



ZCINTRO - IBM z/OS Connect

An introduction and overview

Mitch Johnson

mitchj@us.ibm.com

Washington System Center



IBM Z
Wildfire Team –
Washington System Center

Agenda

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

Washington System Center

mitchj@us.ibm.com



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 Open API 3

Designer and z/OS Native server
Experiences and Observations

Mitch Johnson
mitchj@us.ibm.com
Washington Systems Center

mitchj@us.ibm.com



© 2022 IBM Corporation
Slide 1

mitchj@us.ibm.com

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

© 2018, 2022 IBM Corporation

z/OS Connect Wildfire Github Site

<https://ibm.biz/zCEEWorkshopMaterial>

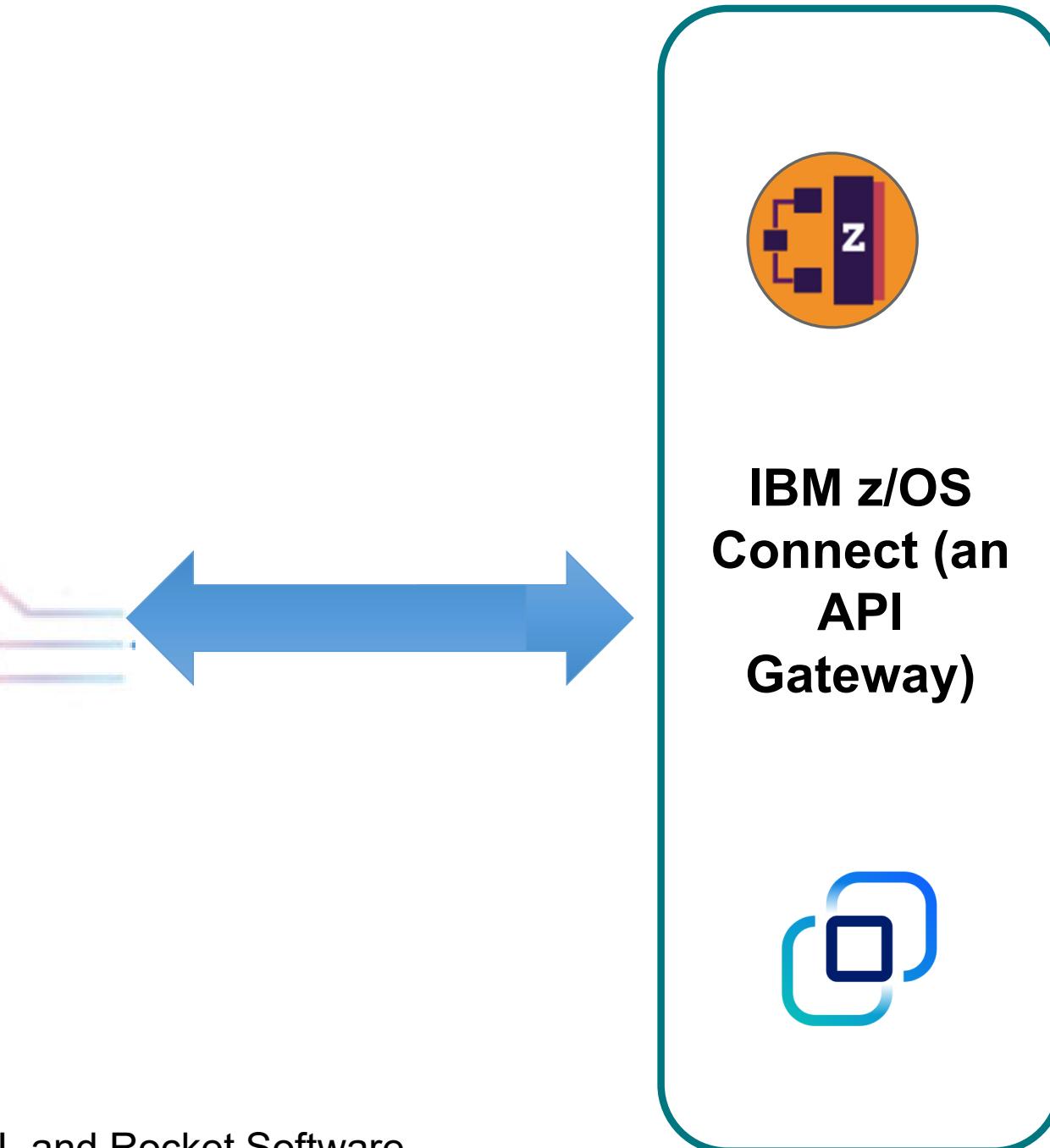


A screenshot of a GitHub repository page. The repository name is 'ibm-wsc/zCONNEE-Wildfire-Workshop'. The master branch has 1 branch and 0 tags. The commit history shows 457 commits from user 'emitchj'. A red oval highlights the 'OpenAPI2' and 'OpenAPI3' directories. Other visible files include 'README.md', 'ZCADMIN - zOS Connect ...', 'ZCEESEC - zOS Connect Se...', 'ZCINTRO - Introduction to...', 'ZCREQUEST - Introduction...', 'zOS Connect EE V3 Advan...', and 'zOS Connect EE V3 Gettin...'. The commits are dated from 12 seconds ago to 14 months ago.

A screenshot of a GitHub repository page for the 'OpenAPI2' branch. The repository name is 'zCONNEE-Wildfire-Workshop / OpenAPI2 /'. It shows 1 commit from 'emitchj' titled 'Add files via upload'. Below it is a list of 10 PDF files: '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.pdf', 'Developing RESTful APIs for Db2 REST Services.pdf', 'Developing RESTful APIs for HATS REST Service.pdf', 'Developing RESTful APIs for IMS DVM Services.pdf', 'Developing RESTful APIs for IMS Database RES.pdf', and 'Developing RESTful APIs for IMS Transactions.pdf'. The second screenshot shows the 'OpenAPI3' branch, which has 1 commit from 'emitchj' titled 'Delete admin'.

- Contact your IBM representative to request access to these exercises

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



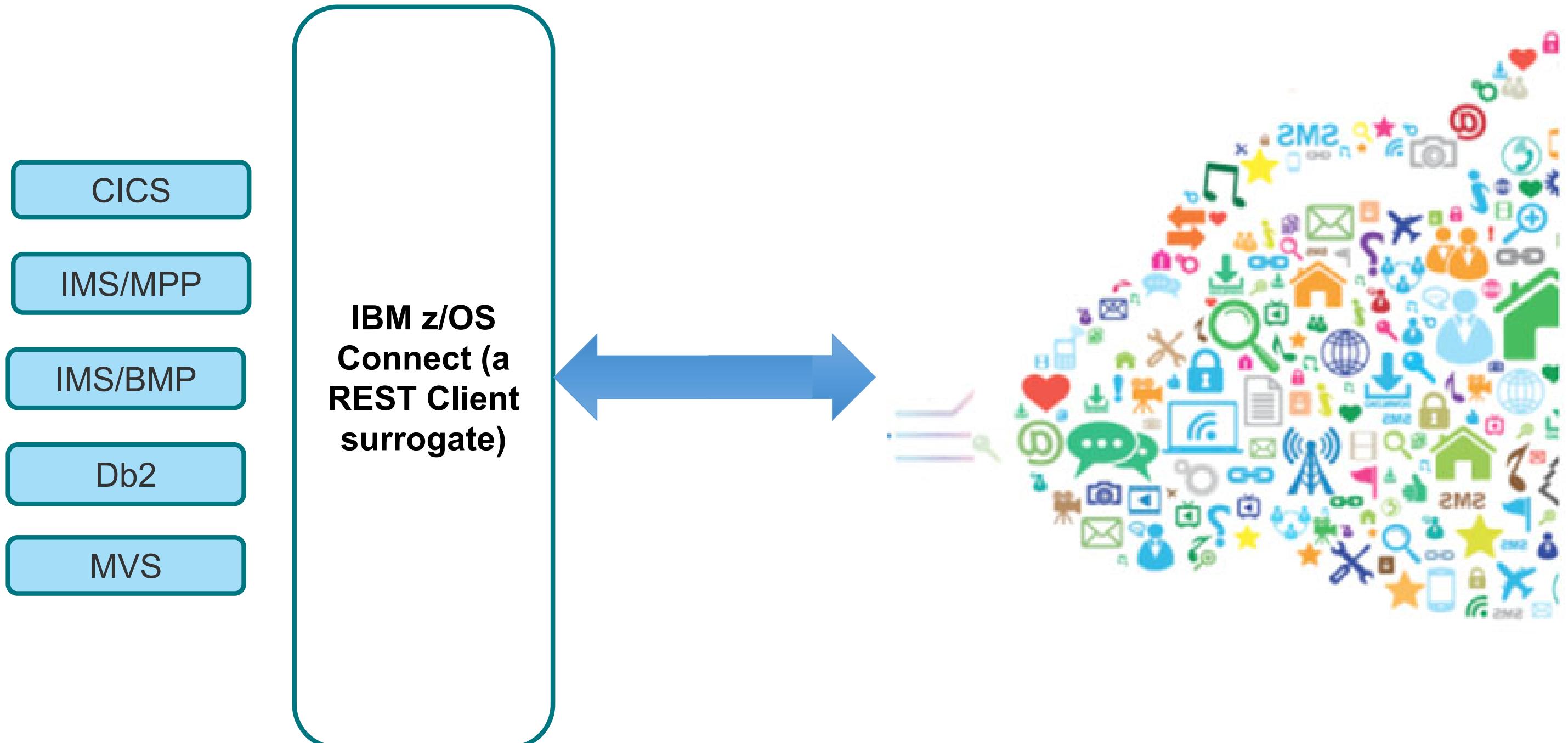
CICS
IMS/TM
IMS/DB
Db2
MQ
IBM File Manager ⁺
HATS(3270)
IBM DVM ⁺
MVS
WAS
Custom*

+ 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

/but_first, what_is_REST?

What makes an API “RESTful”?



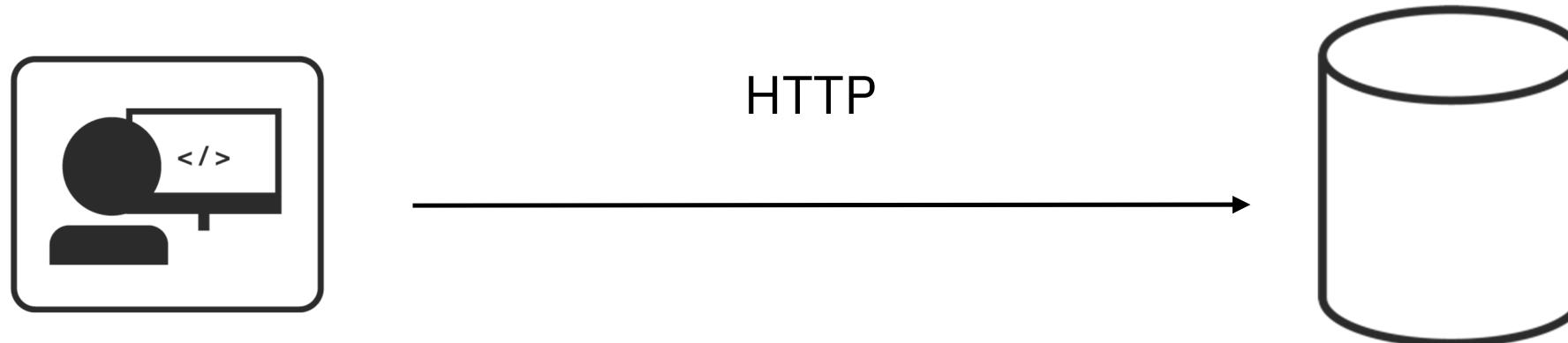
REST is architectural programming style

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?

Ubiquitous Foundation

It's based on HTTP, which operates on TCP/IP, which is a ubiquitous networking topology.

Relatively Lightweight

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.

Relatively Easy Development

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.

Increasingly Common

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.

Stateless

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.



Results: Client code is unaware of the z/OS infrastructure

ZeeCICSGet.java

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

Problems @ Javadoc Declaration Console Coverage

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

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

USERID: CICSUSER

CEIBRESP: 0

CEIBRESP2: 0

name: DR E. GRIFFITHS

employeeNumber: 222222

amount: \$0022.00

address: FRANKFURT, GERMANY

phoneNumber: 20034151

date: 26 11 81

Response Message: {"cscvincSelectServiceOperationResponse": {"cscvincContainer": {"response": {"CEIBRESP2": 0, "USERID": "CICSUSER",

ZeeMqGet.java

```
// Invoke the REST API using a GET method
URL url = new URL("https://wg31.washington.ibm.com:9453/mqapi/");
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", new String(bytesEncoded));
try {
    if (conn.getResponseCode() != 200) {
        throw new RuntimeException("Failed : HTTP error code : " + conn.getResponseCode());
    }
}
```

Problems @ Javadoc Declaration Console Coverage

<terminated> ZeeMqGet [Java Application] C:\Program Files\IBM\Java80\jre\bin\javaw.exe (May 7, 2021, 8:53:07 AM)

URL: https://wg31.washington.ibm.com:9453/mqapi/

NAME: TINA J YOUNG

NUMB: 001781

ADDRX: SINDELFINGEN, GERMANY

PHONE: 70319990

DATEX: 21 06 77

AMOUNT: \$0009.99

MQ

ZeeDb2Get.java

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

Problems @ Javadoc Declaration Console Coverage

<terminated> ZeeDb2Get [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: 2078

Db2

ZeeIMSGet.java

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

Problems @ Javadoc Declaration Console Coverage

<terminated> ZeeIMSGet [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

IMS

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

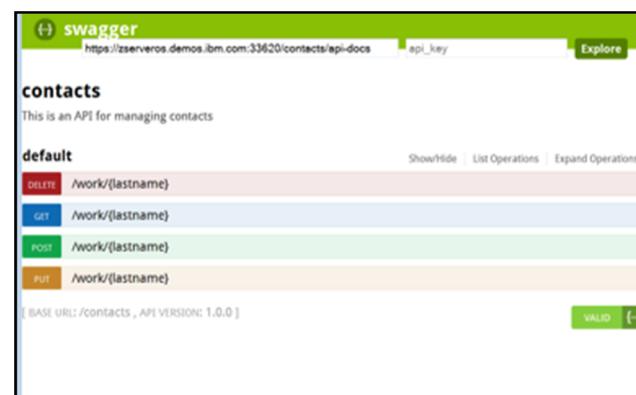


There are a number of tools available to aid consumption:

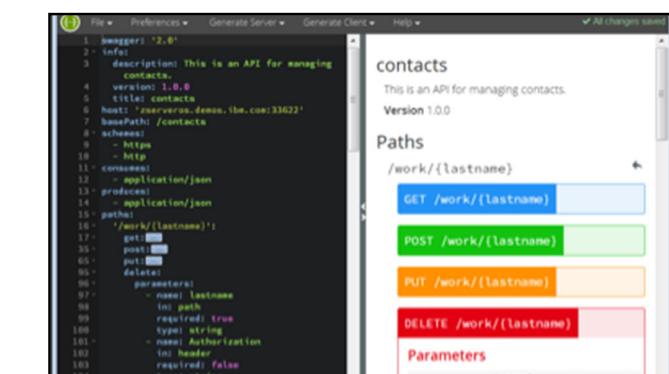
Code Generation* - create stub code to consume APIs from various languages



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**



An OPENAPI 2 versus an OPENAPI 3 specification document

JSON
format

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

```
cscvinc.yaml - Notepad
File Edit Format View Help
openapi: 3.0.1
info:
  title: cscvinc
  description: ""
  version: 1.0.0
servers:
- url: /cscvinc
x-ibm-zcon-roles-allowed:
- Manager
paths:
  /employee/{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

```

YAML
Format*



Why /zos_connect?

Truly RESTful APIs to and from your mainframe.

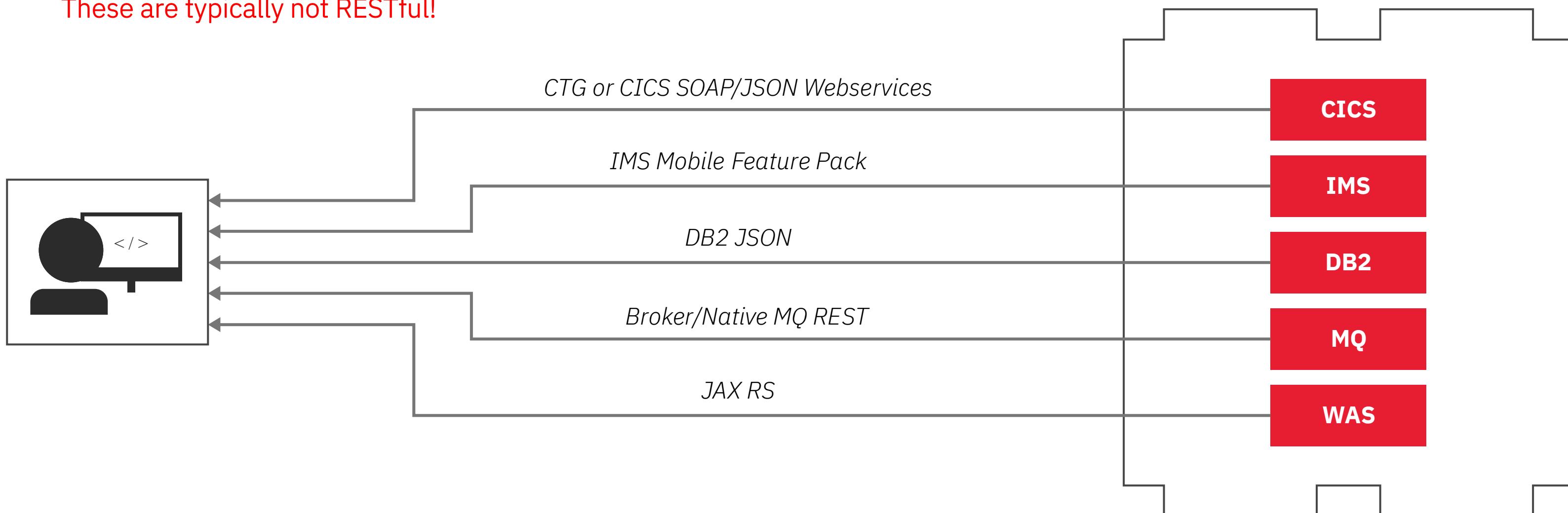


There was support for REST before z/OS Connect but..

Completely different configuration and management.

Multiple endpoints for developers to call/maintain access to.

These are typically not RESTful!



