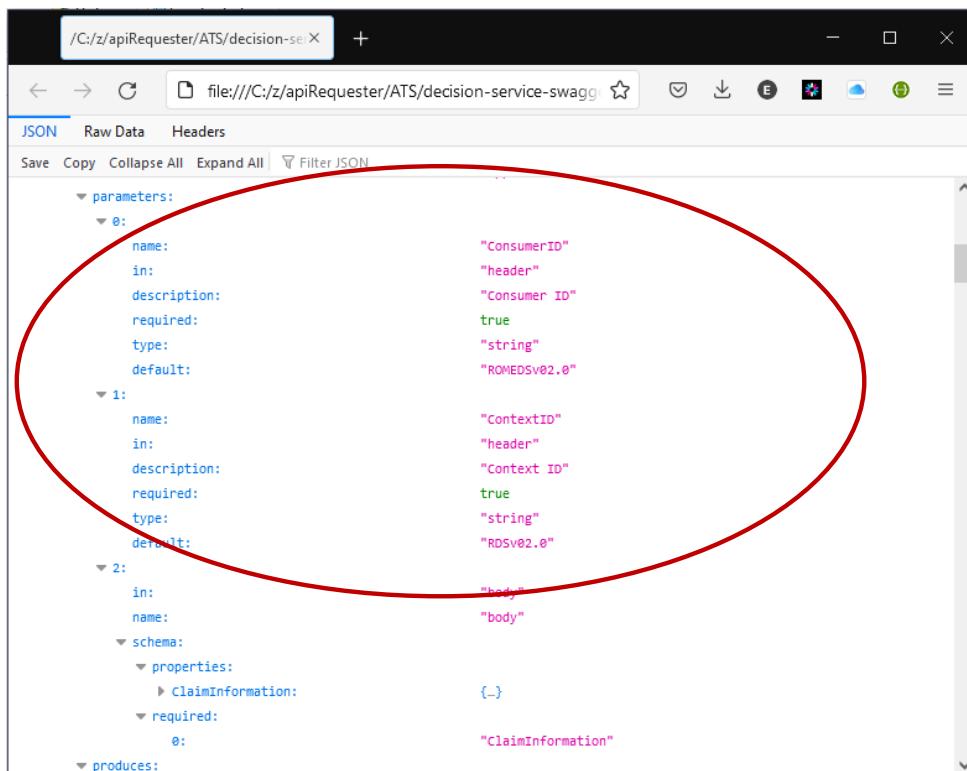
```
cscvinc.properties - Notepad
File Edit Format View Help
apiDescriptionFile=./cscvinc.json
dataStructuresLocation=./syslib
apiInfoFileLocation=./syslib
logFileDirectory=./logs
language=COBOL
connectionRef=cscvincAPI
requesterPrefix=ats
apiKeyMaxLength=40
apiKeyParmNameInHeader=X-IBM-Client-ID

Ln 8, Col 19 100% Unix (LF) UTF-8
```

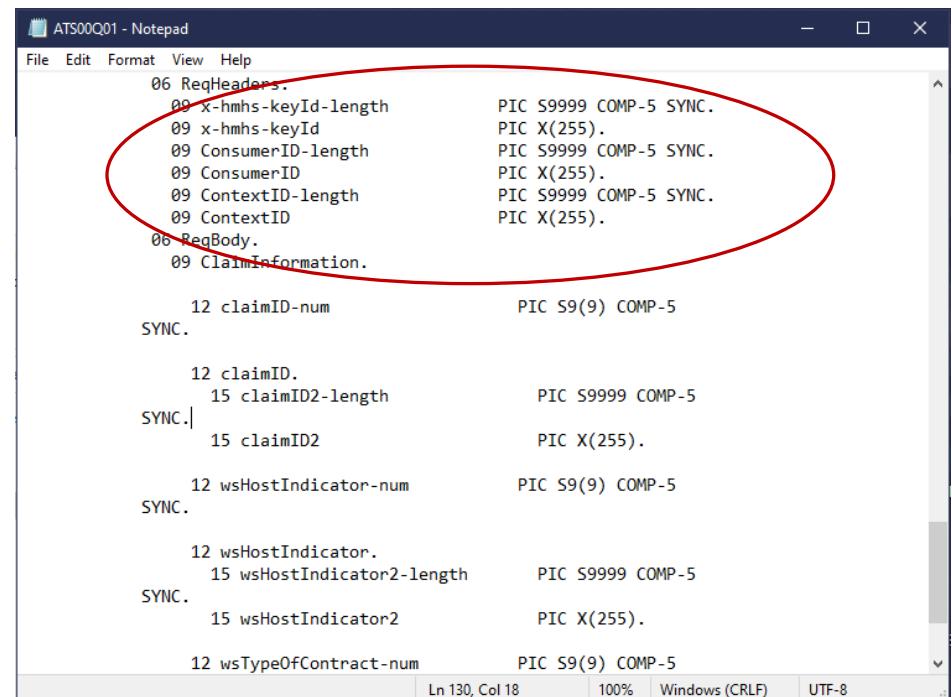


# And there may be other header properties can be required

The application can also set the values for other header properties that may be required by the API



```
/C:/apiRequester/ATS/decision-se X +  
file:///C:/apiRequester/ATS/decision-service-swagg  
JSON Raw Data Headers  
Save Copy Collapse All Expand All Filter JSON  
parameters:  
  0:  
    name: "ConsumerID"  
    in: "header"  
    description: "Consumer ID"  
    required: true  
    type: "string"  
    default: "ROMEDSv2.0"  
  1:  
    name: "ContextID"  
    in: "header"  
    description: "Context ID"  
    required: true  
    type: "string"  
    default: "RDSv02.0"  
  2:  
    in: "body"  
    name: "body"  
    schema:  
      properties:  
        ClaimInformation: (...)  
      required:  
        0: "ClaimInformation"  
    produces:
```

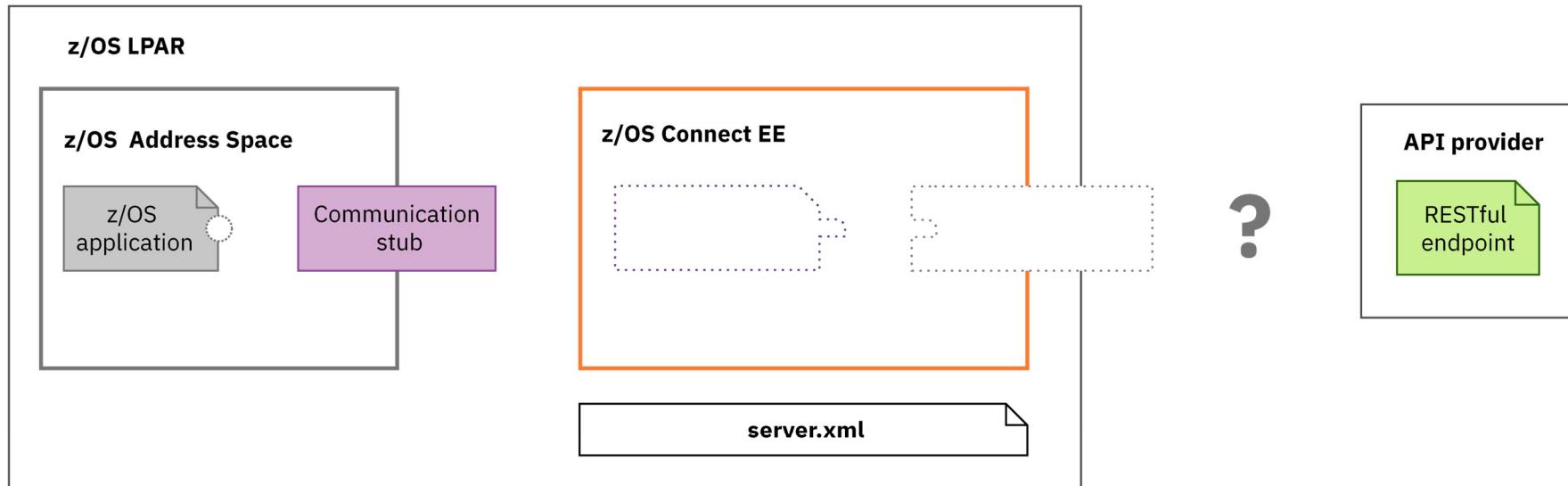


```
ATS00Q01 - Notepad  
File Edit Format View Help  
06 ReqHeaders.  
09 x-hmhs-keyId-length PIC S9999 COMP-5 SYNC.  
09 x-hmhs-keyId PIC X(255).  
09 ConsumerID-length PIC S9999 COMP-5 SYNC.  
09 ConsumerID PIC X(255).  
09 ContextID-length PIC S9999 COMP-5 SYNC.  
09 ContextID PIC X(255).  
06 ReqBody.  
09 ClaimInformation.  
  
12 claimID-num SYNC. PIC S9(9) COMP-5  
  
12 claimID.  
15 claimID2-length SYNC.] PIC S9999 COMP-5  
15 claimID2 PIC X(255).  
  
12 wsHostIndicator-num SYNC. PIC S9(9) COMP-5  
  
12 wsHostIndicator.  
15 wsHostIndicator2-length SYNC. PIC S9999 COMP-5  
15 wsHostIndicator2 PIC X(255).  
  
12 wsTypeOfContract-num PIC S9(9) COMP-5
```

# Involving an external API from a COBOL program (OpenAPI2)



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
*-----*
```



# Call the communication stub

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

The diagram illustrates the mapping between the `GETAPI` source code and the `CSC00101` copy book definitions.

**GETAPI Source Code:**

```
* 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.
```

**CSC00101 Copy Book Definitions:**

- `BAQ-APINAME`: PIC X(255), VALUE 'cscvincapi\_1.0.0'.
- `BAQ-APINAME-LEN`: PIC S9(9) COMP-5 SYNC, VALUE 16.
- `BAQ-APIPATH`: PIC X(255), VALUE 'S2fcsvincapi%2Femployee%7D'.
- `BAQ-APIPATH-LEN`: PIC S9(9) COMP-5 SYNC, VALUE 41.
- `BAQ-APIMETHOD`: PIC X(255), VALUE 'GET'.
- `BAQ-APIMETHOD-LEN`: PIC S9(9) COMP-5 SYNC, VALUE 3.

**Subsequent CSC00101 Definitions:**

- `employee`: A varying length array of characters or binary data.
  - `employee-length`: PIC S9999 COMP-5 SYNC.
  - `employee`: PIC X(6).
- `ReqPathParameters`:
  - `employee-length`: PIC S9999 COMP-5 SYNC.
  - `employee`: PIC X(6).
- `RespBody`:
  - `cscvincSelectServiceOp-num`: PIC S9(9) COMP-5 SYNC.
  - `cscvincSelectServiceOperatio`:
    - `Container1`: PIC S9(9) COMP-5 SYNC.
  - `RESPONSE-CONTAINER2-num`: PIC S9(9) COMP-5 SYNC.



# Response come back into working storage

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.

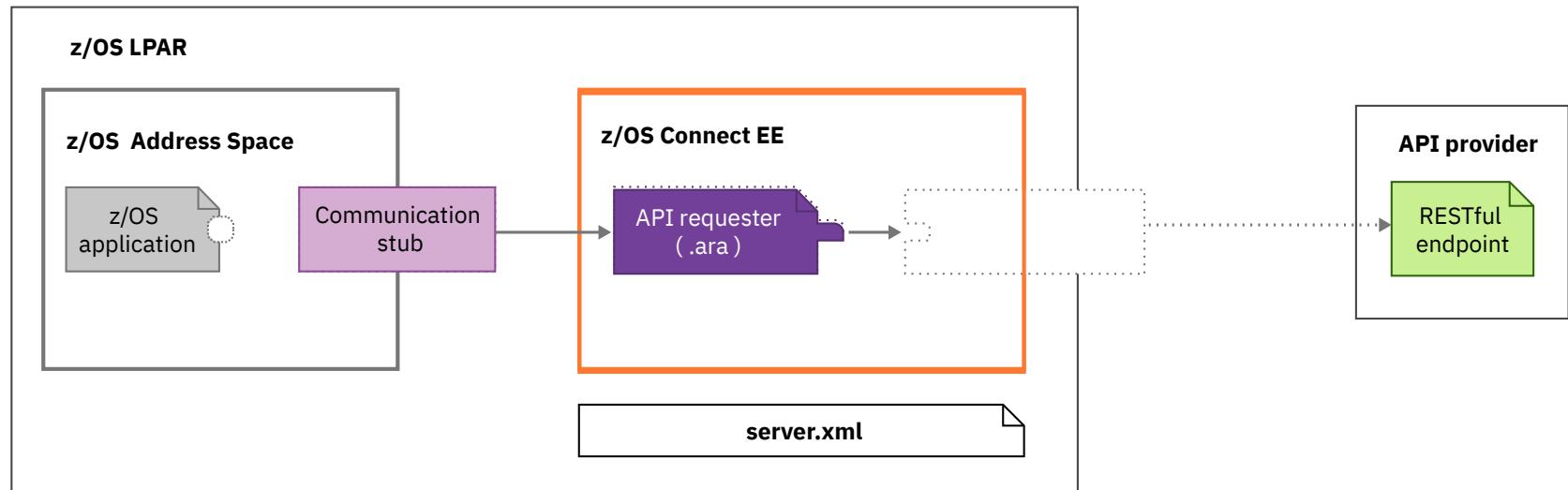
* When BAQ-RETURN-CODE is 'BAQ-SUCCESS', response is
* successfully returned and fields in RESPONSE copybook
* can be obtained. Display the translation result.
IF BAQ-SUCCESS THEN
    DISPLAY "NUMB: " Numb2 of API_RESPONSE
    DISPLAY "NAME: " name2 of API_RESPONSE
    DISPLAY "ADDRX: " addrx2 of API_RESPONSE
    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
```

```
mpz3
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 *****
```



# Deploy the API requester (.ara) archive

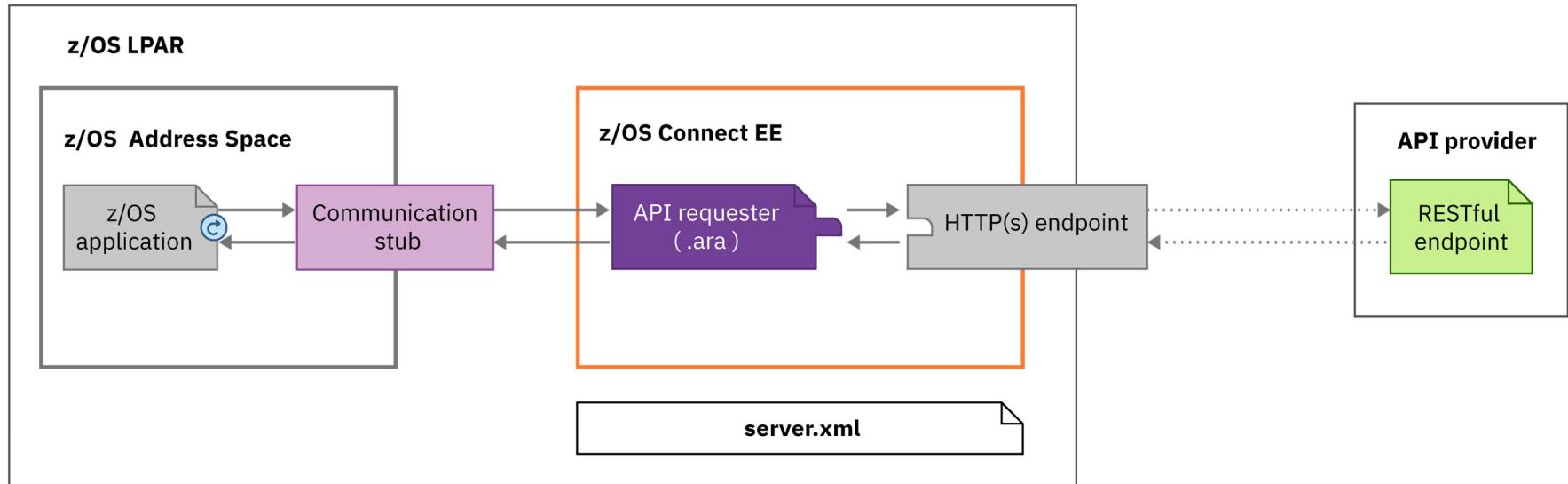


Deploy your API requester archive to the *apiRequesters* directory.



# Calling an external OpenAPI2 API

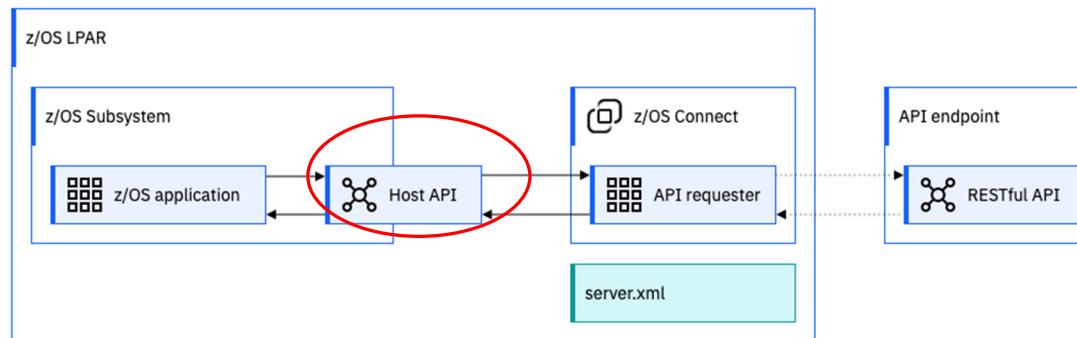
Done





# What is different with OpenAPI3 requester support?

In this case, the **z/OS Applications** calls z/OS Connect provided APIs to address issues encountered with the OpenAPI2 solution



- The Host API callable interface is a replacement for the IBM z/OS Connect OpenAPI 2.0 specification communication stub. It enables COBOL applications to call OpenAPI 3.0 specification RESTful APIs.
- The Host API has an extended set of callable verbs to support the richer set of features available in the OpenAPI 3.0 specification. For example, the Host API supports receiving **dynamically sized arrays** and/or **multiple mapped HTTP response codes** from the API endpoint. This capability enables a different response payload to be received by the COBOL program depending on whether the HTTP status code is 200 (OK), 201 (CREATED), or 404 (NOTFOUND). Each of the response payloads that are documented in the OpenAPI 3.0 definition operation are mapped to a different COBOL language structure.
- To support these new features, a **data area** is used. A data area is a location in the LINKAGE section of program storage that contains data *elements* that can be accessed sequentially by the COBOL program. Each dynamic array and each return response code has its own data area. To manage these data areas, and to manage IBM z/OS Connect connections, a set of Host API verbs is defined that are collectively known as the *Host API callable interface*.