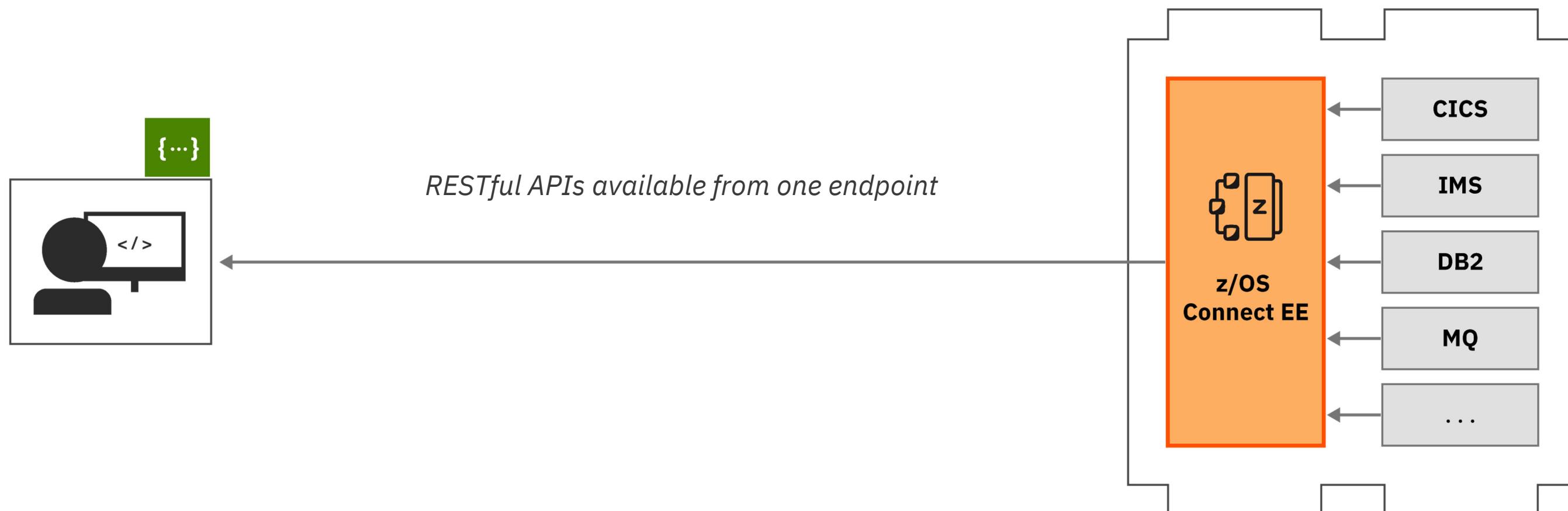


z/OS Connect provides a single-entry point

- 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.



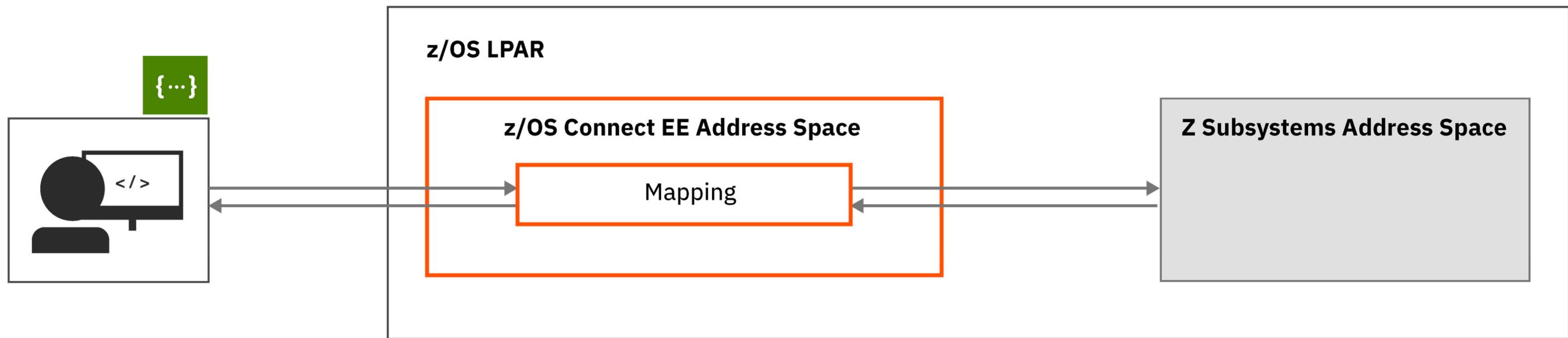


**Other than a RESTful interface,
what else does z/OS Connect provide?**



Data mapping

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

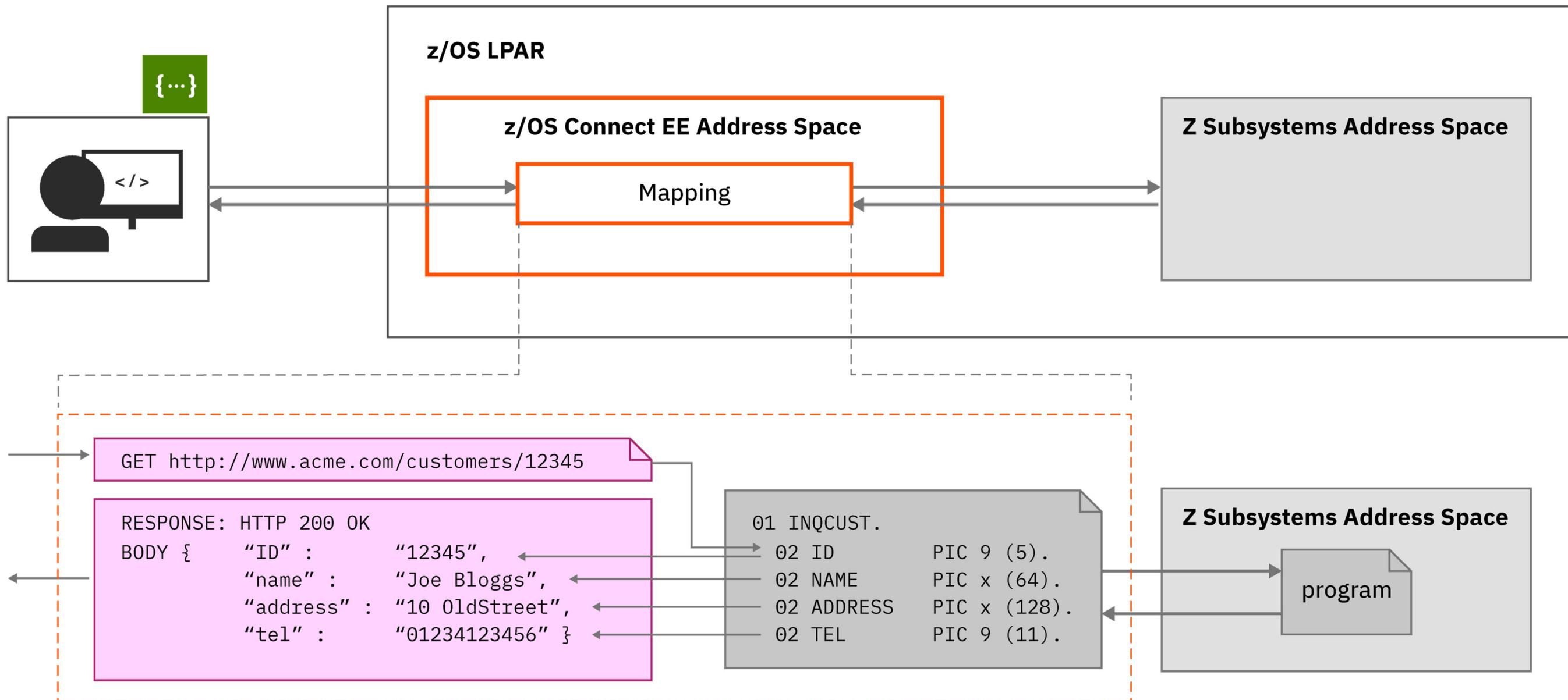


* Most z/OS subsystems depend on information in a serial data format and do not normally work with JSON request/response messages. Examples of non-JSON formats are CICS COMMAREA and CONTAINERS, IMS or MQ messages, or records stored in sequential or VSAM data sets. Data mapping and transformation refers 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

Describing an OPENAPI 2 versus an OPENAPI 3 Response Message

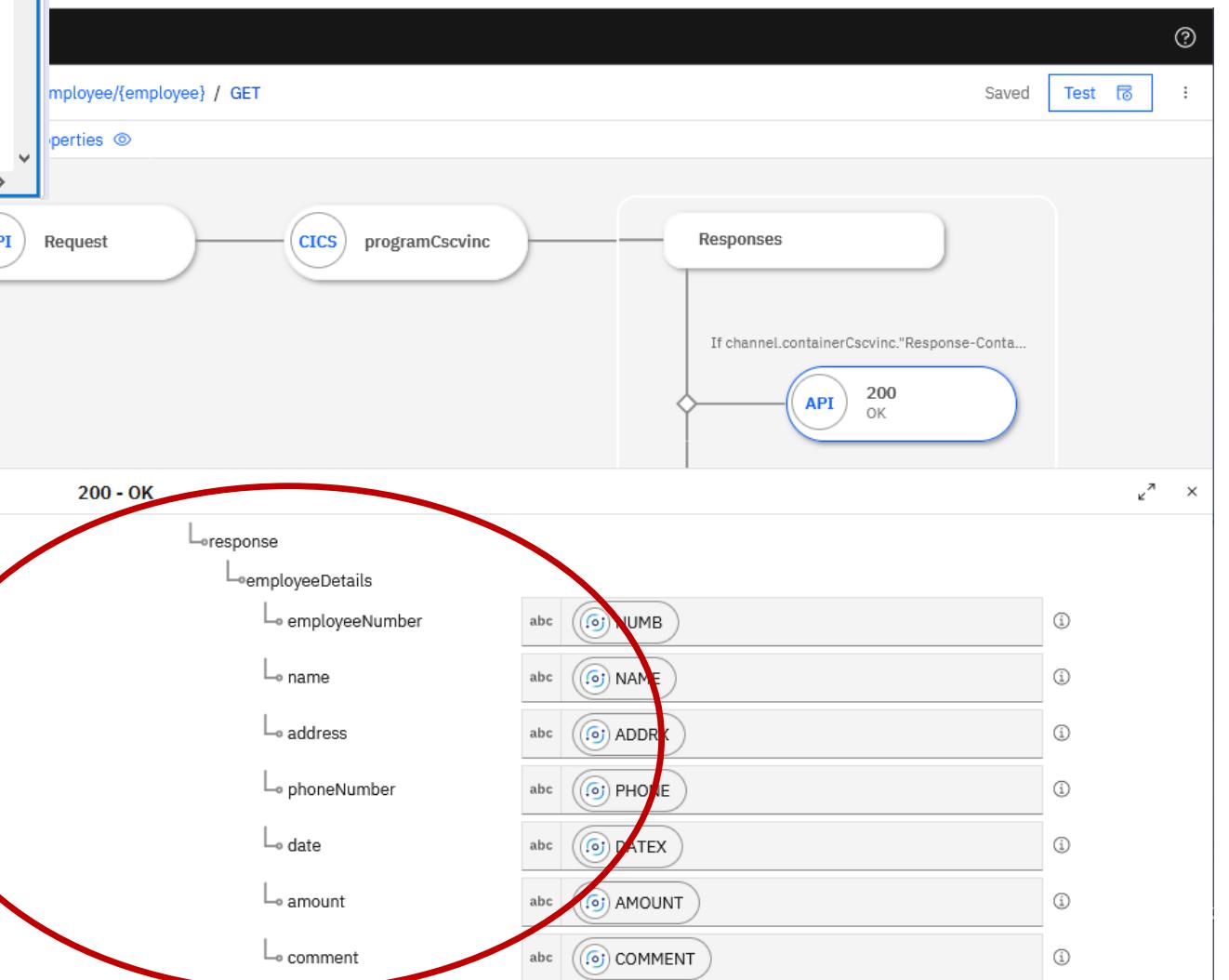


The screenshot shows the z/OS Connect Designer interface. At the top, there are tabs for 'cscvinc API' and 'response_404'. Below the tabs, the main area displays two API definitions for 'GET.employee.{employee}' and 'GET.employee.{employee}'. A red circle highlights the response body sections for both APIs, which are identical and include fields like 'cscvincContainer', 'response', 'CEIBRESP', 'CEIBRESP2', 'USERID', and 'filea'. Below these definitions is a sidebar with sections for 'Paths', 'Components', and 'Tags'. A red circle also highlights the '200 - OK' response section at the bottom, which contains a detailed schema for the 'employeeDetails' object.

Web browser - z/OS Connect Designer

mitchj@us.ibm.com

Eclipse based - z/OS Connect API Toolkit



© 2010, 2020 IBM Corporation

Slide 29



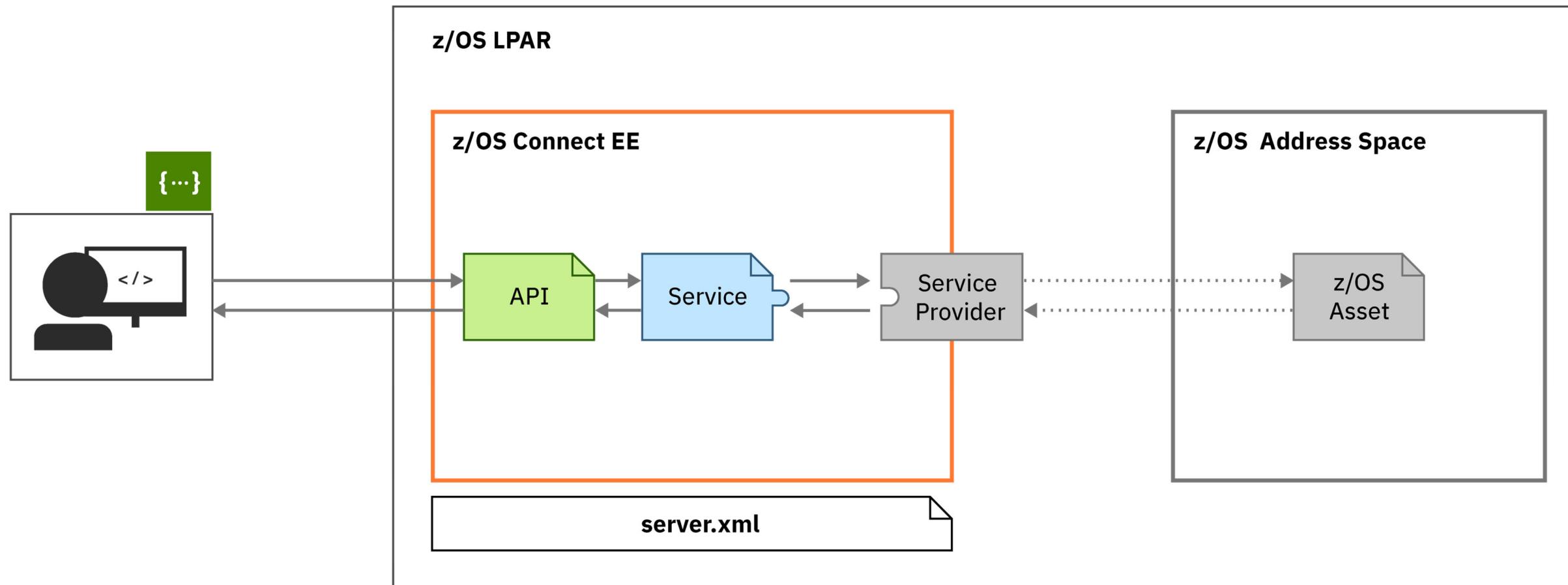
/api_toolkit/services

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)

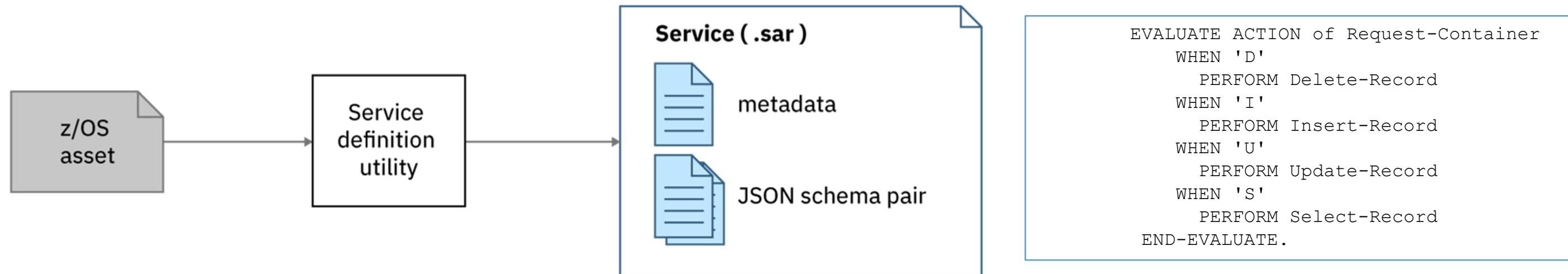
Describing the interaction (service) with the z/OS resource (OpenAPI 2)



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



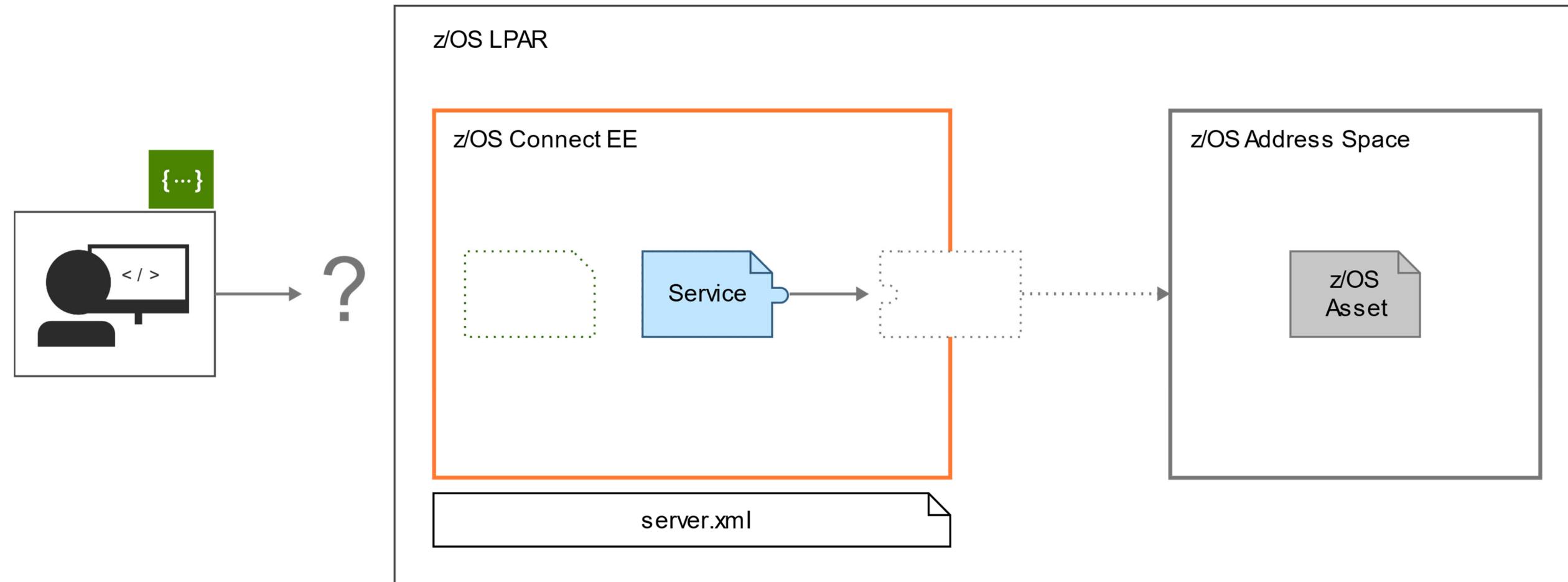
Use a system-appropriate utility to generate a service archive file for the z/OS application