# The Host API Verbs

## BAQINIT

- INITialize the storage required by the Host API and establish a connection to z/OS Connect

## BAQEXEC

- EXECute the call to the desired API taking request data as input and providing response data as output

## BAQGETN

- *[Optional]* - GET the Next data element from a named Data Area in the response

## BAQFREE

- *[Optional]* - FREE the storage currently in use by the Host API

## BAQTERM

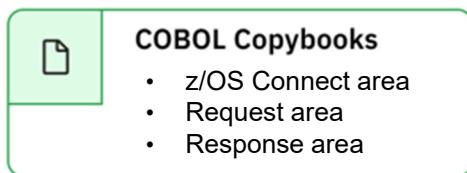
- TERMinate the connection to z/OS Connect and free all storage currently in use by the Host API



# COBOL copybooks

z/OS Connect provides two copybooks that are required to interface with the Host API and tooling to generate application specific copybooks

## 1. Product provided static structures



- **BAQHAREC** provides the 3 communication areas listed above
- **BAQHCONC** provides useful constants for the COBOL developer

## 2. Generated dynamic structures



- Output from running the API requester Gradle plug-in
- Maximum of 3 per operation
- Contents vary based on the OAS3 document and Gradle plug-in options specified



# BAQHAREC

## z/OS Connect area

### BAQ-ZCONNECT-AREA

Input & Output interface for all Host API verbs

- z/OS Connect parameters
  - e.g., URIMAP, z/OS Connect credentials
- Completion and Reason codes
- Service ID and Service Codes
- Return Message

```
01 BAQ-ZCONNECT-AREA.  
 03 BAQ-ZCON-AREA-EYE  
 03 BAQ-ZCON-AREA-LENGTH  
 03 BAQ-ZCON-AREA-VERSION  
 03 BAQ-ZCON-RESERVED-01  
 03 BAQ-ZCON-PARM-INIT  
 05 BAQ-ZCON-RESERVED-02  
 03 BAQ-ZCON-PARAMETERS  
 05 BAQ-ZCON-PARMS  
 07 BAQ-ZCON-PARM-NAME  
 07 BAQ-ZCON-PARM-ADDRESS  
 07 BAQ-ZCON-PARM-LENGTH  
 03 BAQ-ZCON-RETURN-CODES.  
 05 BAQ-ZCON-COMPLETION-CODE  
 88 BAQ-SUCCESS  
 88 BAQ-WARNING  
 88 BAQ-ERROR  
 88 BAQ-SEVERE  
 88 BAQ-CRITICAL  
 05 BAQ-ZCON-REASON-CODE  
 05 BAQ-ZCON-SERVICE-ID  
 05 BAQ-ZCON-SERVICE-CODE  
 05 BAQ-ZCON-RESERVED-03  
 03 BAQ-ZCON-RETURN-MESSAGE-LEN  
 03 BAQ-ZCON-RETURN-MESSAGE  
 PIC X(4) VALUE 'BAQZ'.  
 PIC 9(8) COMP-5 VALUE 4648.  
 PIC 9(8) COMP-5 VALUE 1.  
 PIC 9(8) COMP-5 VALUE 0.  
 VALUE LOW-VALUES.  
 PIC X(3584).  
 REDEFINES BAQ-ZCON-PARM-INIT.  
 OCCURS 64.  
 PIC X(48).  
 USAGE POINTER.  
 PIC 9(9) BINARY.  
  
 PIC 9(8) COMP-5 VALUE 0.  
 VALUE 0.  
 VALUE 4.  
 VALUE 8.  
 VALUE 12.  
 VALUE 16.  
 PIC 9(8) COMP-5 VALUE 0.  
 PIC X(1024).
```



# BAQHAREC

Request area

## BAQ-REQUEST-AREA

Input to the **BAQEXEC** for an API call

- Provide the Host API with the address and length of the request data
- Set any parameters required for calling the API
  - e.g., For OAuth or Token authentication with API endpoint

```
01 BAQ-REQUEST-AREA.  
 03 BAQ-REQ-AREA-EYE  
 03 BAQ-REQ-AREA-LENGTH  
 03 BAQ-REQ-AREA-VERSION  
 03 BAQ-REQ-RESERVED-01  
 03 BAQ-REQ-BASE-ADDRESS  
 03 BAQ>REQ-BASE-LENGTH  
 03 BAQ-REQ-PARM-INIT  
   05 BAQ-REQ-RESERVED-02  
 03 BAQ-REQ-PARAMETERS  
   05 BAQ-REQ-PARMS  
     07 BAQ-REQ-PARM-NAME  
     07 BAQ-REQ-PARM-ADDRESS  
     07 BAQ-REQ-PARM-LENGTH
```

```
PIC X(4) VALUE 'BAQR'.  
PIC 9(8) COMP-5 VALUE 3608.  
PIC 9(8) COMP-5 VALUE 1.  
PIC 9(8) COMP-5 VALUE 0.  
USAGE POINTER.  
PIC 9(8) BINARY.  
VALUE LOW-VALUES.  
PIC X(3584).  
REDEFINES BAQ-REQ-PARM-INIT.  
OCCURS 64.  
PIC X(48).  
USAGE POINTER.  
PIC 9(9) BINARY.
```



# BAQHAREC

## Response area

### BAQ-RESPONSE-AREA

#### Output from the BAQEXEC call

- Provides the address and length of the response BASE area
- The status code received from the API endpoint
- Any status message received from the API endpoint
  - Contains response payload when z/OS connect could not handle the API endpoint HTTP response code

```
• 01 BAQ-RESPONSE-AREA.  
  • 03 BAQ-RESP-AREA-EYE  
  • 03 BAQ-RESP-AREA-LENGTH  
  • 03 BAQ-RESP-AREA-VERSION  
  • 03 BAQ-RESP-RESERVED-01  
  • 03 BAQ-RESP-BASE-ADDRESS  
  • 03 BAQ-RESP-BASE-LENGTH  
  • 03 BAQ-RESP-RESERVED-02  
  • 03 BAQ-RESP-STATUS-CODE  
  • 03 BAQ-RESP-STATUS-MESSAGE-LEN  
  • 03 BAQ-RESP-STATUS-MESSAGE  
    PIC X(4) VALUE 'BAQP'.  
    PIC 9(8) COMP-5 VALUE 1060.  
    PIC 9(8) COMP-5 VALUE 1.  
    PIC 9(8) COMP-5 VALUE 0.  
    USAGE POINTER.  
    PIC 9(8) COMP-5 VALUE 0.  
    PIC X(1024).  
  
```



# BAQHCONC

Product provided constants for convenience

- Host API entry point names for dynamic calling
- Host API Request specific parameters
- Host API z/OS Connect parameters

```
Host API entry point names  
77 BAQ-INIT-NAME          PIC X(8) VALUE 'BAQINIT'.  
77 BAQ-EXEC-NAME          PIC X(8) VALUE 'BAQEXEC'.  
77 BAQ-GETN-NAME          PIC X(8) VALUE 'BAQGETN'.  
77 BAQ-FREE-NAME          PIC X(8) VALUE 'BAQFREE'.  
77 BAQ-TERM-NAME          PIC X(8) VALUE 'BAQTERM'.  
  
Host API Request parameter names  
77 BAQR-OAUTH-USERNAME    PIC X(22)  
                           VALUE 'BAQHAPI-oAuth-Username'.  
77 BAQR-OAUTH-PASSWORD    PIC X(22)  
                           VALUE 'BAQHAPI-oAuth-Password'.  
77 BAQR-OAUTH-SCOPE       PIC X(19)  
                           VALUE 'BAQHAPI-oAuth-Scope'.  
....  
  
Host API ZCON parameter names  
77 BAQZ-SERVER-URIMAP     PIC X(21)  
                           VALUE 'BAQHAPI-Server-URIMAP'.  
77 BAQZ-SERVER-USERNAME    PIC X(23)  
                           VALUE 'BAQHAPI-Server-Username'.  
77 BAQZ-SERVER-PASSWORD    PIC X(23)  
                           VALUE 'BAQHAPI-Server-Password'.  
....
```



# The API information (“I”) copybook

- Static structure required by the Host API [BAQEXEC](#) call
- Contains information about the API to be called
- **Must not** be changed!

```
01 BAQ-API-INFO-RBK02I01.  
 03 BAQ-API-INFO-EYE          PIC X(4)  
        VALUE 'BAQA'.  
 03 BAQ-API-INFO-LENGTH       PIC 9(9) COMP-5 SYNC  
        VALUE 1052.  
 03 BAQ-API-INFO-VERSION      PIC 9(9) COMP-5 SYNC  
        VALUE 1.  
 03 BAQ-API-INFO-RESERVED01   PIC 9(9) COMP-5 SYNC  
        VALUE 0.  
 03 BAQ-API-NAME             PIC X(255)  
        VALUE 'RedbookApi'.  
 03 BAQ-API-NAME-LEN         PIC 9(9) COMP-5 SYNC  
        VALUE 10.  
 03 BAQ-API-PATH              PIC X(255)  
        VALUE '%2Fredbooks'.  
 03 BAQ-API-PATH-LEN         PIC 9(9) COMP-5 SYNC  
        VALUE 11.  
 03 BAQ-API-METHOD            PIC X(255)  
        VALUE 'GET'.  
 03 BAQ-API-METHOD-LEN        PIC 9(9) COMP-5 SYNC  
        VALUE 3.  
 03 BAQ-API-OPERATION         PIC X(255)  
        VALUE 'getAllRedbooks'.  
 03 BAQ-API-OPERATION-LEN     PIC 9(9) COMP-5 SYNC  
        VALUE 14.
```



# The Request message (“Q”) copybook

- Working storage structure required by the Host API [BAQEXEC](#) call
- Contains request data to be sent to the API
- Must be initialized by the calling COBOL program
  - Avoid random data being sent in the request
  - e.g.,

```
01 BAQBASE-RBK02Q01.  
03 requestQueryParameters.
```

```
06 Xauthor-existence          PIC S9(9) COMP-5 SYNC.
```

```
06 Xauthor.  
  09 Xauthor2-length          PIC S9999 COMP-5 SYNC.  
  09 Xauthor2                PIC X(40).
```

```
INITIALIZE BAQBASE-RBK02Q01.
```



# The Response message (“P”) copybook

- Multiple dynamic structures which live in the **LINKAGE SECTION** of the COBOL program
- Each structure must be addressed before being accessed
  - **BAQ-RESP-BASE-ADDRESS**
  - **BAQGETN**
- Every **01** level structure beyond BAQBASE is considered a Data Area

```
01 BAQBASE-RBK02P01.  
  03 responseCode200-num          PIC S9(9) COMP-5 SYNC.  
  03 responseCode200-dataarea    PIC X(16).  
  
  03 responseCode404-existence  PIC S9(9) COMP-5 SYNC.  
  03 responseCode404-dataarea  PIC X(16).  
  
01 RBK02P01-responseCode200.  
  03 responseCode200.            PIC S9999 COMP-5 SYNC.  
    06 Xtitle-length           PIC X(80).  
    06 Xtitle.  
  
    06 authors-num             PIC S9(9) COMP-5 SYNC.  
    06 authors-dataarea        PIC X(16).  
  
    06 Xstatus-length          PIC S9999 COMP-5 SYNC.  
    06 Xstatus                PIC X(9).  
    06 formNumber              PIC X(12).  
  
.....  
  
01 RBK02P01-authors.  
  03 authors-length           PIC S9999 COMP-5 SYNC.  
  03 authors                PIC X(40).  
.....
```

# Host API Verbs – INIT & EXEC

1. Initialize the Host API and fail if the initialization was not successful
2. Ensure the request data is initialized and then call the API endpoint
1. If the call succeeded, address the response BAQBASE structure

```
Initialise the Host API  
CALL BAQ-INIT-NAME USING  
      BY REFERENCE BAQ-ZCONNECT-AREA.
```

```
Exit if initialisation fails  
IF NOT BAQ-SUCCESS THEN GO TO EXIT-PROGRAM.
```

```
Prepare the request data  
INITIALIZE BAQBASE-RBK02Q01.  
SET BAQ-REQ-BASE-ADDRESS TO ADDRESS OF BAQBASE-RBK02Q01.  
MOVE LENGTH OF BAQBASE-RBK02Q01 TO BAQ-REQ-BASE-LENGTH.
```

```
Call the API  
CALL BAQ-EXEC-NAME USING  
      BY REFERENCE BAQ-ZCONNECT-AREA  
      BY REFERENCE BAQ-API-INFO-RBK02I01  
      BY REFERENCE BAQ-REQUEST-AREA  
      BY REFERENCE BAQ-RESPONSE-AREA.
```

```
Address the response data  
SET ADDRESS OF BAQBASE-RBK02P01 TO BAQ-RESP-BASE-ADDRESS.
```

## Host API Verbs - GETN

4. Prepare the length and get the address of the next element from the Data Area
5. Use this address to access the Data Area element
6. Process the data in the Data Area element

```
Element length as input to Host API  
MOVE LENGTH OF RBK02P01-responseCode200 TO WS-ELEMENT-LENGTH.
```

```
Get the next element  
CALL BAQ-GETN-NAME USING  
    BY REFERENCE BAQ-ZCONNECT-AREA  
    BY REFERENCE responseCode200-dataarea  
    BY REFERENCE WS-ELEMENT  
    BY REFERENCE WS-ELEMENT-LENGTH.
```

```
Address the element  
SET ADDRESS OF RBK02P01-responseCode200 TO WS-ELEMENT.
```

```
Print the title of the Redbook  
STRING 'Title -> '  
      Xtitle OF RBK02P01-responseCode200  
      (1:Xtitle-length OF RBK02P01-responseCode200)  
      DELIMITED BY SIZE  
      INTO WS-TERMINAL-MSG.  
  
PERFORM WRITE-RESPONSE-MSG.
```

## Host API Verbs – **FREE & TERM**

7. Free the storage in use by the Host API if we have a long running transaction

```
 Optionally free the storage in use by the Host API  
 CALL BAQ-FREE-NAME USING  
       BY REFERENCE BAQ-ZCONNECT-AREA.
```

8. Disconnect from the z/OS Connect server and free all storage in use by the Host API

```
 Terminate the connection  
 CALL BAQ-TERM-NAME USING  
       BY REFERENCE BAQ-ZCONNECT-AREA.
```

# Host API Parameters

9. Specify z/OS Connect parameters for example when overriding the URIMAP or providing authentication credentials
  
10. Specify request parameters for example when obtaining a JWT for authenticating with the API endpoint

```
Specify a URIMAP to use  
MOVE BAQZ-SERVER-URIMAP TO BAQ-ZCON-PARM-NAME(1)  
SET BAQ-ZCON-PARM-ADDRESS(1) TO ADDRESS OF WS-URIMAP  
MOVE LENGTH OF WS-URIMAP to BAQ-ZCON-PARM-LENGTH(1).
```

Initialise the Host API

```
Authentication server credentials  
01 JWT-USER PIC X(10) VALUE 'myUsername'.  
01 JWT-PSWD PIC X(10) VALUE 'myPassword'.
```

```
Send JWT credentials to z/OS Connect  
MOVE BAQR-TOKEN-USERNAME TO BAQ-REQ-PARM-NAME OF BAQ-REQ-PARMS(1)  
SET BAQ-REQ-PARM-ADDRESS OF BAQ-REQ-PARMS(1) TO ADDRESS OF JWT-USER  
MOVE LENGTH OF JWT-USER TO BAQ-REQ-PARM-LENGTH OF BAQ-REQ-PARMS(1)
```

```
MOVE BAQR-TOKEN-PASSWORD TO BAQ-REQ-PARM-NAME OF BAQ-REQ-PARMS(2)  
SET BAQ-REQ-PARM-ADDRESS OF BAQ-REQ-PARMS(2) TO ADDRESS OF JWT-PSWD  
MOVE LENGTH OF JWT-PSWD TO BAQ-REQ-PARM-LENGTH OF BAQ-REQ-PARMS(2)
```

Call the API endpoint using BAQEXEC

# Host API Completion Codes

0000 - 1999

- z/OS Connect runtime messages

- Review documentation and z/OS Connect server logs for further details

```
BAQ-ZCON-COMPLETION-CODE = 8  
BAQ-ZCON-REASON-CODE = 1095  
BAQ-ZCON-RETURN-MESSAGE = BAQR1095E The user credentials required to  
request a token from the authorization  
server, were not supplied.
```

2000 - 2999

## Host API messages

- More details on the next slide

```
BAQ-ZCON-COMPLETION-CODE = 8  
BAQ-ZCON-REASON-CODE = 2008  
BAQ-ZCON-RETURN-MESSAGE = BAQH2008E: BAQINIT must be called at the  
start of the program.
```

3000 - 3999

- Host API running in CICS

- CICS specific messages

```
BAQ-ZCON-COMPLETION-CODE = 8  
BAQ-ZCON-REASON-CODE = 3008  
BAQ-ZCON-RETURN-MESSAGE = BAQH3008E: Socket error when using  
URIMAP(BAQHZCON)
```

# Host API Completion and Reason Codes

- 2000 - 2999
- **Warning** – The COBOL program can continue to use the Host API.
- **Error** – The COBOL program must take an action to continue. Review the message for required action.
- **Severe** – The COBOL program must call the [BAQTERM](#) verb immediately. Contact IBM support if the problem persists.

```
BAQ-ZCON-COMPLETION-CODE = 4  
BAQ-ZCON-REASON-CODE = 2007  
BAQ-ZCON-RETURN-MESSAGE = BAQH2007W: BAQINIT has already been called.
```

```
BAQ-ZCON-COMPLETION-CODE = 8  
BAQ-ZCON-REASON-CODE = 2008  
BAQ-ZCON-RETURN-MESSAGE = BAQH2008E: BAQINIT must be called at the start of the program.
```

```
BAQ-ZCON-COMPLETION-CODE = 12  
BAQ-ZCON-REASON-CODE = 2001  
BAQ-ZCON-RETURN-MESSAGE = BAQH2001S: The call to BAQINIT for the initialization of the Host API failed unexpectedly. Service ID=16843008. Service Code=1536950272.
```



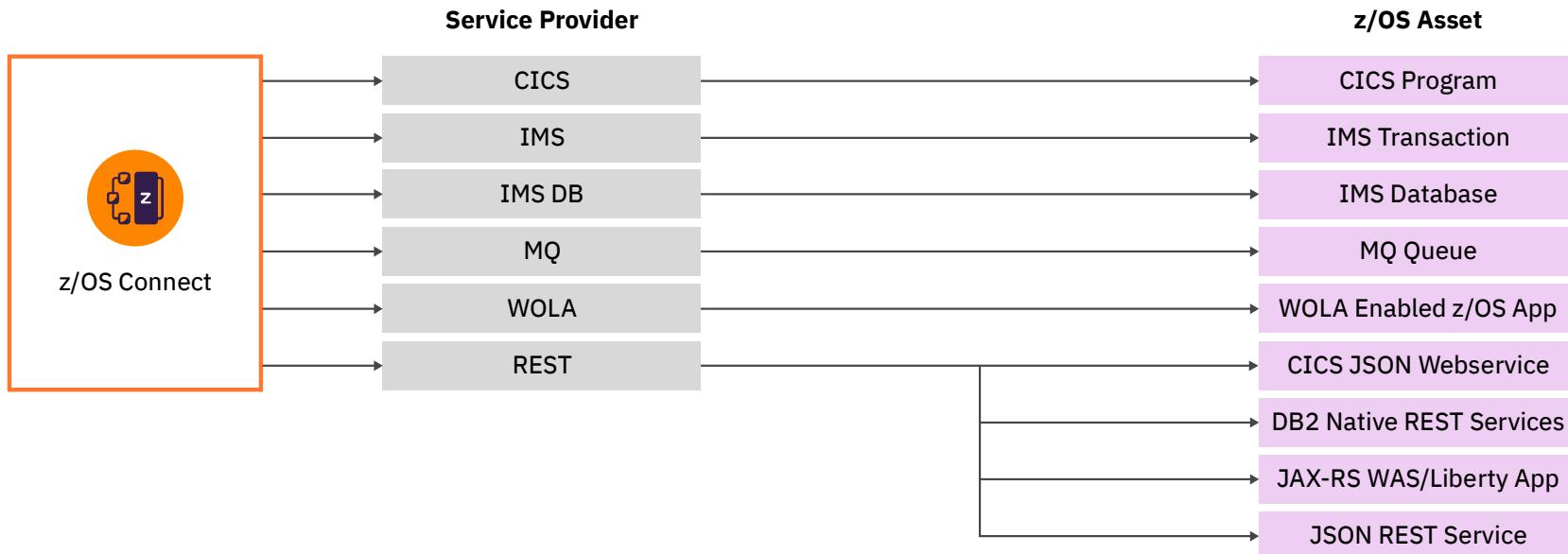
## /miscellaneousTopics

performance, high availability, Liberty



# What assets can z/OS Connect EE map to?

And which service provider could I use?