- 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

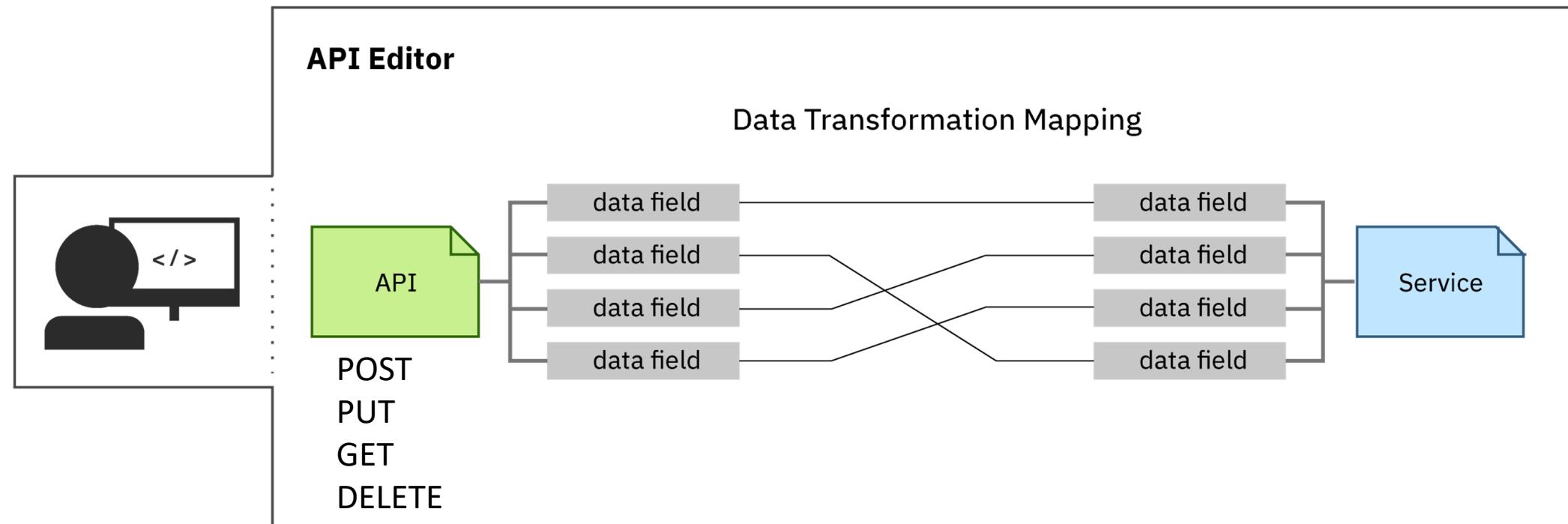


Deploy the service archive file generated in **Step 1** using the right-click deploy in **the API toolkit**.

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

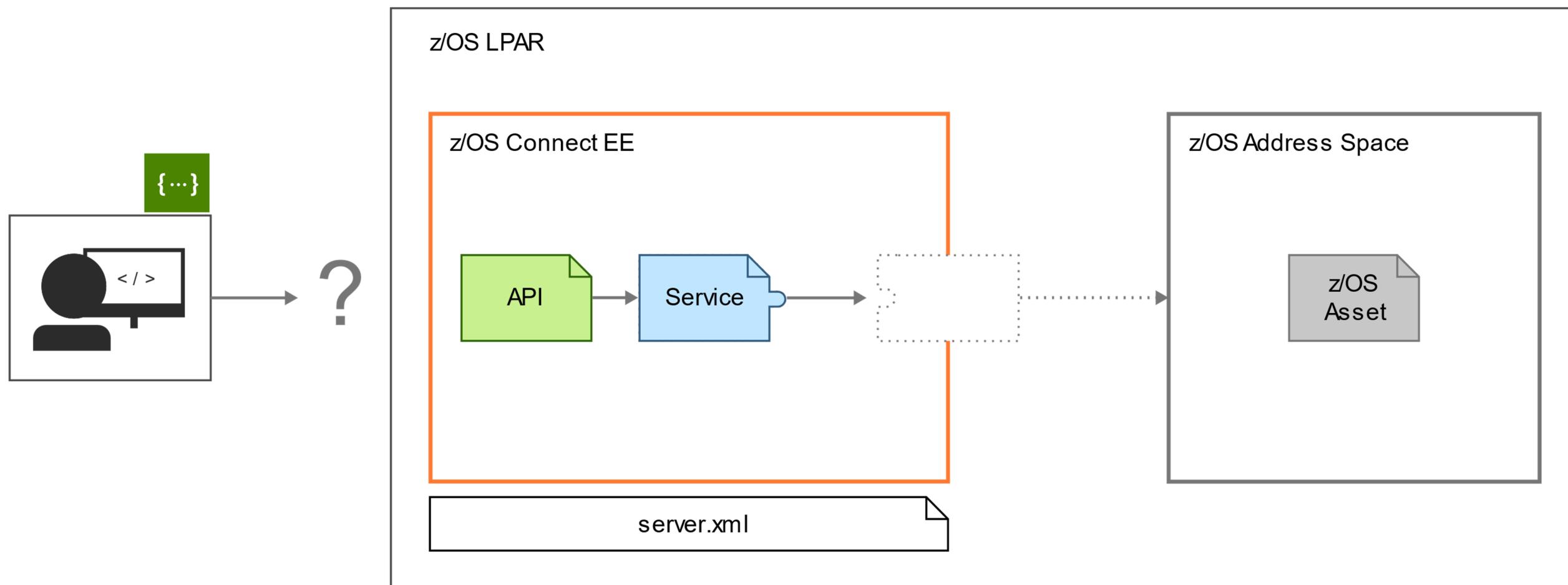


- Import the service archive file into the **API toolkit** and start designing the RESTful API.
- Provides additional data mapping
- Use the editor to describe the API and how it maps to underlying services.



Deploy the API (Open API 2)

Deploy the API archive file

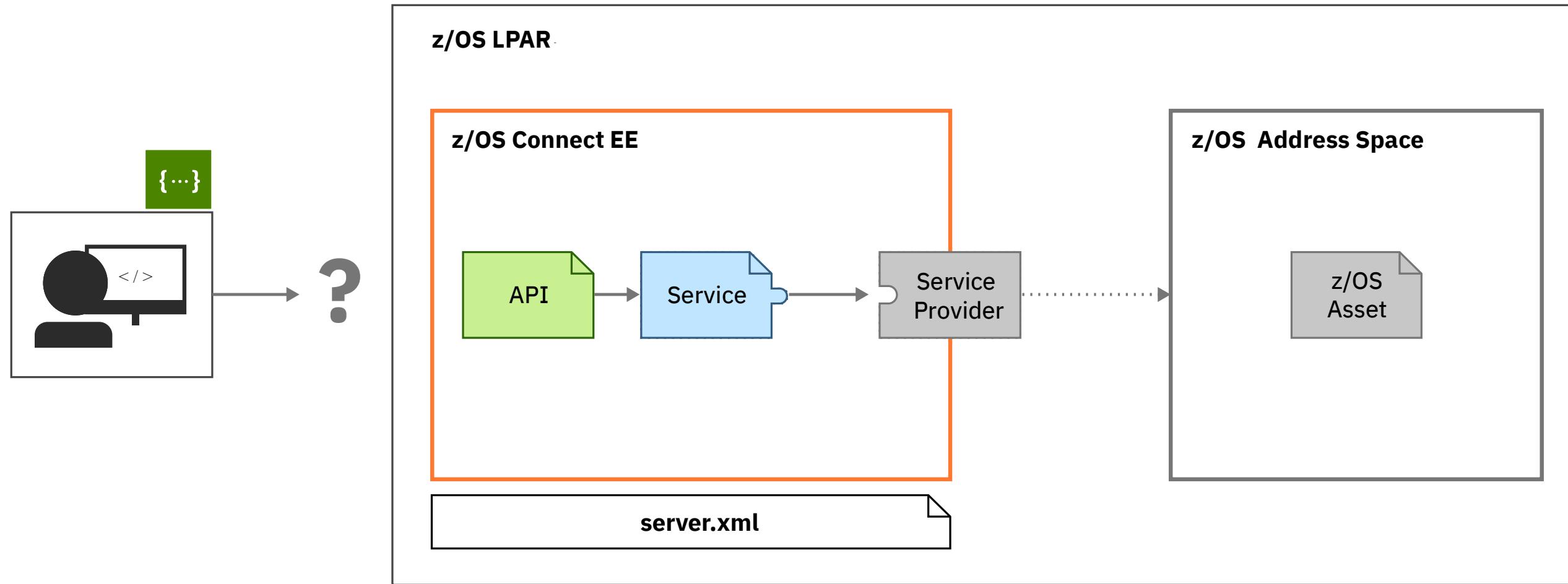


Deploy your API using the right-click deploy in **the API toolkit**



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

Configure the service provider

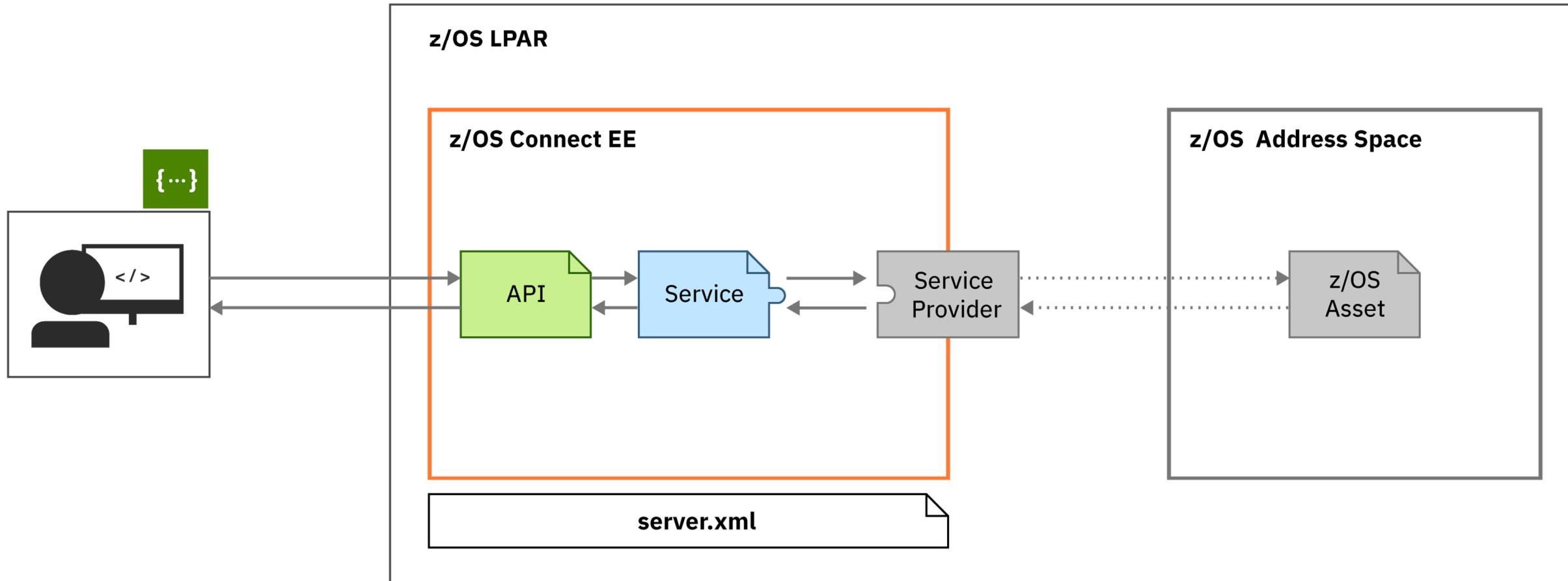


Configure the system-appropriate service provider to connect to your backend system in your `server.xml`.



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)



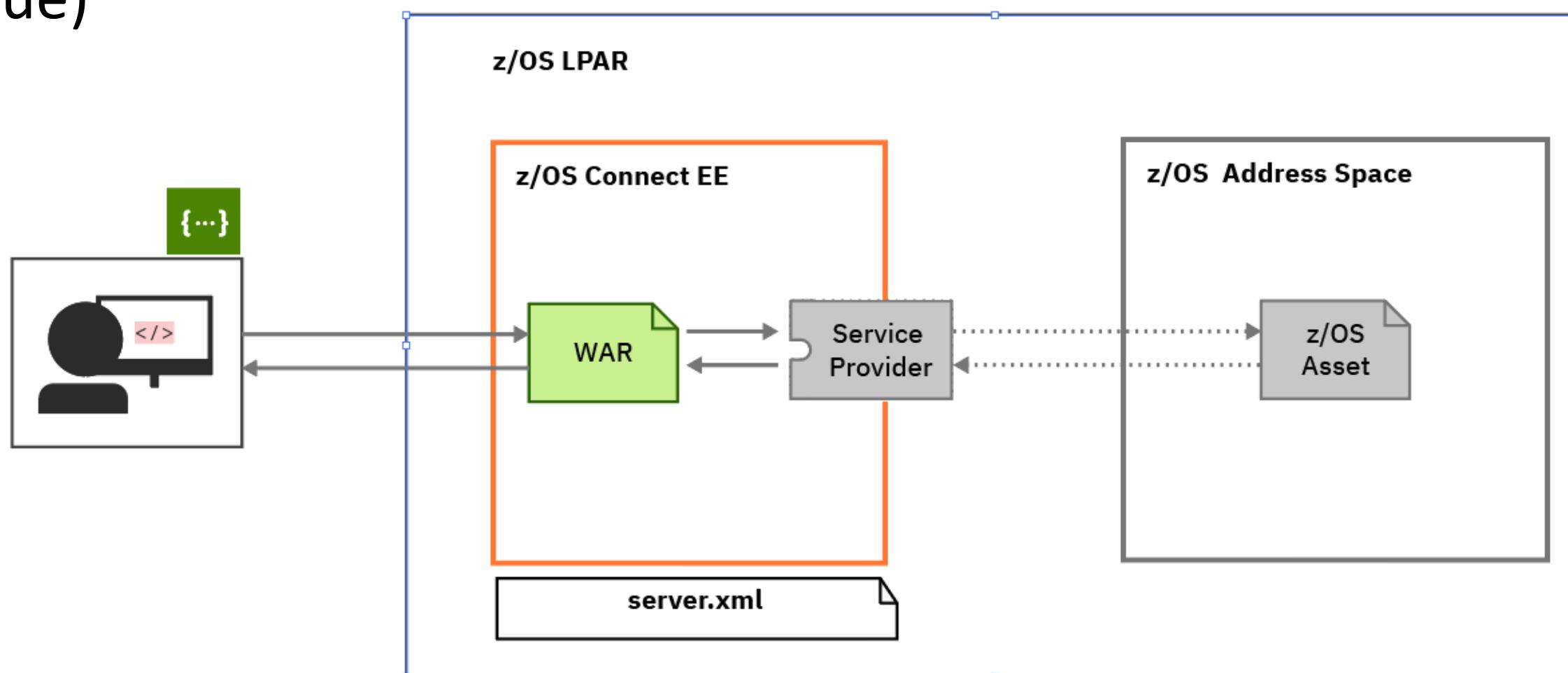
/zosConnect/designer

Simple API Development using the z/OS Connect Designer



Accessing the CICS or Db2 asset (Open API 3)