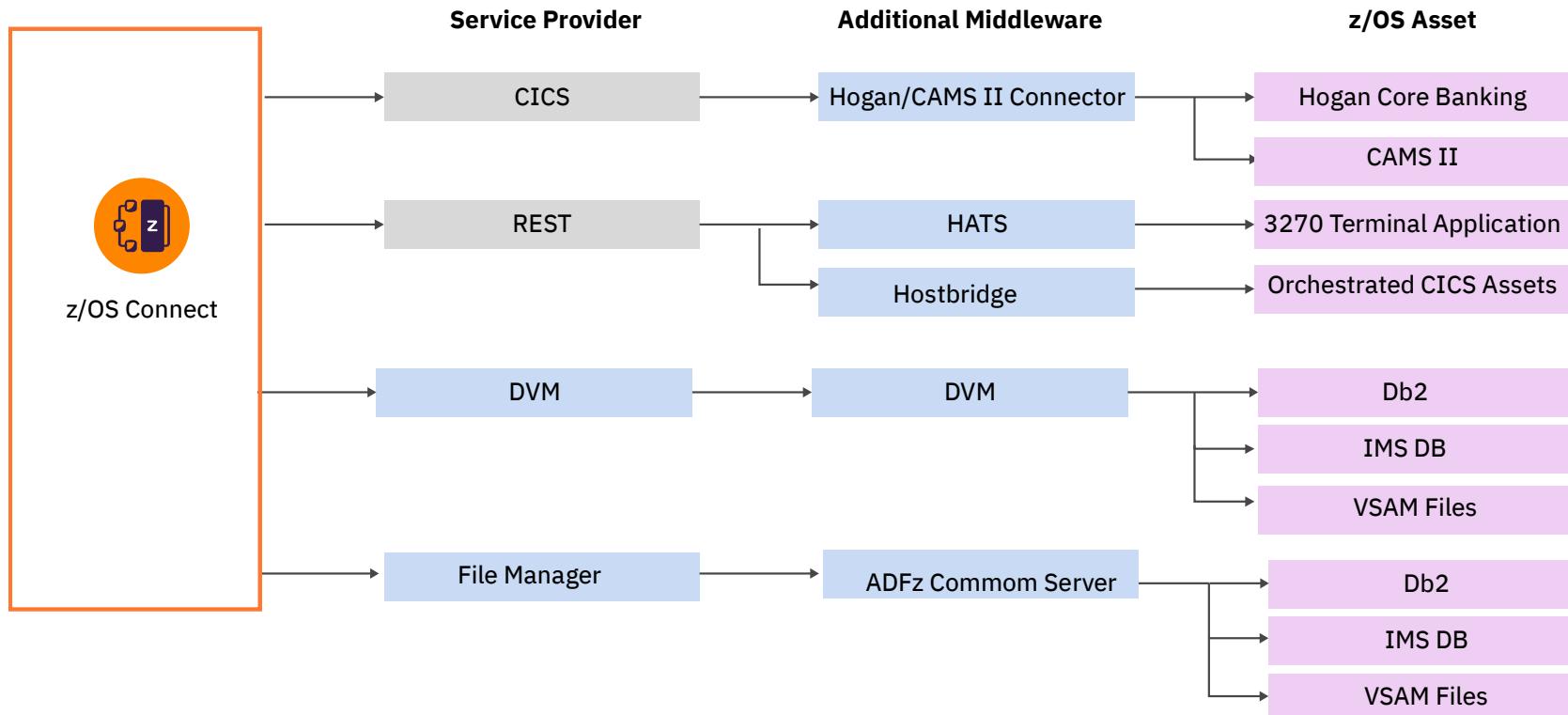


The core **service providers** included with z/OS Connect EE provide API access to a wide range of z/OS assets.



# Additional Middleware

Additional value from the ecosystem



z/OS Connect EE is **pluggable** and **extensible** allowing the use of additional middleware to expand the list of z/OS assets you can expose as APIs



# API Policies

- HTTP header properties can be used to select alternative for IMS (V3.0.4) , CICS (V3.0.10), Db2 (V3.0.36) or MQ (V3.0.39)
- Policies can be configured globally for every API in the server or for individual APIs (V3.0.11)

CICS attributes

- cicsCcsid
- cicsConnectionRef
- cicsTransId

IMS attributes

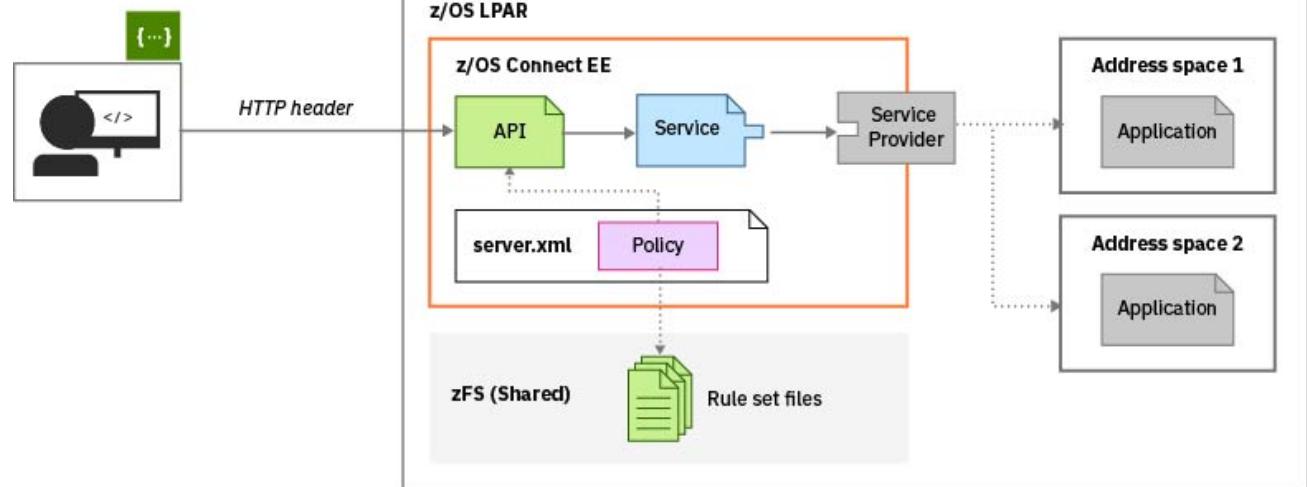
- imsConnectionRef
- imsInteractionRef
- imsInteractionTimeout
- imsLtermOverrideName
- imsTranCode
- imsTranExpiration

Db2 attributes

- db2ConnectionRef
- db2CollectionID

MQ attributes

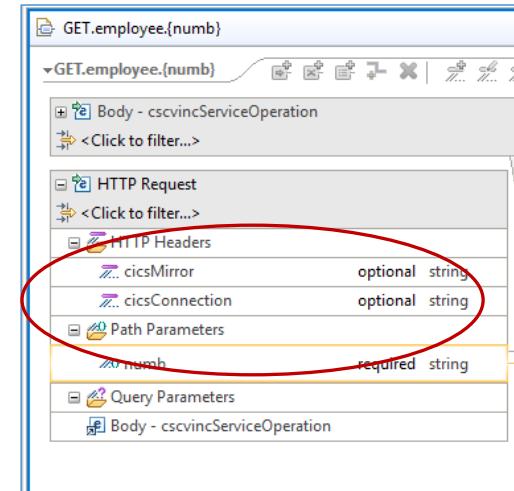
- mqConnectionFactory
- mqDestination
- mqReplyDestination





# A sample API Policies for CICS

```
<ruleset name="CICS rules">
  <rule name="csmi-rule">
    <conditions>
      <header name="cicsMirror" match="ANY_VALUE"/> *
    </conditions>
    <actions>
      <set property="cicsTransId" value="${cicsMirror}"/>
    </actions>
  </rule>
  <rule name="connection-rule">
    <conditions>
      <header name="cicsConnection" value="" match="ANY_VALUE"/>
    </conditions>
    <actions>
      <set property="cicsConnectionRef" value="${cicsConnection}">
    </actions>
  </rule>
</ruleset>
```



## Curl

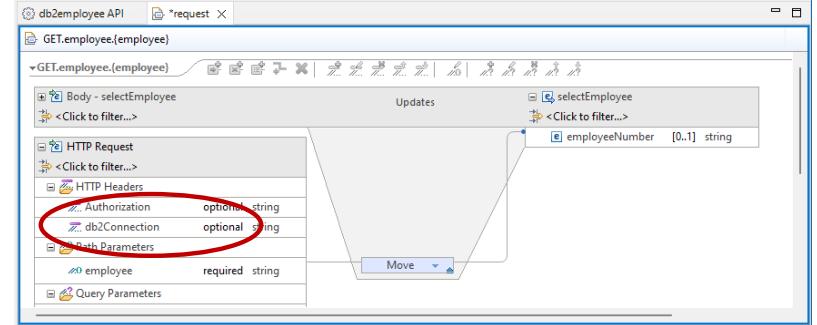
```
curl -X GET --header 'Accept: application/json' --header 'cicsMirror: MIJO' --header 'cicsConnection: cscvinc' 'https://m...
```

\*Transaction MIJO needs to be a clone of CSMI (e.g., invoke program DFHMIRS)

# A sample API Policies for Db2



```
<ruleset name="Db2 rules">
    <rule name="connection-rule">
        <conditions>
            <header name="db2Connection" value="" match="ANY_VALUE"/>
        </conditions>
        <actions>
            <set property="db2ConnectionRef" value="${db2Connection}"/>
        </actions>
    </rule>
</ruleset>
```



```
<zosconnect_zosConnectServiceRestClientConnection id="Db2Conn"
    sslCertsRef="db2SSLSettings" host="wg31.washington.ibm.com" port="2445" basicAuthRef="dsn2Auth" />
<zosconnect_zosConnectServiceRestClientBasicAuth id="dsn2Auth" applName="DSN2APPL"/>
<ssl id="db2SSLSettings" keyStoreRef="Db2KeyStore" trustStoreRef="Db2KeyStore"/>

<zosconnect_zosConnectServiceRestClientConnection id="fred"
    sslCertsRef="fredSSLSettings" host="wg31.washington.ibm.com" port="2445" />
<ssl id="fredSSLSettings" keyStoreRef="Db2KeyStore" trustStoreRef="Db2KeyStore" clientKeyAlias="FRED"/>

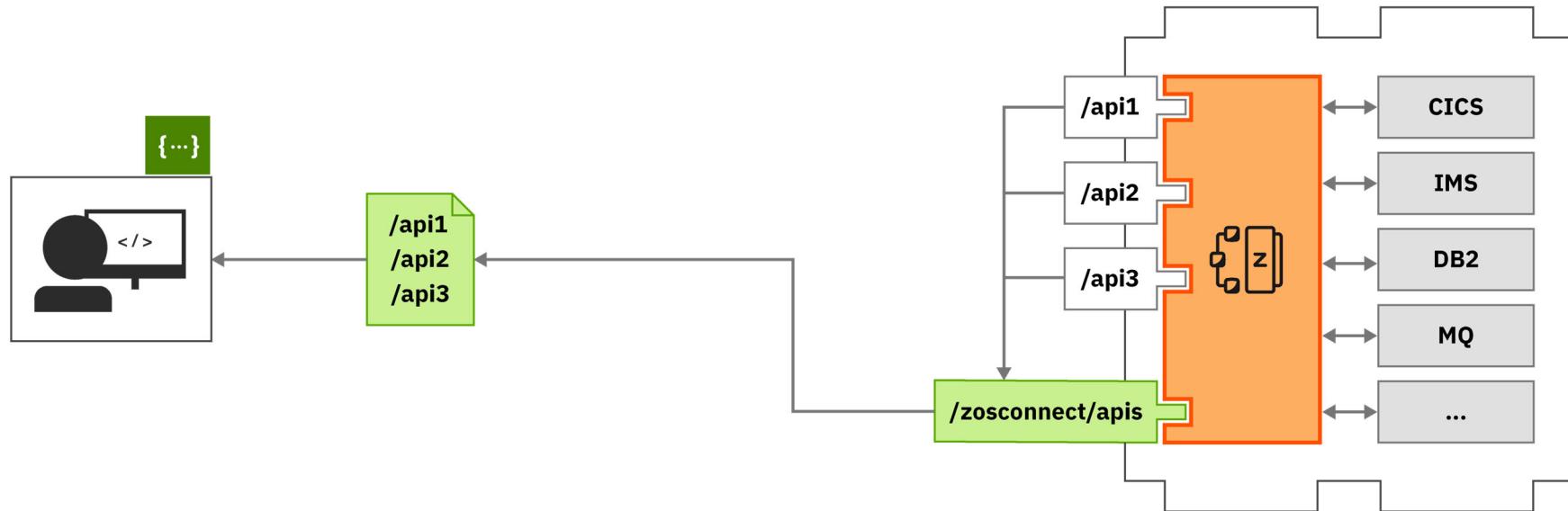
<zosconnect_zosConnectServiceRestClientConnection id="user1"
    sslCertsRef="user1SSLSettings" host="wg31.washington.ibm.com" port="2445" />
<ssl id="user1SSLSettings" keyStoreRef="Db2KeyStore" trustStoreRef="Db2KeyStore" clientKeyAlias="USER1"/>

<keyStore id="Db2KeyStore" location="safkeyring:///zCEE.Db2.KeyRing"
    password="password" type="JCERACFKS" fileBased="false" readOnly="true" />
```

Requires APAR PH53162



# API Documentation



APIs are discoverable via Swagger docs served from **z/OS Connect EE**.



## 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



# RESTful Administrative Interface for Services

The administration interface for services is available in paths under /zosConnect/services.

Most administration tasks are supported by the RESTful administration interface

Method	Administrative Task
GET	Get details of a service
	Get the status of a service
	Get the request schema of a service
	Get the response schema of a service
POST	Deploy a service*
PUT	Update a service
	Change the status of a service
DELETE	Delete a service

POST /zosConnect/services inquireSingle.sar  
PUT /zosConnect/services/{serviceName}?status=started|stopped  
PUT /zosConnect/services inquireSingle.sar  
GET /zosConnect/services  
GET /zosConnect/services/{serviceName}  
DELETE /zosConnect/services/{serviceName}

\*Useful for deploying service archive files, service archives generated by zconbt, e.g., HATS



# RESTful Administrative Interface for APIs

The administration interface for services is available in paths under /zosConnect/apis.

Most administration tasks are supported by the RESTful administration interface

Method	Administrative Task
GET	Get a list of APIs
	Get the details of an API
POST	Deploy an API
PUT	Update an API
	Change the status of an API
DELETE	Delete an API

```
POST  /zosConnect/apis CatalogManager.aar
PUT   /zosConnect/apis/{apiName}?status=started|stopped
PUT   /zosConnect/apis CatalogManager.aar
GET   /zosConnect/apis
GET   /zosConnect/apis/{apiName}
DELETE /zosConnect/apis/{apiName}
```



# RESTful Administrative Interface for API Requesters

The administration interface for services is available in paths under /zosConnect/apisRequesters. Most administration tasks are supported by the RESTful administration interface

Method	Administrative Task
GET	Get a list of API Requesters
	Get the details of an API Requester
POST	Deploy an API Requester
PUT	Update an API Requester
	Change the status of an API Requester
DELETE	Delete an API Requester

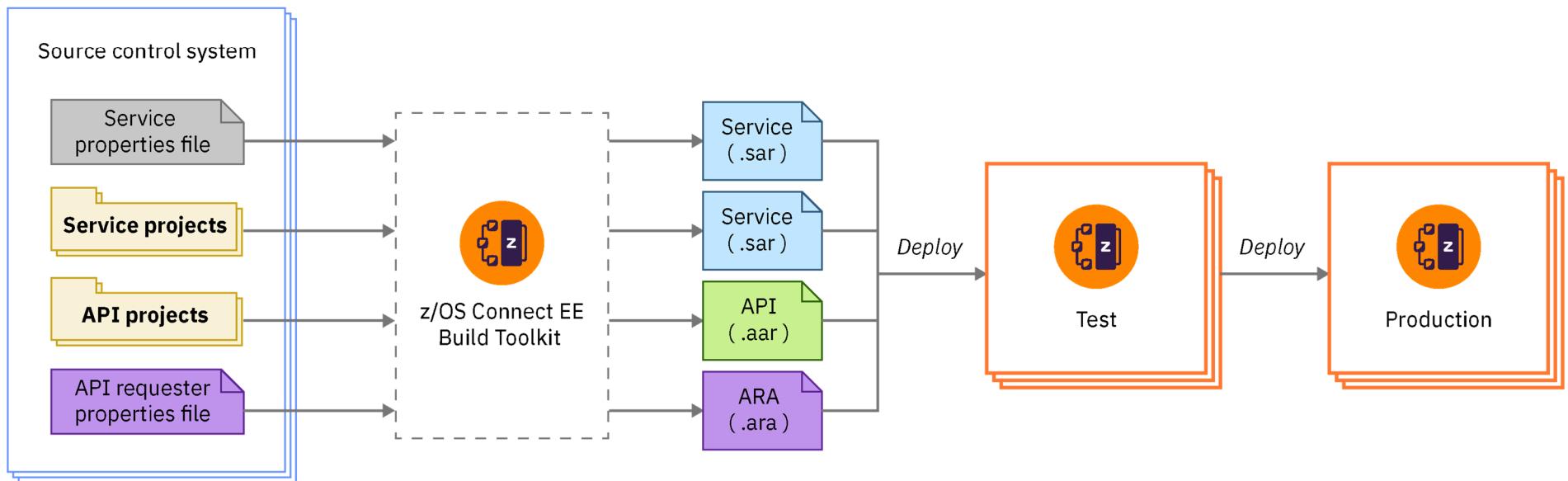
```
GET  /zosConnect/apiRequesters cscvinc.aar
PUT  /zosConnect/apiRequesters/{apiRequesterName}?status=started|stopped
PUT  /zosConnect/apiRequesters cscvinc.aar
GET  /zosConnect/apiRequesters
GET  /zosConnect/apiRequesters/{apRequesterName}
DELETE /zosConnect/apiRequesters
```



# DevOps using z/OS Connect EE

Automate the development and deployment of services, APIs, and API requesters for continuous integration and delivery.

- The build toolkit supports the generation of service archives and API archives from projects created in the z/OS Connect EE API toolkit
- The build toolkit also supports the use of properties files to generate API requester archives
- Run the build toolkit from a build script to generate these archive files
- Deploy them to z/OS Connect servers by copying them to their deployment directories or by using the REST Admin API



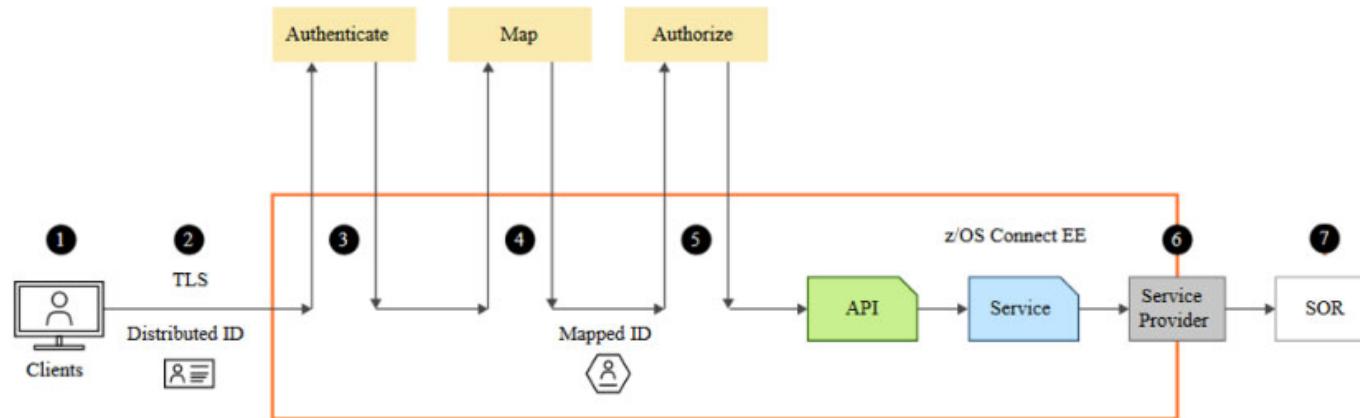


## /security

How is security implement?