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



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

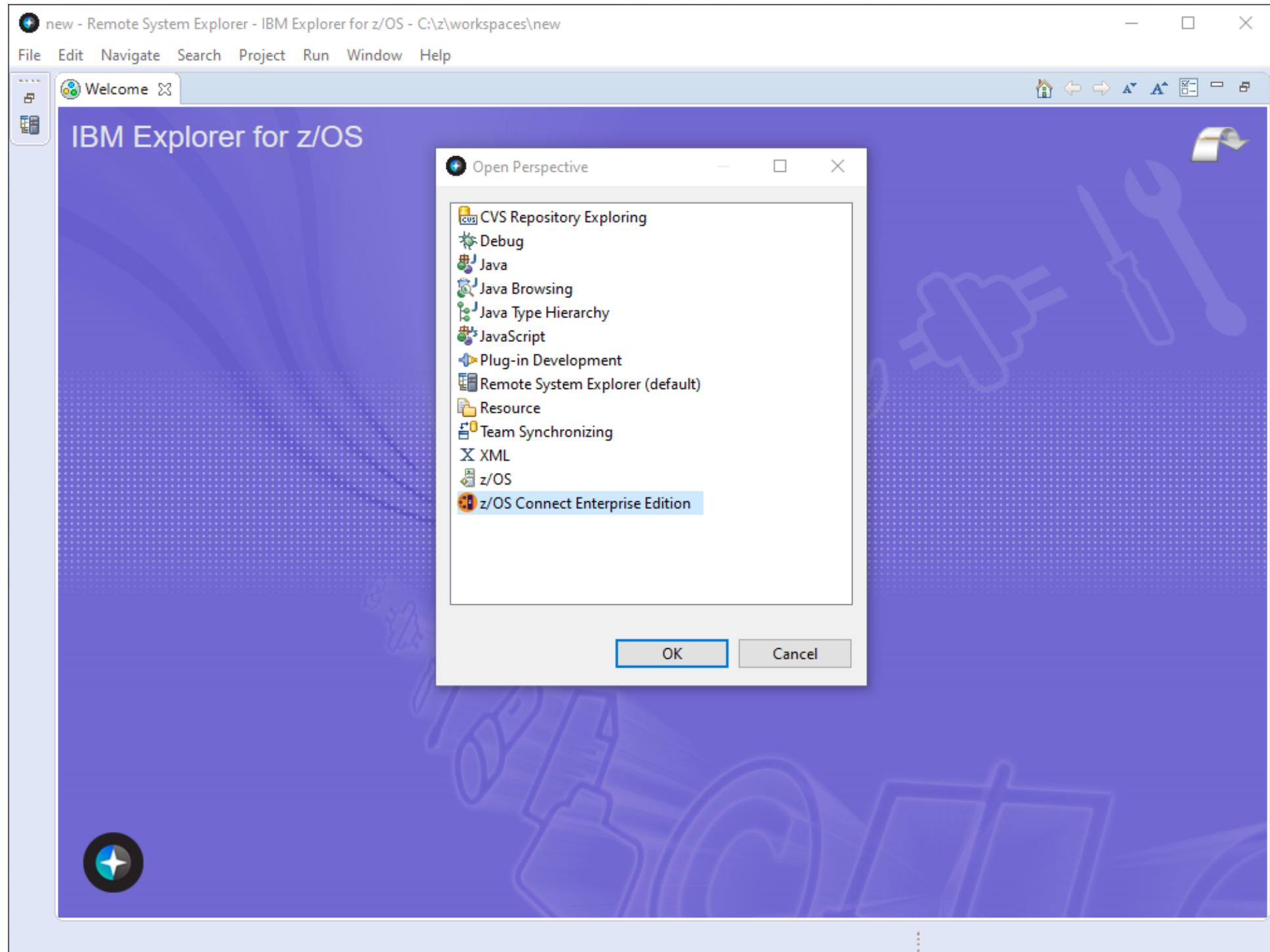


/api_toolkit/services

Simple service creation for OpenAPI 2 APIs



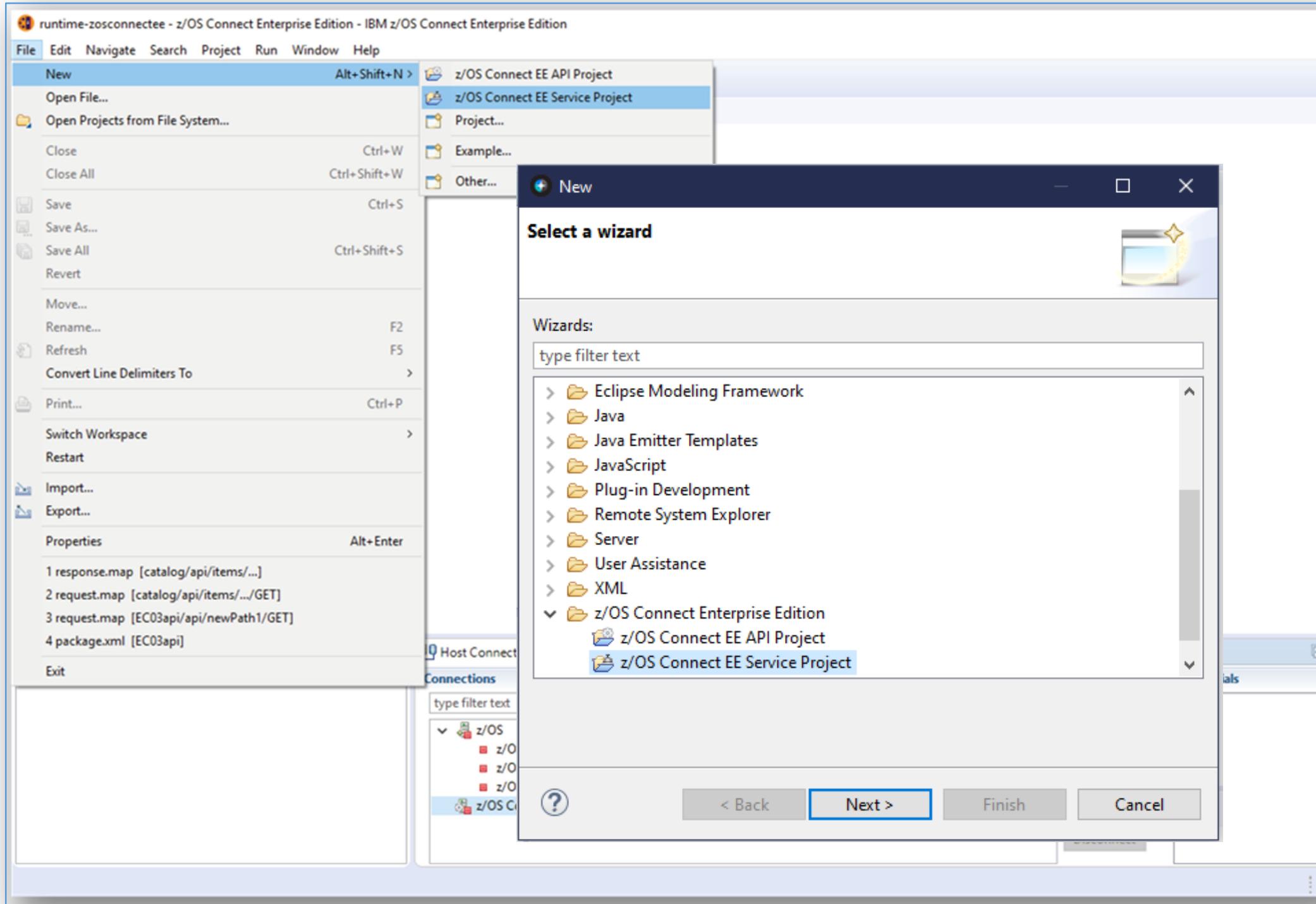
Eclipse API toolkit – Creating Services for CICS, IMS TM, IMS DB, Db2 and MQ



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



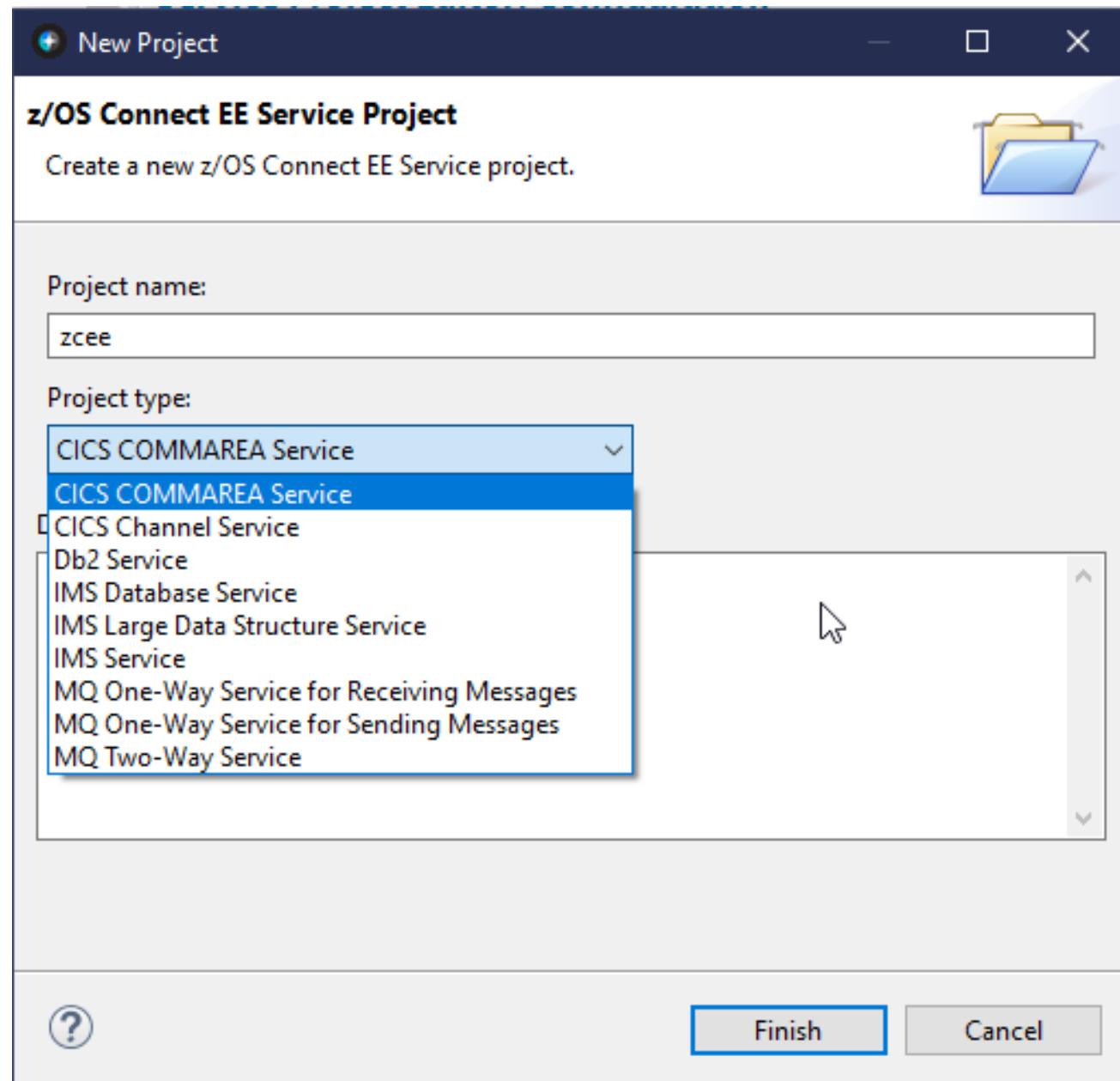
Use the **API toolkit** to create services through Eclipse-based tooling.

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



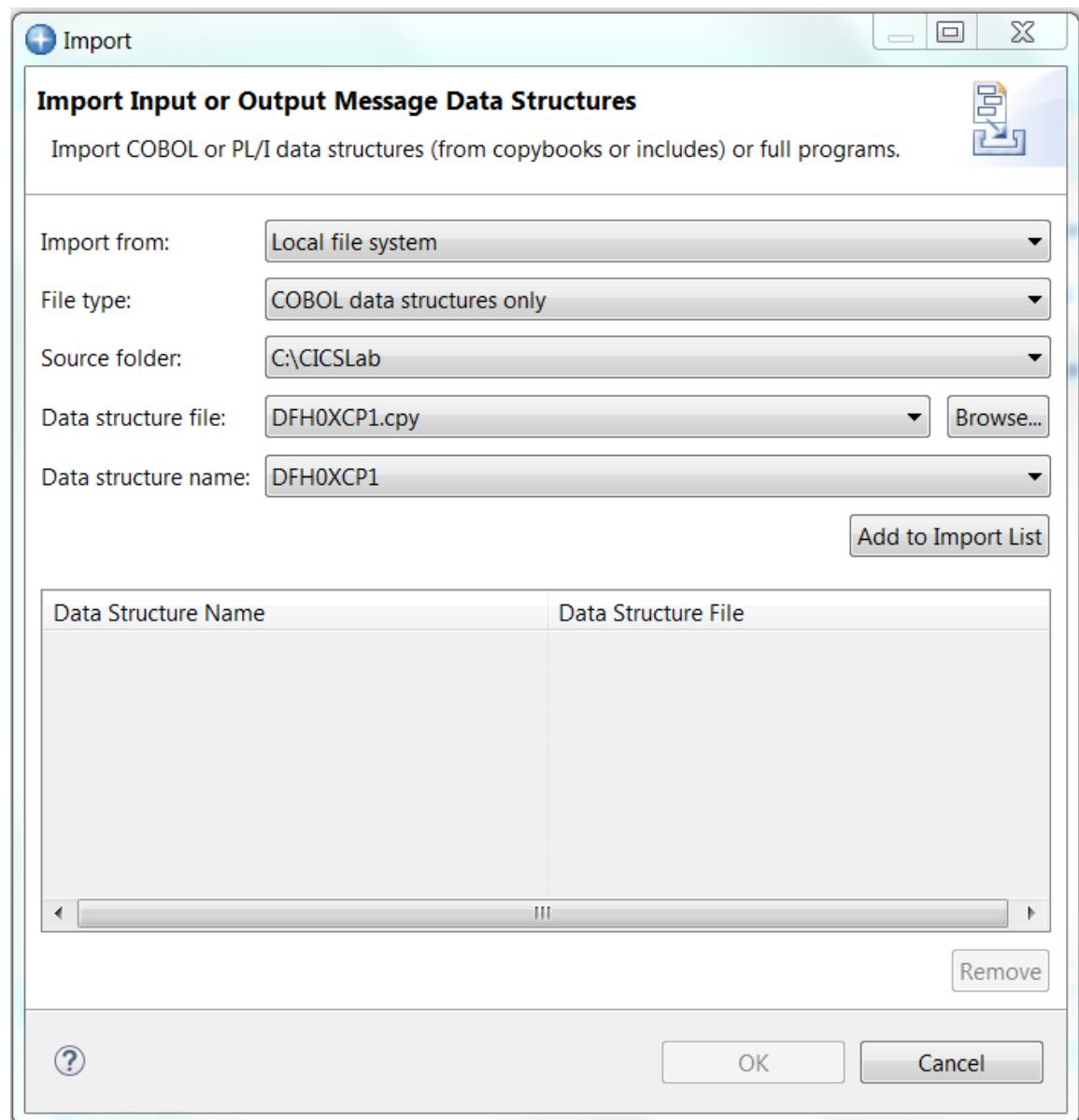
- CICS services that can invoke almost any CICS programs accessed by EXEC CICS LINK request (COMMAREA or CHANNEL). See URL <http://www.ibm.com/docs/en/cics-ts/5.6?topic=link-exception-conditions-command> for a list of EXEC CICS APIs not allowed in a program when invoked using a CICS Dynamic Program Link 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

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



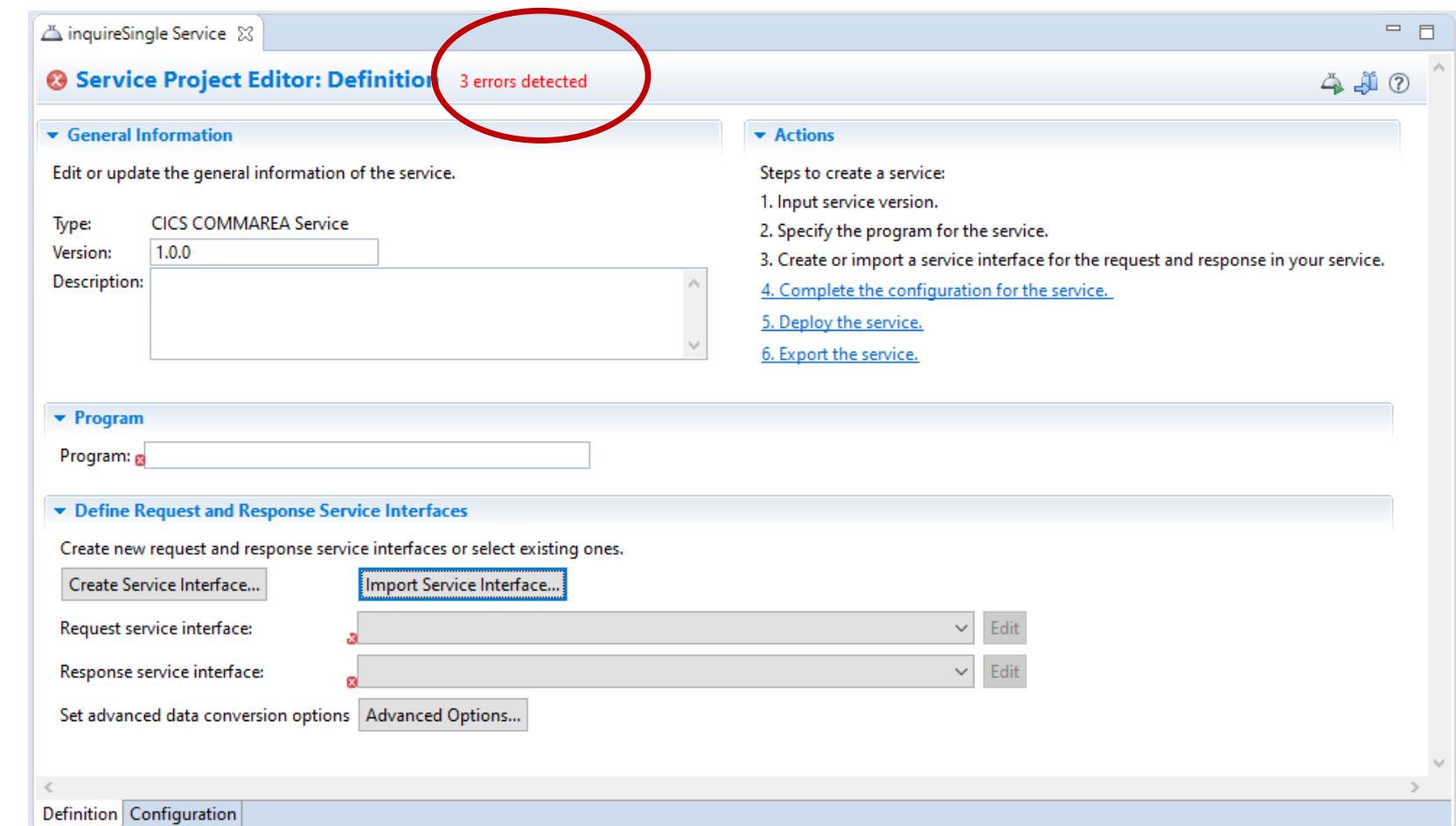
API toolkit – Creating Services for CICS, IMS TM and MQ

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 for a service named "inquireSingle Service". The table has columns for Fields, Include, Interface rename, Default Field Value, Data Type, Field Length, and Start Byte. A red circle highlights the "Default Field Value" column for the CA_REQUEST_ID field, which is set to "01INQS". Another red circle highlights the "Interface rename" column for the CA_INQUIRE_REQUEST field, which is set to "inquireSingle". A red box highlights the "Include" checkbox for the CA_ITEM_REF_REQ field, which is checked.

Fields	Include	Interface rename	Default Field Value	Data Type	Field Length	Start Byte
COMMAREA						
DFH0XCP1						
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)		CA_REQUEST_SPECIFIC		CHAR	911	88
CA_INQUIRE_REQUEST redefine	<input type="checkbox"/>	CA_INQUIRE_REQUEST		STRUCT	911	88
CA_INQUIRE_SINGLE redefines C	<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 C	<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 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_ORDER_REQUEST	<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 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

The screenshot shows two separate Service Interface Editor windows. The top window is for the service 'cscvincSelectService Service' and the bottom window is for the service 'cscvincSelectRequest'. Both windows display tables for defining request and response service interfaces. In both cases, there is a 'REQUEST_CONTAINER' section under the 'Container1' node. Within this section, the 'ACTION' field is circled in red and has a value of 'S' in the 'cscvincSelectService Service' window, and a value of 'I' in the 'cscvincSelectRequest' window. Other fields like 'USERID', 'FILEA_AREA', etc., are also listed with their respective values and data types.