# Typical z/OS Connect EE API Provider security flow



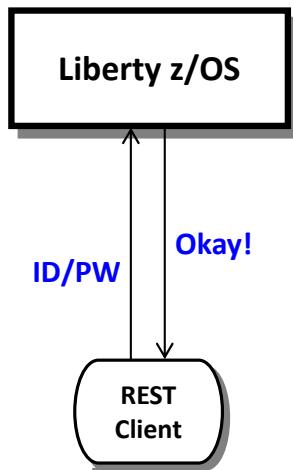
1. The credentials provided by the client
2. Secure the connection to the z/OS Connect EE server
3. Authenticate the client. This can be within the z/OS Connect EE server or by requesting verification from a third-party server
4. Map the authenticated identity to a user ID in the user registry
5. Authorize 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



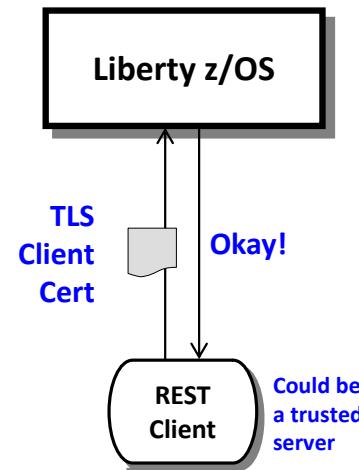
Server prompts for ID/PW

Client supplies ID/PW or  
ID/Passticket

Server checks registry:

- Basic (server.xml)
- LDAP
- SAF

## Client Certificate



Server prompts for cert.

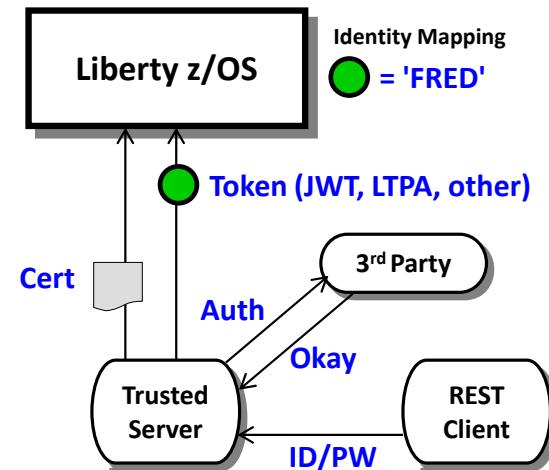
Client supplies certificate

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**

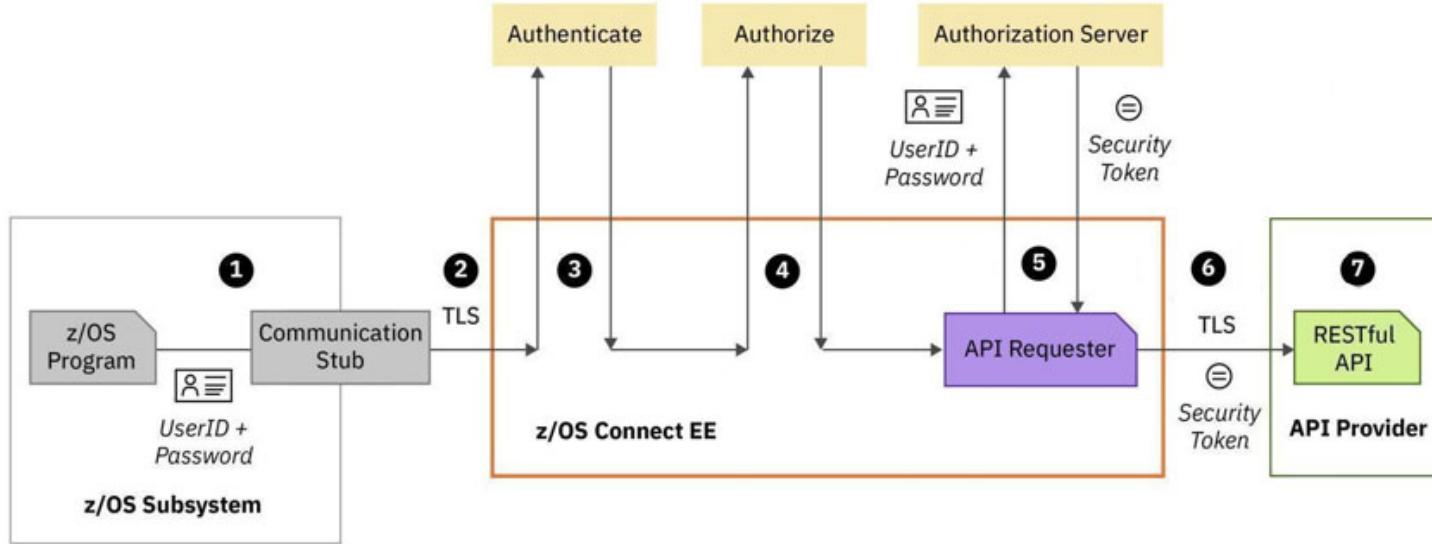
This screenshot shows the UPS Sign Up page at <https://wwwapps.ups.com/doapp/signup>. The page features the UPS logo and a "Sign Up" button. It includes links for existing users to log in and search or track packages. Below the main heading, it says "Use one of these sites." followed by social media integration buttons for Google, Facebook, Amazon, Apple, and Twitter. It also provides fields for entering personal information: Name\*, Email\*, User ID\*, Password\*, and Phone. A note indicates that asterisks denote required fields.

**Right Screenshot: myNCDMV Log In**

This screenshot shows the myNCDMV Log In page at <https://payments.ncdot.gov/auth>. The page has a "Log In" and "Sign Up" tab. It features a background image of autumn foliage. The log in form includes fields for Email Address and Password, a "Remember Me" checkbox, and "Log In" and "Forgot Password" buttons. Below the form are three social media integration buttons: "Continue with Apple", "Continue with Facebook", and "Continue with Google". A "Continue as Guest" link is also present. A notice for public computer users and a "powered by payit" footer are visible at the bottom.



# Typical z/OS Connect EE API Requester security flow



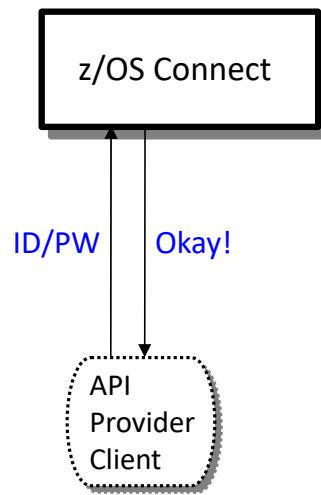
1. A user ID and password can be used for basic authentication by the z/OS Connect EE server
2. Connection between the CICS, IMS, or z/OS application and the z/OS Connect EE server can use TLS
3. Authenticate the CICS, IMS, or z/OS application.
4. Authorize the authenticated user ID to connect to z/OS 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

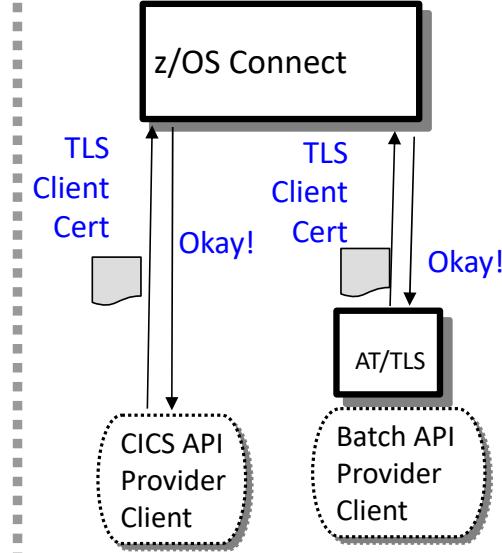
Two options for providing credentials for authentication

## Basic Authentication



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

## Client Certificate



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

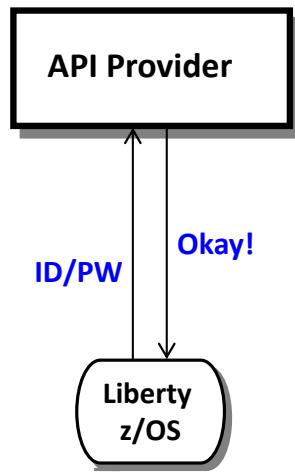
**CICS or AT/TLS supplies a  
client certificate**



# API Requester - API Provider Authentication

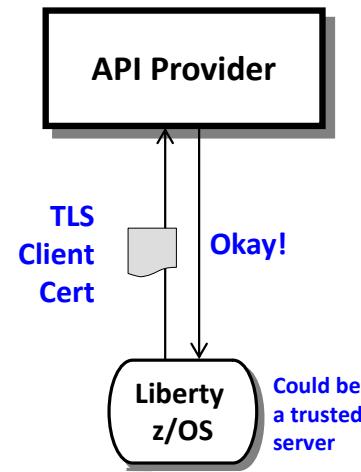
Several different ways this can be accomplished:

## Basic Authentication



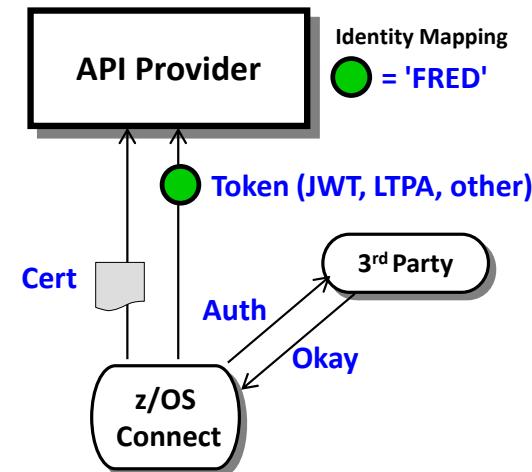
zCEE supplies ID/PW or  
ID/Passticket

## 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
MB  C
Connected to remote server/host wg31 using lu/pool TCP00133 and port 23
```

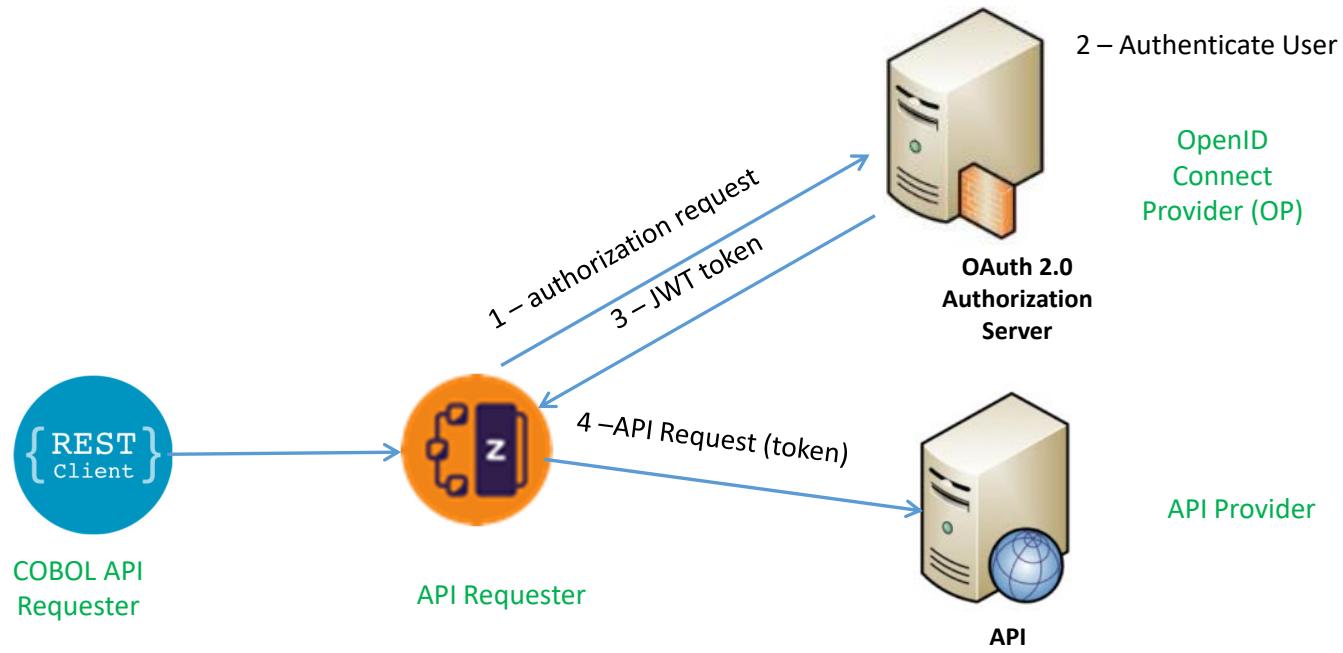
  

```
CICS RELEASE = 0710
File Edit Settings View Communication Actions Window Help
OVERTYPE TO MODIFY
CEDA ALTER UriMap( BAQURIMP )
  UriMap      : BAQURIMP
  Group       : SYSGRP
  Description ==> URIMAP for z/OS Connect EE server
  Status      ==> Enabled   Enabled | Disabled
  Usage       ==> Client    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
MB  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.**

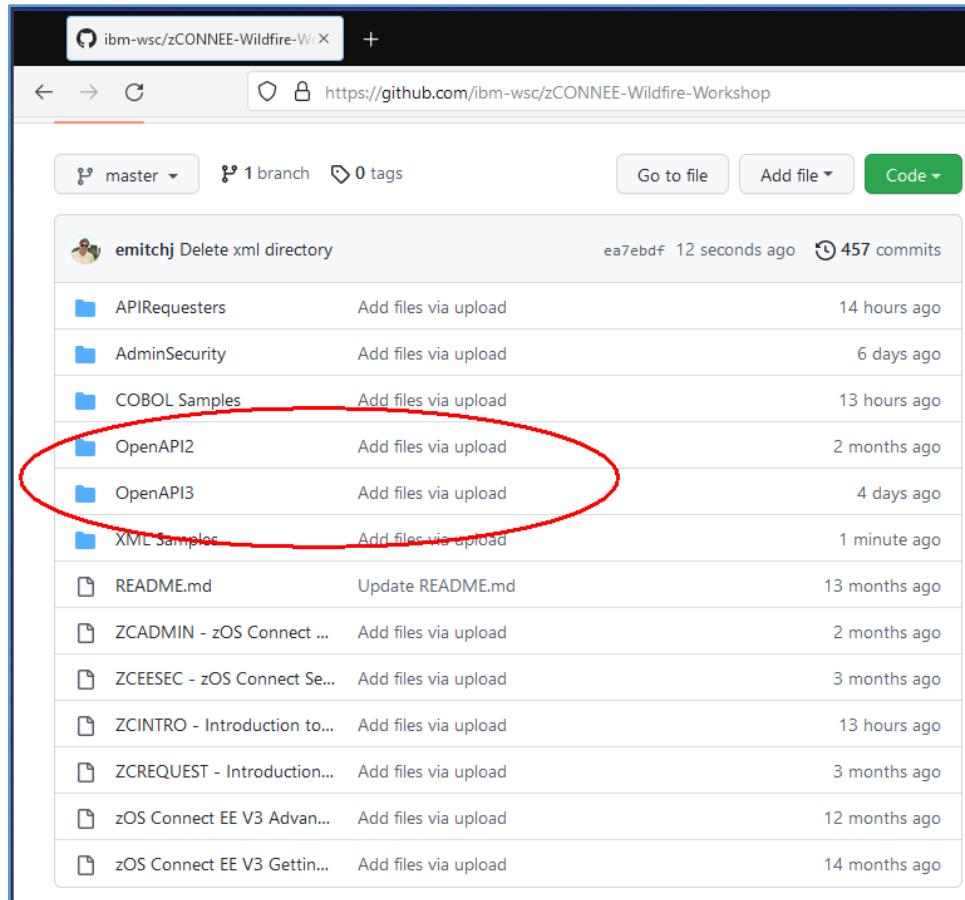
OAuth 2.0 specification entity	password	client_credentials	Where Set
Client ID	required	Required	server.xml or by application
Client Secret	optional	Required	server.xml or by application
Username	required	N/A	by application
Password	required	N/A	by application

# 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*

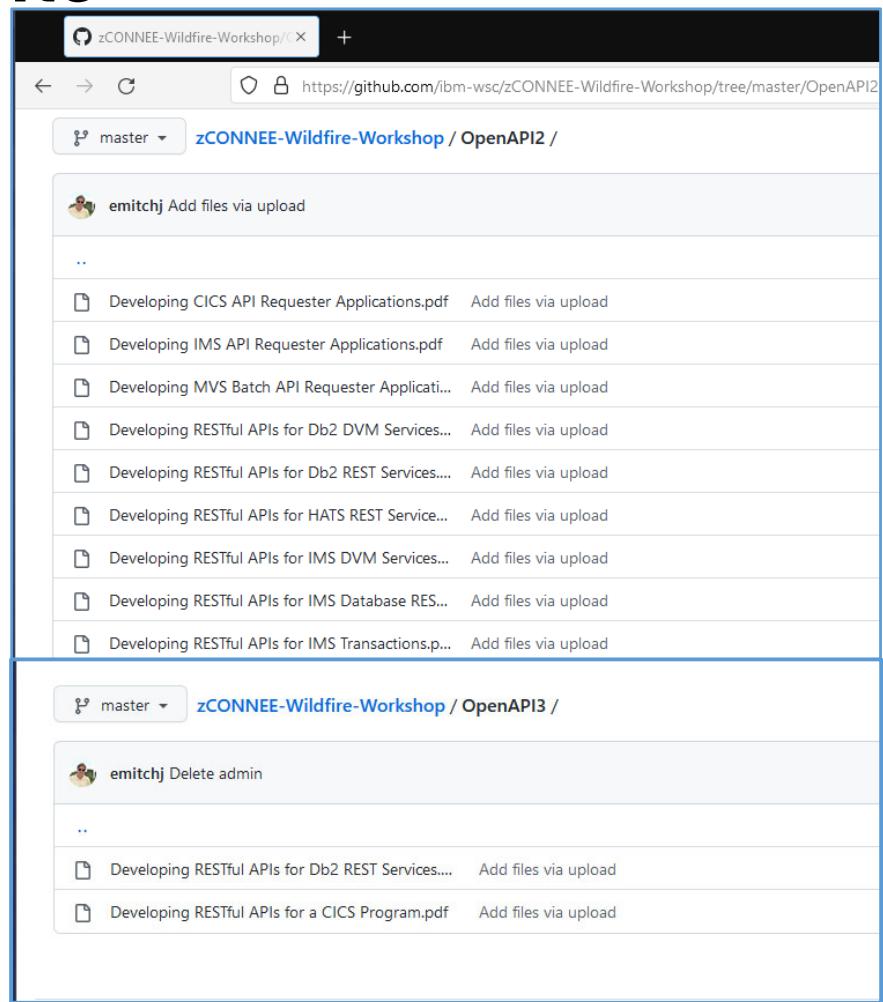
# z/OS Connect Wildfire Github Site



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?**

Not at all  
likely

0

1

2

3

4

5

6

7

8

9

10

Extremely  
likely

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**

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

© 2017, 2023 IBM Corporation  
Slide 187