The screenshot shows the 'Service Project Editor: Definition' tab for a service named 'cscvincSelectService Service'. Under the 'General Information' section, the 'Type' is listed as 'CICS Channel Service' and the 'Version' is '1.0.0'. The 'Program' field is circled in red and contains the value 'CSCVINC'. In the 'Actions' section, steps for creating a service are outlined: Input service version, Specify the program for the service, Create or import a service interface for the request and response in your project, Complete the configuration for the service, Deploy the service, and Export the service.

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



Accessing a CICS program – Transaction ID Usage

The screenshot shows the 'Service Project Editor: Configuration' window for a service named 'cscvincSelectService'. Under 'Required Configuration', 'Coded character set identifier (CCSID)' is set to 37 and 'Connection reference' is set to 'cscvinc'. Under 'Optional Configuration', 'Transaction ID' is set to 'MIJO' and 'Transaction ID usage' is set to 'EIB_AND_MIRROR'. A red circle highlights the 'EIB_AND_MIRROR' option.

EIB_ONLY

Three terminal windows labeled 'WG31 - 3270' show transaction details:

- Top Window:** TRANSACTION: CSMI PROGRAM: DFHMIRS TASK: 0008501 APPLID: CICS53Z DISPLAY: 00 STATUS: PROGRAM INITIATION. Red circles highlight 'TRANSACTION:' and 'DFHMIRS'.
- Middle Window:** TRANSACTION: MIJO PROGRAM: DFHMIRS TASK: 0008476 APPLID: CICS53Z DISPLAY: 00 STATUS: ABOUT TO EXECUTE COMMAND EXEC CICS LINK PROGRAM ('CSCVINC') SYNCONRETURN CHANNEL ('Channel') NOHANDLE. Red circles highlight 'TRANSACTION:' and 'DFHMIRS'.
- Bottom Window:** TRANSACTION: MIJO PROGRAM: CSCVINC TASK: 0008837 APPLID: CICS53Z STATUS: PROGRAM INITIATION. Red circles highlight 'TRANSACTION:' and 'CSCVINC'.

EIB_AND_MIRROR

Three terminal windows labeled 'WG31 - 3270' show transaction details:

- Top Window:** TRANSACTION: MIJO PROGRAM: DFH STATUS: PROGRAM INITIATION. Red circles highlight 'TRANSACTION:' and 'DFH'.
- Middle Window:** TRANSACTION: MIJO PROGRAM: DFH STATUS: ABOUT TO EXECUTE COM EXEC CICS LINK PROGRAM ('CSCVINC') SYNCONRETURN TRANSID ('MIJO') CHANNEL ('Channel') NOHANDLE. Red circles highlight 'TRANSACTION:', 'DFH', 'EXEC CICS LINK PROGRAM', and 'TRANSID'.
- Bottom Window:** TRANSACTION: MIJO PROGRAM: CSCVINC TASK: 0008949 APPLID: CICS53Z STATUS: PROGRAM INITIATION. Red circles highlight 'TRANSACTION:' and 'CSCVINC'.

Legend at the bottom right:

- 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

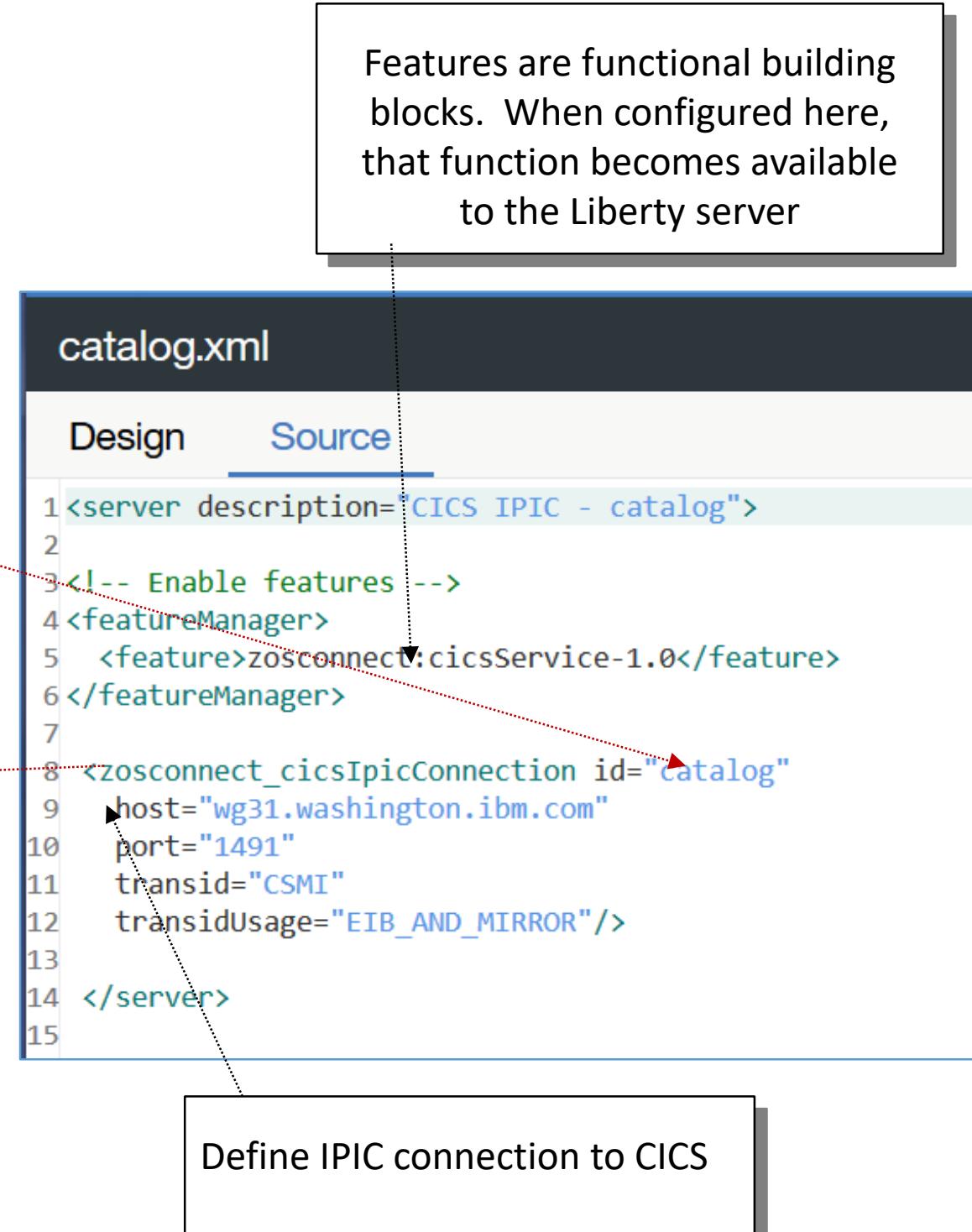
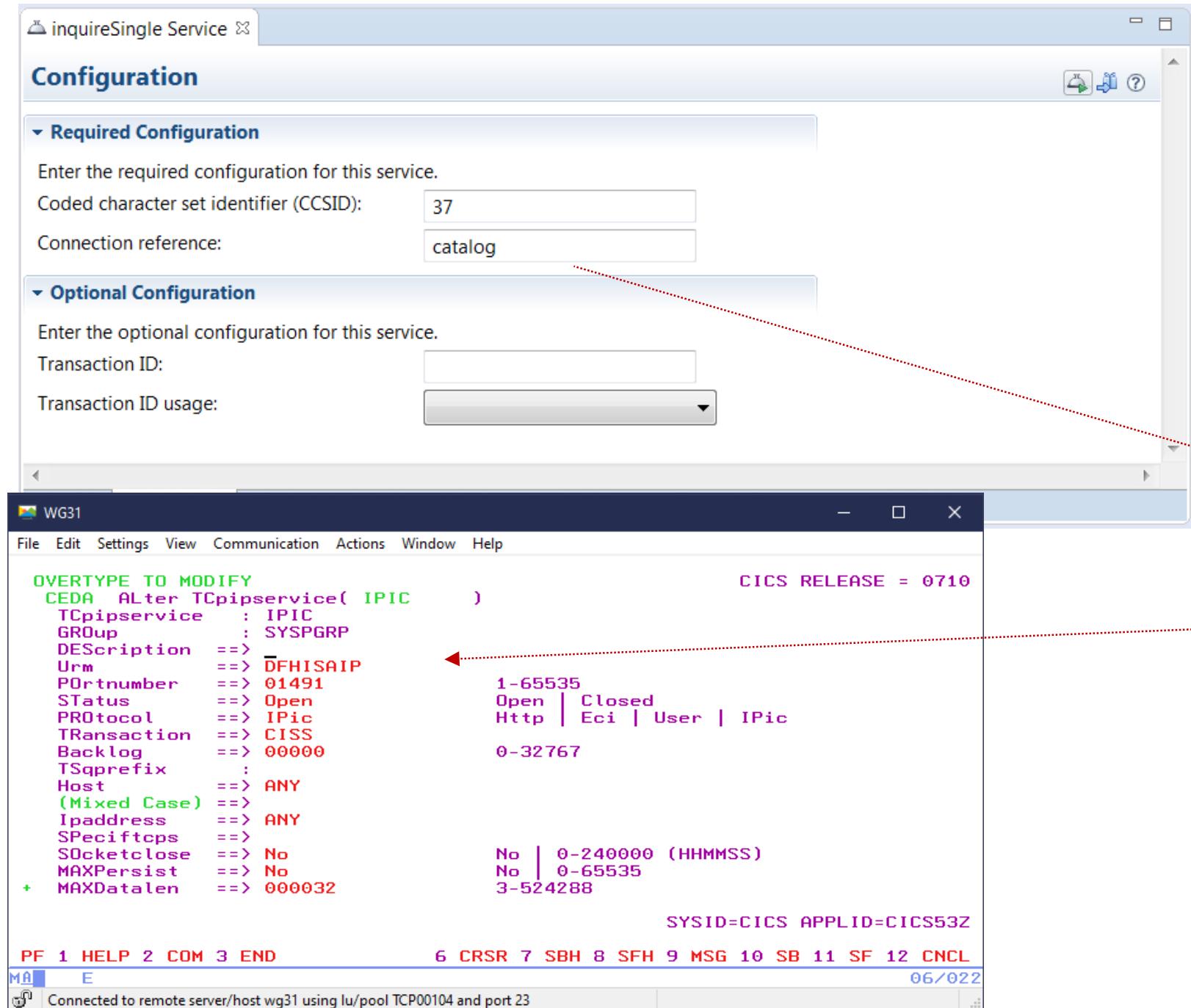
- **Transaction ID** attaches a CICS transaction (CSMI is the default) that starts the CICS DFHMIRS program.

- **Transaction ID Usage** attribute useful for:
 - Transaction security requirements
 - Db2 plan selection
 - Transaction classification and reporting



Accessing a CICS program using IPIC

The server.xml file is the key configuration file:





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
```

Service Interface Editor

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

Fields	Include	Interface Rename	Default Field Value	Data Type	Field Length
ivtnoDisplayRequest					
Segment 1					
INPUT_MSG		phonebookRequest			
IN_LL	<input type="checkbox"/>	IN_LL		SHORT	2
IN_ZZ	<input type="checkbox"/>	IN_ZZ		SHORT	2
IN_TRANCDE	<input type="checkbox"/>	IN_TRANCDE		CHAR	10
IN_COMMAND	<input type="checkbox"/>	IN_COMMAND	IVTNO	CHAR	8
IN_LAST_NAME	<input checked="" type="checkbox"/>	lastName	DISPLAY	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)

Service Project Editor: Configuration

Required Configuration
Enter the required configuration for this service.
Connection profile: **IMSCONN**
Interaction profile: **IMSINTER** (circled in red)

Optional Configuration
Enter the optional configuration for this service.
IMS destination override:
Program name:



IMS Connections and Interactions (server XML)

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

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

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

Interaction

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

IMS Connect HWSCFG

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



API toolkit – Creating Services for IMS DB

Creating a service project from the IMS Catalog

Service Project Editor: Definition

General Information

Edit or update the general information of the service.

Type: IMS Database Service
Version: 1.0.0
Description:

Actions

Steps to create a service:

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.](#)
- [6. Export the service.](#)

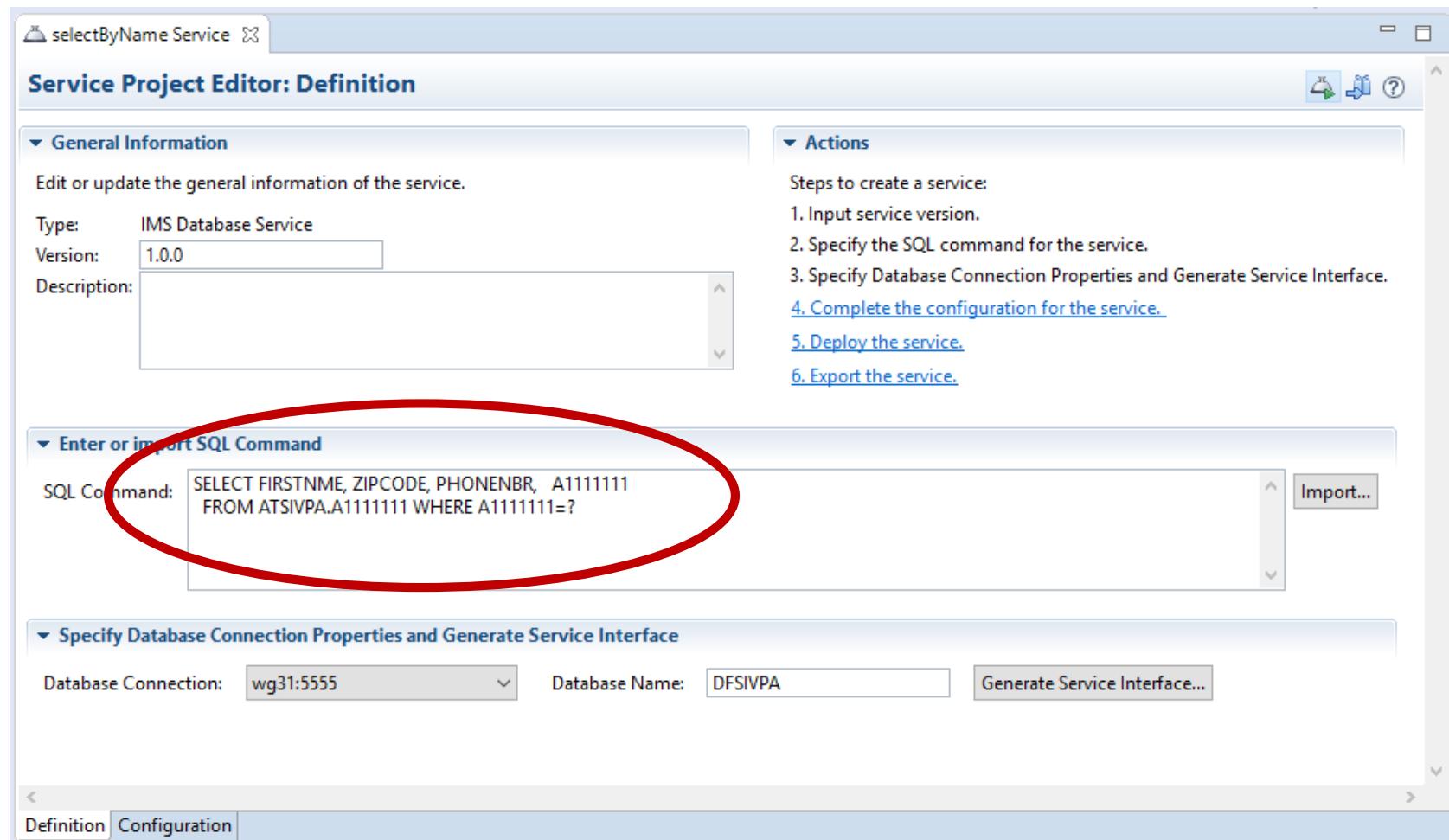
Enter or import SQL Command

SQL Command: `SELECT FIRSTNME, ZIPCODE, PHONENBR, A1111111
FROM ATSVPA.A1111111 WHERE A1111111=?`

Specify Database Connection Properties and Generate Service Interface

Database Connection: wg31:5555 Database Name: DFSIVPA [Generate Service Interface...](#)

Definition Configuration



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=DFSVVD1,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=FIRSTNME,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=IDX,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 with the title "response". The main area is a table titled "Fields" with columns: Fields, Include, Interface Rename, Default Field Value, Data Type, and Field Length. The table lists a hierarchy of fields under "selectByName_Response" and "Segment 1". A red circle highlights the "Include" column for the "Result [0..*]" row, which has several checkboxes checked. A red box highlights the "Interface Rename" column for the "result" row, which contains the values "response", "result", "firstName", "zipCode", "phoneNuber", and "lastName".

Fields	Include	Interface Rename	Default Field Value	Data Type	Field Length
selectByName_Response					
Segment 1					
Output Columns					
Result [0..*]	<input checked="" type="checkbox"/>	response result firstName zipCode phoneNuber lastName		ARRAY	0
FIRSTNME				CHAR	10
ZIPCODE				CHAR	7
PHONENBR				CHAR	10
A1111111				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: DFSIVPACConn

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, TMEMBER=OTMAMEM)
IMSPLEX=(MEMBER=IMS14HWS, TMEMBER=PLEX1)
ODACCESS=(ODBMAUTOCONN=Y,
DRDAPORT=(ID=5555,PORTTMOT=6000), ODBMTMOT=6000)
```



API toolkit – Creating Services for MQ

Creating a “POST” service interface definition

The screenshot shows two windows from the API toolkit:

- Service Interface Editor:** A table where fields are mapped to an interface. A red box highlights the "Interface Rename" column for the "loan application" row, showing renamed fields like "name", "credit score", etc. A red circle highlights the "Include" column for the "MINILOAN_COMMAREA" row, which has several checked checkboxes.
- Service Project Editor: Configuration:** Configuration settings for a 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

```
1 <server description="MQ Service Provider">
2
3   <featureManager>
4     <feature>zosconnect:mqService-1.0</feature>
5   </featureManager>
6
7   <variable name="wmqJmsClient.rar.location"
8     value="/usr/lpp/mqm/V9R1M1/java/lib/jca/wmq.jmsra.rar"/>
9   <wmqJmsClient nativeLibraryPath="/usr/lpp/mqm/V9R1M1/java/lib"/>
10
11  <zosconnect_services>
12    <service name="mqPutService">
13      <property name="useCallerPrincipal" value="false"/>
14    </service>
15  </zosconnect_services>
16
17  <connectionManager id="ConMgr1" maxPoolSize="5"/>
18
19  <jmsConnectionFactory id="qmgrCf" jndiName="jms/qmgrCf">
20    connectionManagerRef="ConMgr1">
21    <properties.wmqJMS transportType="CLIENT"
22      queueManager="ZMQ1"
23      channel="LIBERTY.DEF.SVRCONN"
24      hostName="wg31.washington.ibm.com"
25      port="1422" />
26  </jmsConnectionFactory>
27
28  <jmsQueue id="q1" jndiName="jms/default">
29    <properties.wmqJms
30      baseQueueName="ZCEE.DEFAULT.MQZCEE.QUEUE"
31      CCSID="37"/>
32  </jmsQueue>
33
34</server>
35
```



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: jms/qmgrCf

Request destination JNDI name: jms/requestQueue

Reply destination JNDI name: jms/replyQueue

Wait interval: 3000

MQMD format: MQSTR

Coded character set identifier (CCSID): 37

Is message persistent:

Reply selection: msgIDToCorrelID

Expiry: -1

Definition Configuration

mq.xml

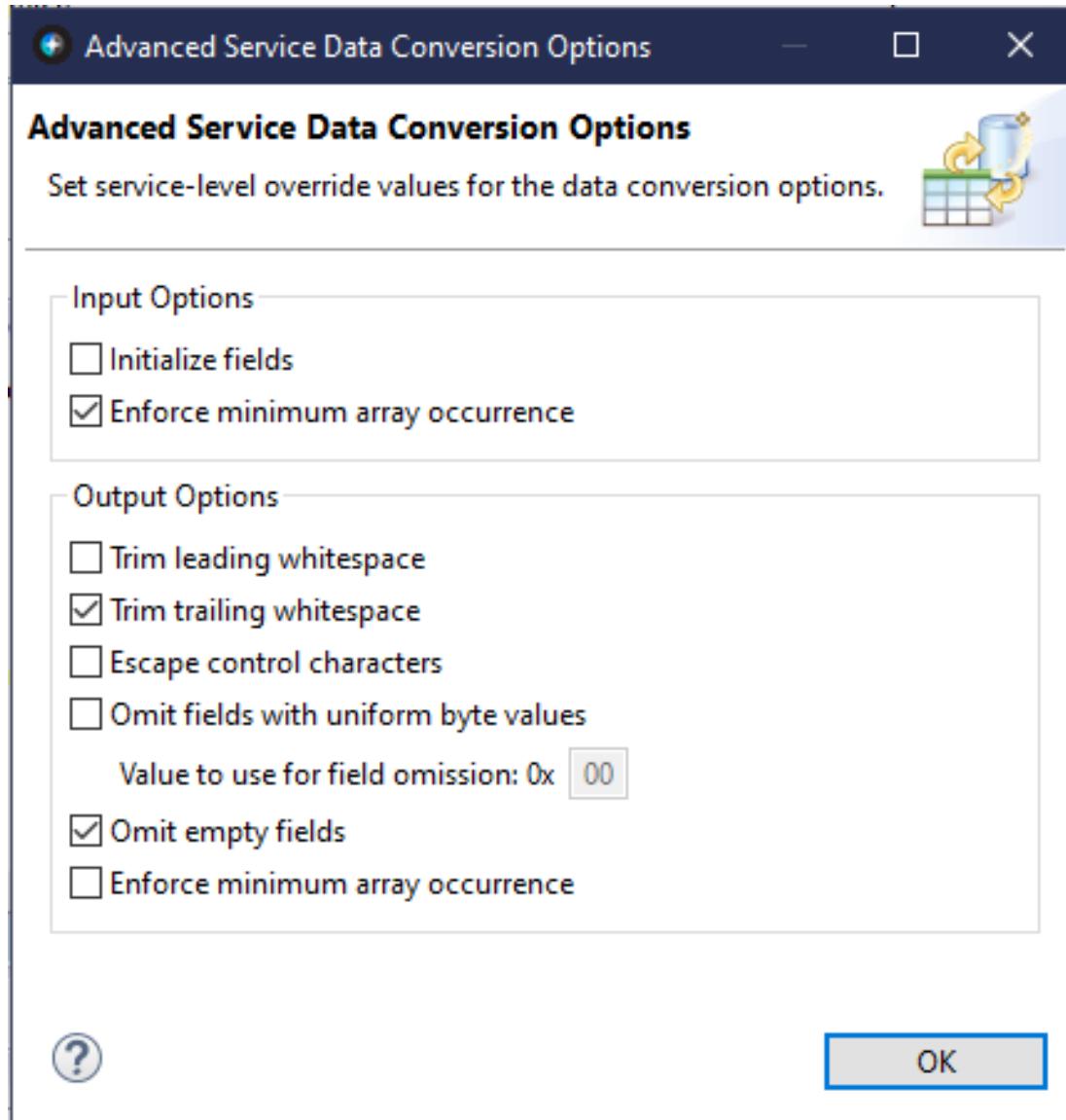
Read only Close

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="BINDINGS"
15     queueManager="QMZ1" />
16 </jmsConnectionFactory>
17
18 <jmsConnectionFactory id="qmgrCf2" jndiName="jms/qmgrCf2"
19   connectionManagerRef="ConMgr1">
20   <properties.wmqJMS transportType="CLIENT"
21     queueManager="ZMQ1"
22     channel="LIBERTY.DEF.SVRCONN"
23     hostName="wg31.washington.ibm.com"
24     port="1422" />
25 </jmsConnectionFactory>
26
27 <jmsQueue id="q1" jndiName="jms/default">
28   <properties.wmqJms
29     baseQueueName="ZCONN2.DEFAULT.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", FIRSTNME 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

?

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...
Import from local machine...

Definition Configuration

Definition	Configuration
Select an employee from table USER1.EMPLOYEE	/services/zCEEService/selectEmployee
Delete an employee from table USER1.EMPLOYEE	/services/zCEEService/deleteEmployee
Display an employee in table USER1.EMPLOYEE	/services/zCEEService/displayEmployee
Insert an employee into table USER1.EMPLOYEE	/services/zCEEService/insertEmployee
Select employees by departments	/services/zCEEService/selectByDepartments
Select an employee based on job and department	/services/zCEEService/selectByRole
Select an employee from table USER1.EMPLOYEE	/services/zCEEService/selectEmployee/V1
Select an employee from table USER1.EMPLOYEE	/services/zCEEService/selectEmployee/V2
Update an employee in table USER1.EMPLOYEE	/services/zCEEService/updateEmployee

Import Cancel

The service developer retrieves details about the Db2 REST services

Note there is no service interface editor available



Accessing a Db2 REST service resource

The screenshot shows the Service Project Editor for a project named "selectEmployee Service". The "Required Configuration" section displays connection reference "db2conn". The "db2pass.xml" file is open in the editor, showing XML code for a server configuration. Red arrows point from the "db2conn" reference in the configuration to the "zosconnect_zosConnectServiceRestClientConnection" element in the XML, and from the "DSN2APPL" application name in the configuration to the "applName" attribute in the XML.

*selectEmployee Service X

Service Project Editor: Configuration

Required Configuration

Enter the required configuration for this service.

Connection reference: db2conn

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

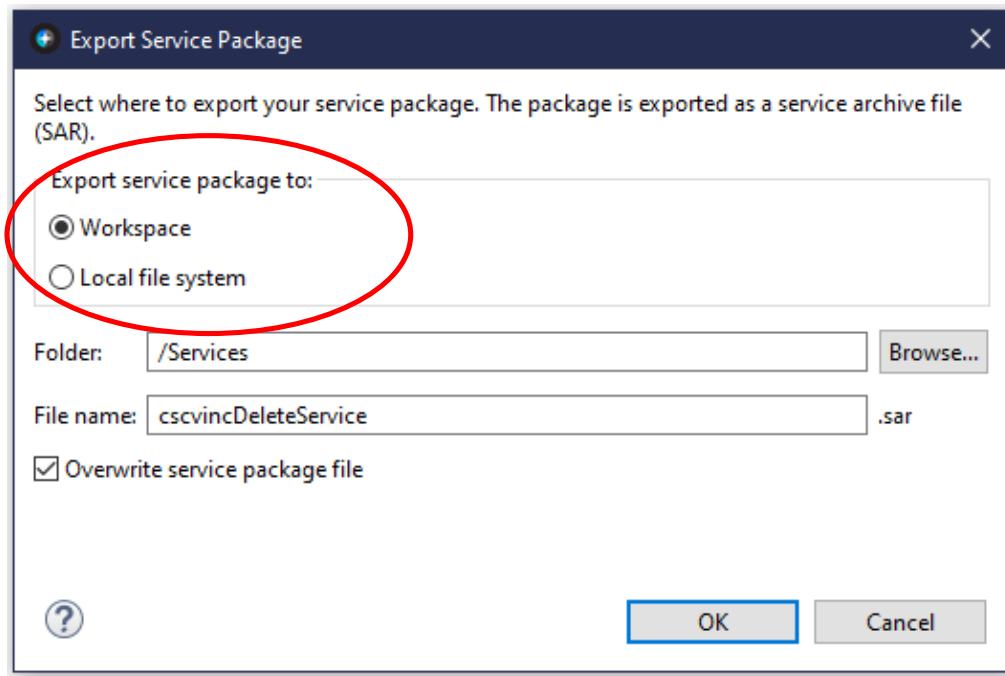
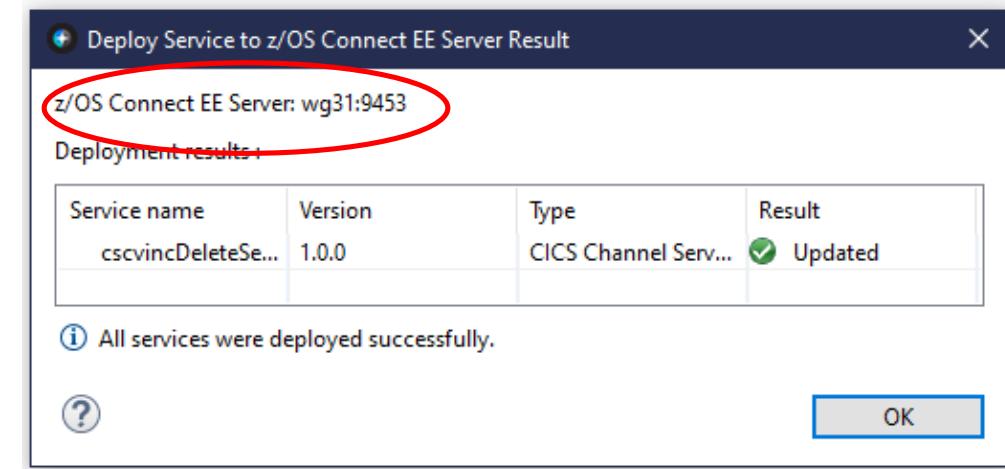
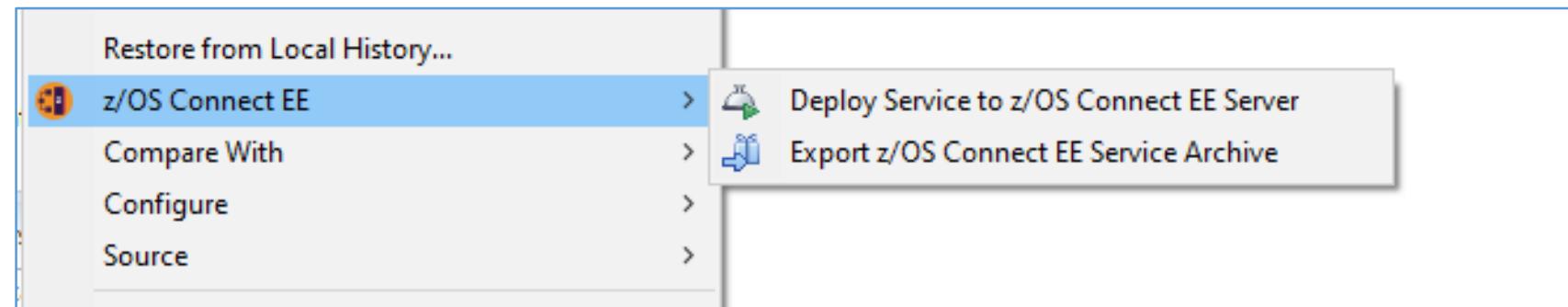
```
1 <server description="DB2 REST">
2
3   <zosconnect_zosConnectServiceRestClientConnection id="db2conn">
4     host="wg31.washington.ibm.com"
5     port="2446"
6     basicAuthRef="dsn2Auth" />
7
8   <zosconnect_zosConnectServiceRestClientBasicAuth id="dsn2Auth">
9     applName="DSN2APPL"/>
10
11 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



API toolkit – Services Editor

Server connection and Services deployment

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





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 application interface. In the top navigation bar, there are tabs for File, Edit, View, Help, + New, Import, Runner, and My Workspace. The main workspace shows three requests: a GET request, a POST request, and the current POST request titled "Untitled Request". The current request is a POST to <https://wg31.washington.ibm.com:9453/zosConnect/services>. The "Body" tab is selected, showing the file "selectEmployee.sar" has been chosen as a binary file. The "Headers" tab shows 15 headers. The "Tests" and "Settings" tabs are also visible. Below the request details, the response pane shows a JSON object with various service details. The status bar at the bottom indicates a 201 Created response.

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://wg31.washington.ibm.com:9453/zosConnect/services/selectEmployee", "serviceInvokeURL": "https://wg31.washington.ibm.com:9453/zosConnect/services/selectEmployee?action=invoke", "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"}}
```



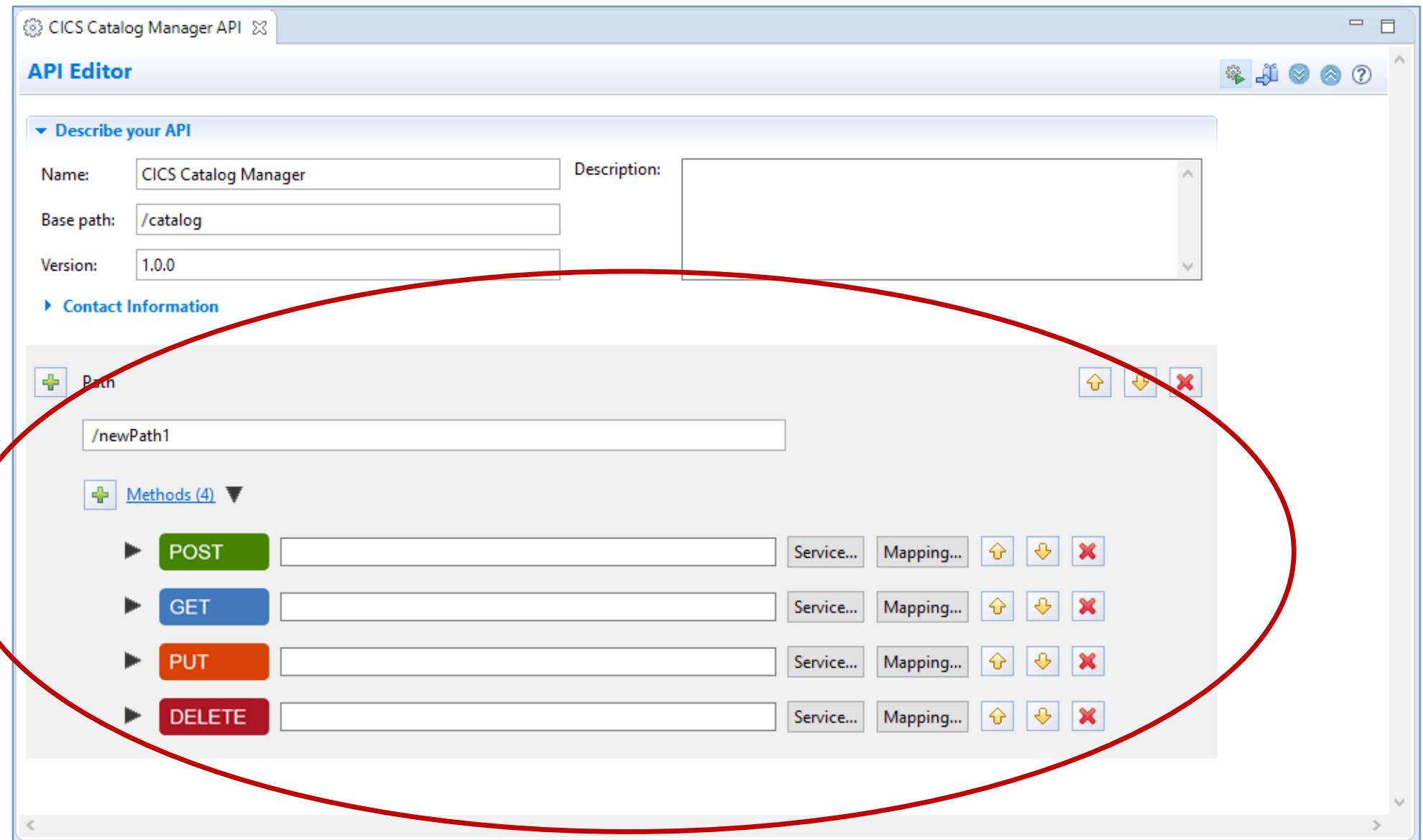
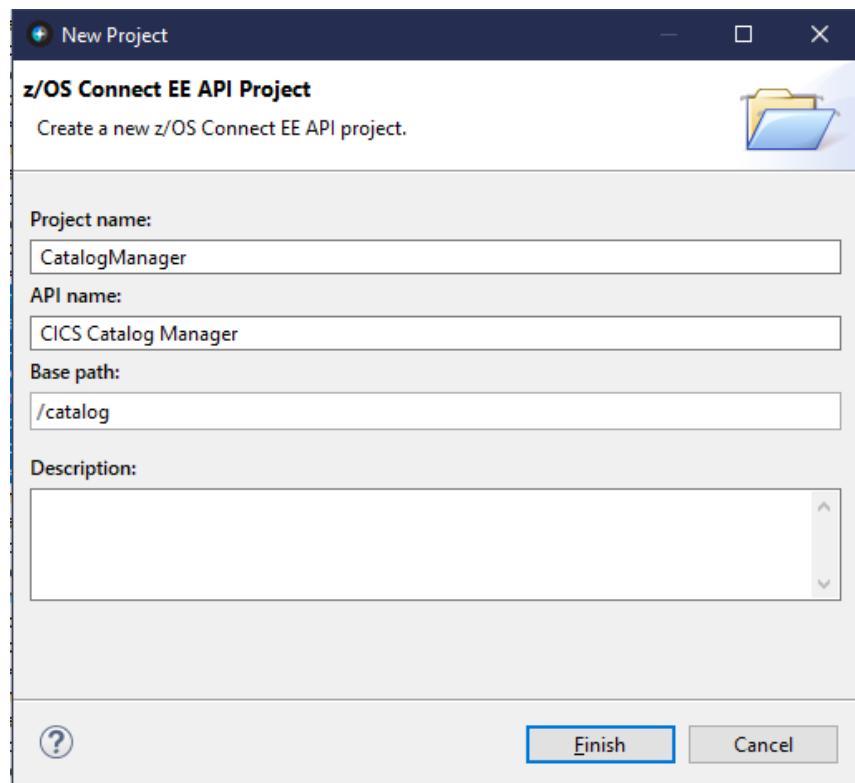
Once we have a Service Archive (SAR) What's next?

Quick and easy **API mapping**.

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



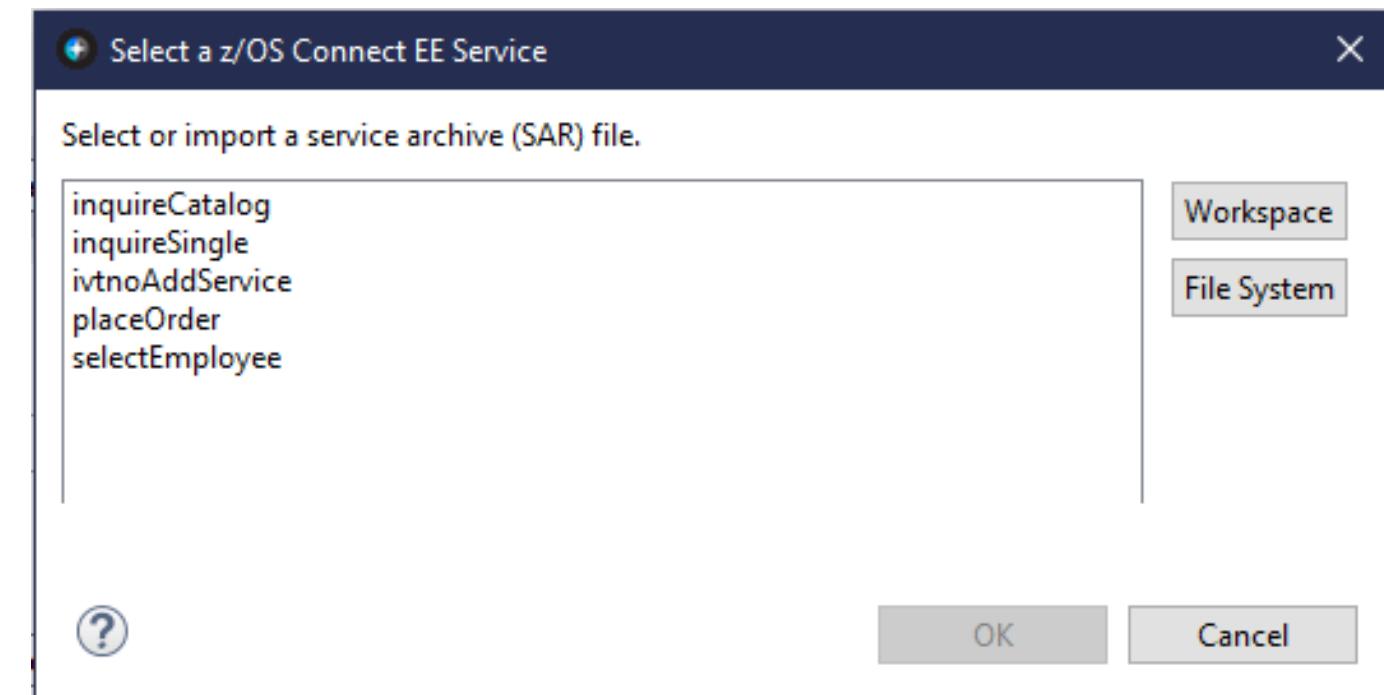
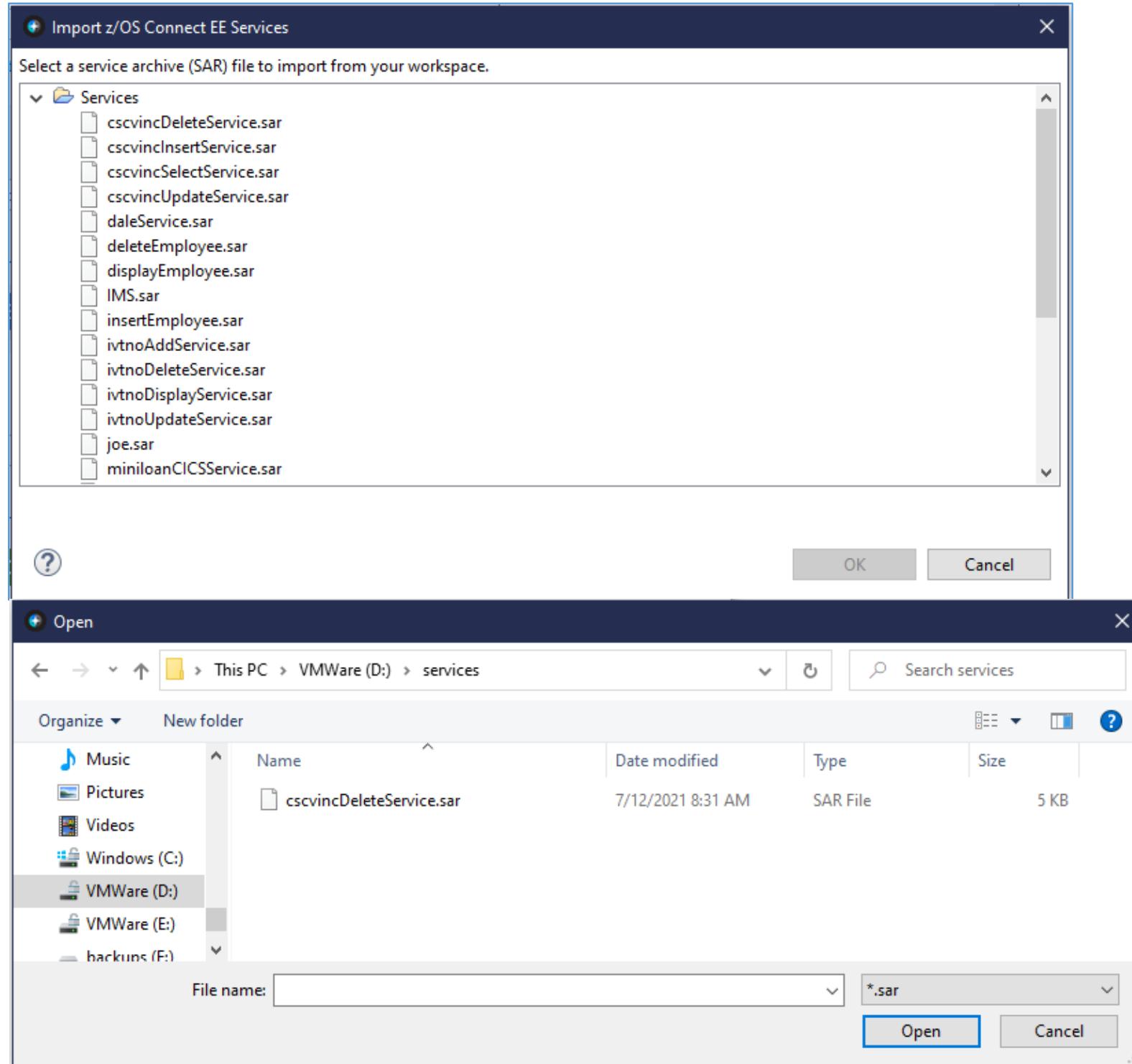
API toolkit – API Editor



The screenshot shows the 'CICS Catalog Manager API' API Editor window. The 'Describe your API' section includes fields for Name (CICS Catalog Manager), Description, Base path (/catalog), and Version (1.0.0). Below this is a 'Contact Information' section. The main area contains a 'Path' entry (/newPath1) and a 'Methods (4)' section with four entries: POST, GET, PUT, and DELETE, each with associated service and mapping fields and up/down arrows.



Importing the service archives files





API toolkit – API Editor

The screenshot shows the API Editor interface with three defined APIs:

- catalog API**:
 - Name: catalog
 - Description: (empty)
 - Base path: /catalog
 - Version: 1.0.0
- order**:
 - Path: /order
 - Methods (2):
 - POST placeOrder
 - PUT selectEmployee
- item**:
 - Path: /item/{itemID}
 - Methods (1):
 - GET inquireSingle

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

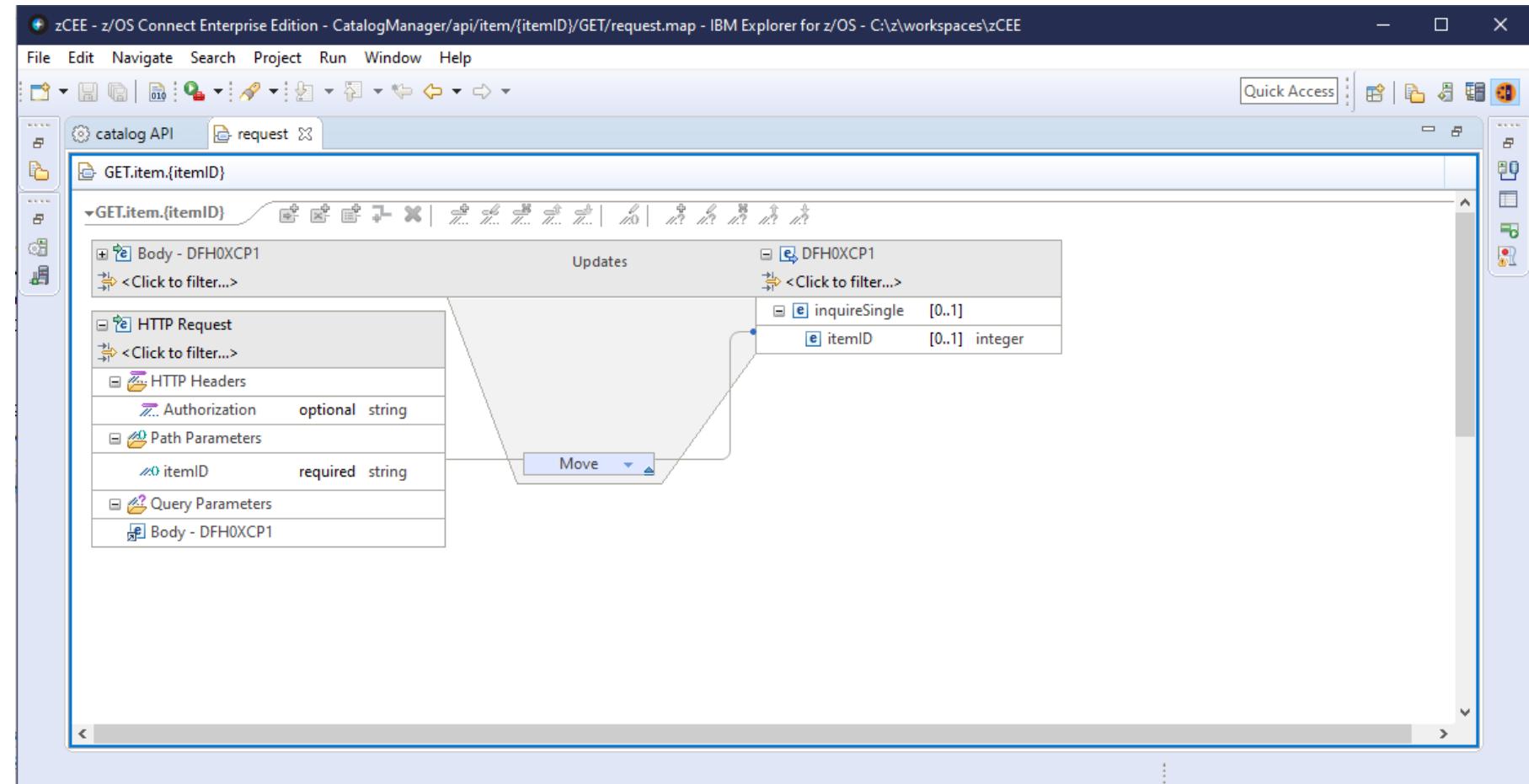
Once you define your API, you can map backend services to each request.

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.

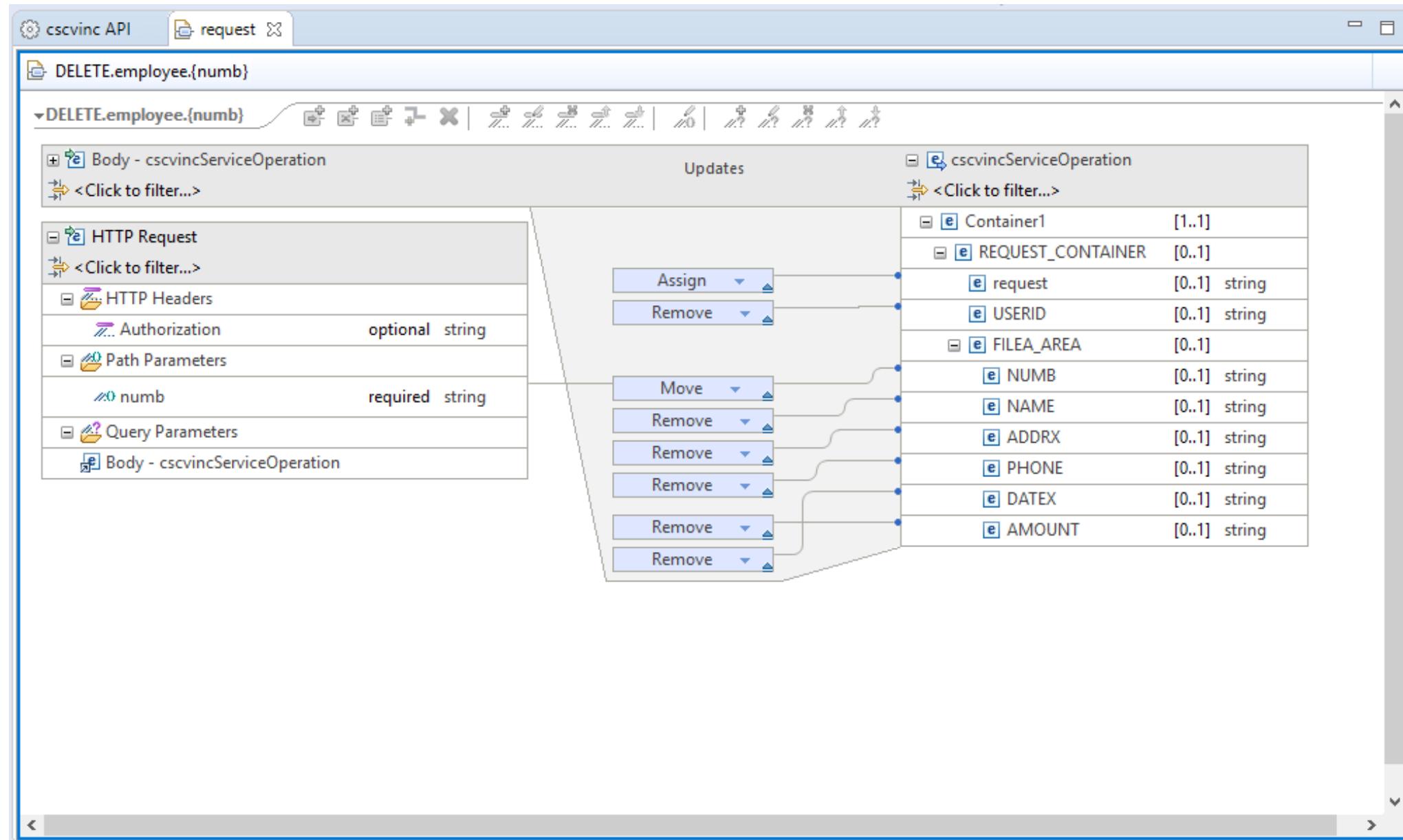
Assign static values to fields, useful for Op codes.

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



API toolkit – API Editor

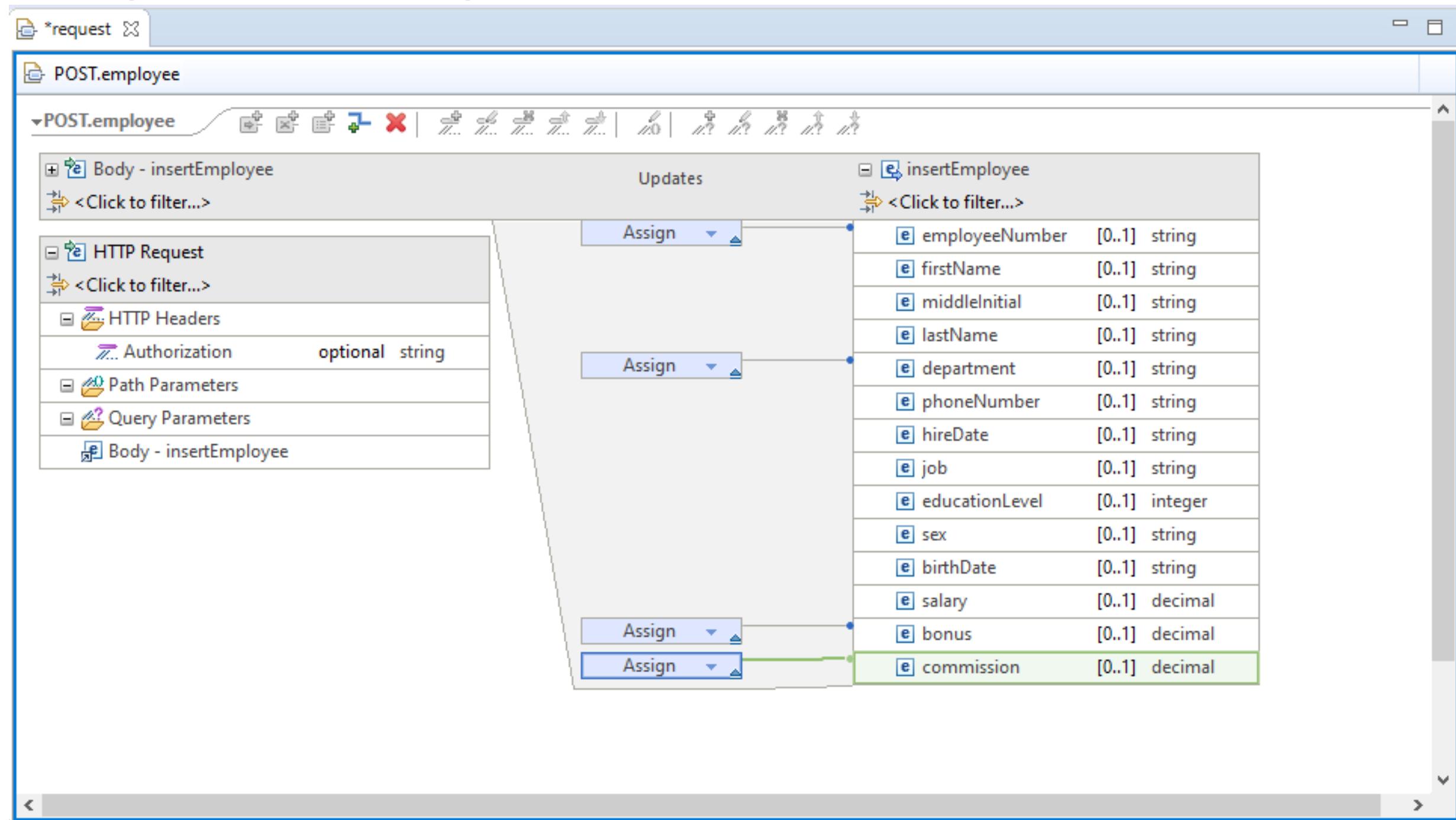
API mapping: Remove or assign values to the fields exposed by service developer





API toolkit – API Editor and Db2 REST service

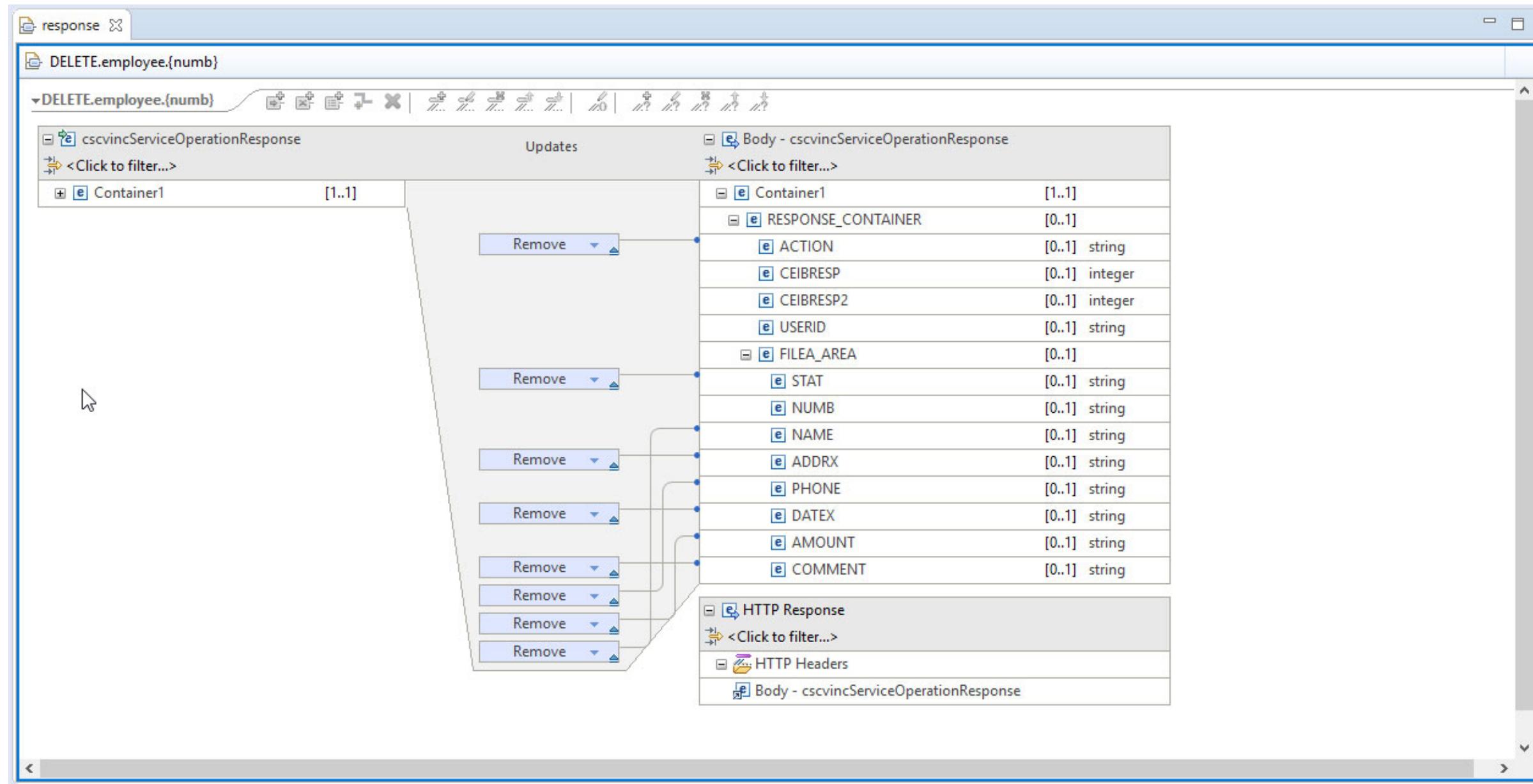
API mapping: Remove/Assign values columns exposed in Db2 REST service





API toolkit – API Editor

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





API toolkit – API Editor

API mapping: Allows adding HTTP header properties

The screenshot shows the API Editor interface with two main sections: 'POST.queue' on the left and 'MQPUTOperation' on the right.

POST.queue Section:

- Contains a 'Body - MQPUTOperation' node.
- Contains an 'HTTP Request' node, which has an 'HTTP Headers' section circled in red. This section contains:
 - 'Authorization' (optional string)
 - 'ibm-mq-md-correlID' (optional string)
- Contains a 'Path Parameters' section.
- Contains a 'Query Parameters' section.
- Contains a 'Body - MQPUTOperation' node.

MQPUTOperation Section:

- Contains an 'MQPUTOperation' node, which has a 'Transferred' section.
- Contains an 'mqmessage' node with attributes:
 - 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 and their mappings. On the left, the API path `/employee/{employee}` is defined with four methods: GET, POST, DELETE, and PUT. The GET method is selected, showing its configuration details. The 'Responses' section for this method lists two entries: '404 Not Found' and '200 OK'. A modal window is open for the 'Edit Response 404' configuration, where the response code is set to '404 - Not Found' and the description is 'Not Found'. Below this, rules are defined to apply this response code based on specific conditions. The rule definitions are as follows:

- Rule 1: Condition: `:e/Container1/RESPONSE_CONTAINER/CEIBRESP` = 13, Action: AND
- Rule 2: Condition: `:/Container1/RESPONSE_CONTAINER/CEIBRESP2` = 80, Action: AND

The summary of the rules is '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.

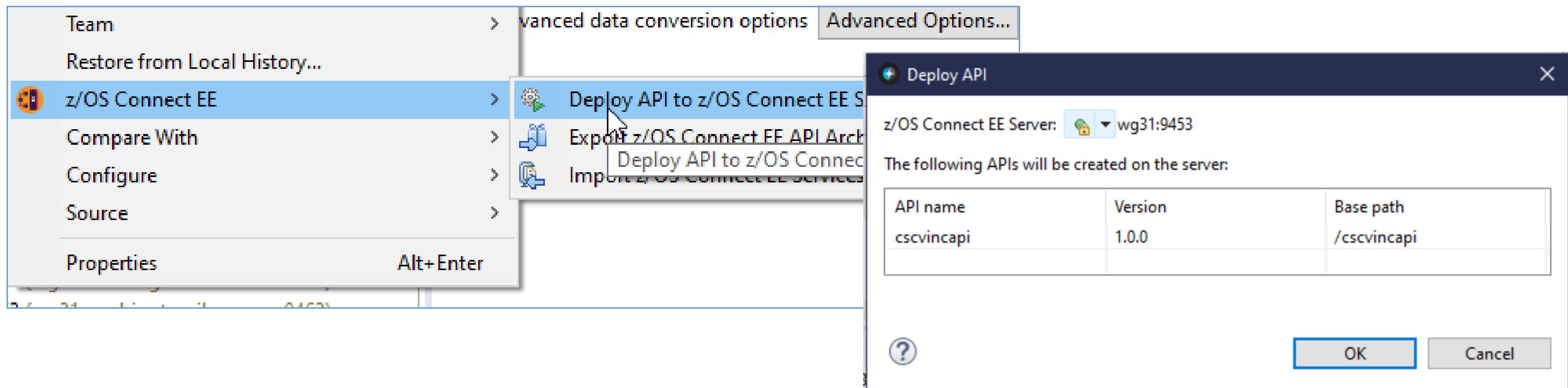
You can define rules based on fields in the service's return interface to tell z/OS Connect EE which response code to return



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 a server named 'wg31:9443 (wg31.washington.ibm.com:9443)' and an 'APIs (1)' node under it, which contains a 'cscvinc' service. A context menu is open over the 'cscvinc' service, with the 'Open In Swagger UI' option selected. This has triggered the opening of two separate browser windows. The left window displays the Swagger UI for the 'cscvinc' service, listing four API endpoints: POST /employee, DELETE /employee/{employee}, GET /employee/{employee}, and PUT /employee/{employee}. The right window shows a detailed view of the GET /employee/{employee} endpoint. It includes the URL 'localhost:55221?url=/api-docs/wg31.washington.ib...', the HTTP method 'GET', the path '/employee/{employee}', and the response class 'Response Class (Status 200) OK'. Below this, the 'Model' tab is active, showing a JSON schema for 'cscvincSelectServiceOperationResponse':

```
{  
  "cscvincContainer": {  
    "response": {  
      "CEIBRESP": 0,  
      "CEIBRESP2": 0,  
      "USERID": "string",  
      "filea": {  
        "employeeNumber": "string",  
        "name": "string"  
      }  
    }  
  }  
}
```

The 'Parameters' section shows two parameters: 'Authorization' (header, string) and 'employee' (path, string). The 'Response Messages' section lists a single entry for '404 Not Found'.



/zosConnect/designer

Create a Web Archive file for OpenAPI 3 APIs

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

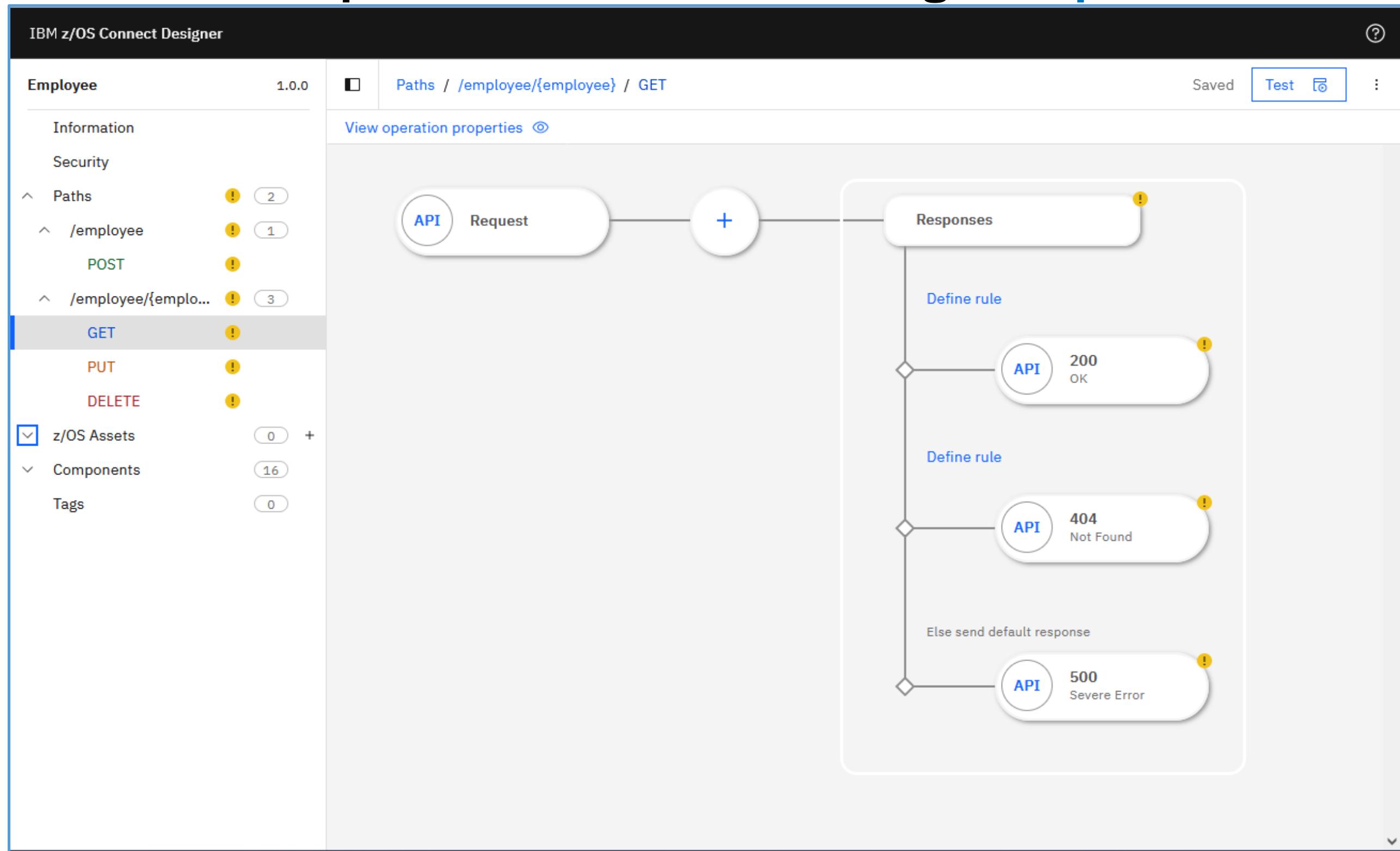


```
cscvinc.yaml - Notepad
File Edit Format View Help
openapi: 3.0.0
info:
  description: "CICS Filea Sample VSAM Application"
  version: 1.0.0
  title: cscvinc
x-ibm-zcon-roles-allowed:
- Manager
paths:
  /employee:
    post:
      tags:
        - cscvinc
      operationId: postCscvincInsertService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

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



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





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

IBM z/OS Connect Designer

cscvinc 1.0.0

Step 2 of 5

Add z/OS Asset

Select a z/OS Asset type

CICS channel program

CICS program name

CSCVINC

Program language

COBOL

CCSID

037

Select a CICS connection

cicsConn

Optional configuration

Transaction ID (optional)

Input Transaction ID

Transaction ID usage (optional)

Select usage

Previous

Next

The screenshot shows the 'Add z/OS Asset' step in the IBM z/OS Connect Designer. The left sidebar lists assets like 'Paths' (2), 'z/OS Assets' (1, currently selected), 'Components' (8), and 'Tags' (0). The main panel is titled 'Add z/OS Asset' and shows the configuration for a 'CICS channel program'. It includes fields for 'CICS program name' (set to 'CSCVINC'), 'Program language' (set to 'COBOL'), 'CCSID' (set to '037'), and a dropdown for 'Select a CICS connection' (set to 'cicsConn'). Below this, there's an 'Optional configuration' section with fields for 'Input Transaction ID' and 'Select usage'. Navigation buttons 'Previous' and 'Next' are at the bottom.



Accessing a z/OS CICS program (OpenAPI 3)

IBM z/OS Connect Designer

cscvinc 1.0.0 z/OS Assets / programCscvinc Saved Test :

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

General
Name: programCscvinc
Type: cicsChannel-1.0
Description: -

z/OS Asset options :

CICS channel program
CICS program name: CSCVINC
Program language: COBOL
CCSID: 037
Connection reference: cicsConn

Request channel
cscvincContainer BIT container

Response channel
cscvincContainer BIT container

The screenshot shows the IBM z/OS Connect Designer interface. On the left, there's a navigation sidebar with sections like Information, Security, Paths (2), z/OS Assets (1+), Components (8), and Tags (0). The z/OS Assets section is expanded, showing 'programCscvinc' selected. The main panel displays the 'General' settings for this asset, including the name 'programCscvinc' and type 'cicsChannel-1.0'. Below the general settings, there's a 'z/OS Asset options' section. Under 'CICS channel program', it lists the CICS program name as 'CSCVINC' and the program language as 'COBOL'. It also specifies CCSID '037' and a connection reference 'cicsConn'. There are two sections for 'Request channel' and 'Response channel', both containing a single entry labeled 'cscvincContainer' under the 'BIT container' category.



Map the method and request message with the CICS program (OpenAPI 3)

IBM z/OS Connect Designer

cscvinc 1.0.0 Paths / /cscvinc/employee/{employee} / GET Saved Test

View operation properties

Request → CICS programCscvinc → Responses

If channel.cscvincContainer.Response-Contai... → API 200 OK

Else send default response

programCscvinc

Request Response z/OS Asset details

Edit mapping View structure

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

Channel

cscvincContainer

Request-Container

- ACTION abc S
- USERID abc
- FILEA-AREA

 - STAT abc
 - NUMB abc API employee

mitchj@us.ibm.com



Map the 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)

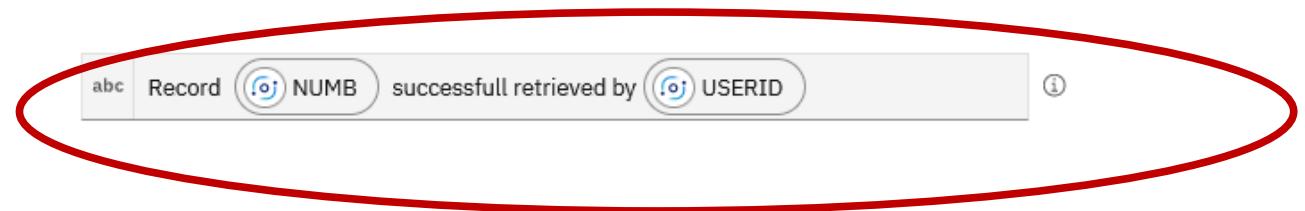
abc (ADDRX)

abc phone

abc (DATEX)

abc (AMOUNT)

abc (COMMENT)



mitchj@us.ibm.com



Add the z/OS asset, a Db2 REST service (OpenAPI 3)

IBM z/OS Connect Designer

EmployeesApi 1.1

Information

Security

Paths 2

/employees/{id} 3

GET

PUT

DELETE

/employees 2

GET

POST

z/OS Assets 6 +

addEmployee

deleteEmployee

getEmployee

getEmployees

selectByRole

updateEmployee

Components 16

Tags 0

View

Step 3 of 4

Add z/OS Asset

Select a Db2 connection

db2Conn

Import from Db2 service manager

Db2 native REST service collection ID
e.g. SYSIBMSERVICE

Db2 native REST service name
e.g. myService

Db2 native REST service version (optional)
e.g. V1

Import Db2 native REST service request schema

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

Specify a URL
http://github.com/example/api-docs Import file

Import Db2 native REST service response schema

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

Previous Next



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

Add z/OS Asset / Import Db2 native REST service

Import Db2 native REST service

Select a Db2 connection

db2Conn

Filter table

12 Db2 native REST service found

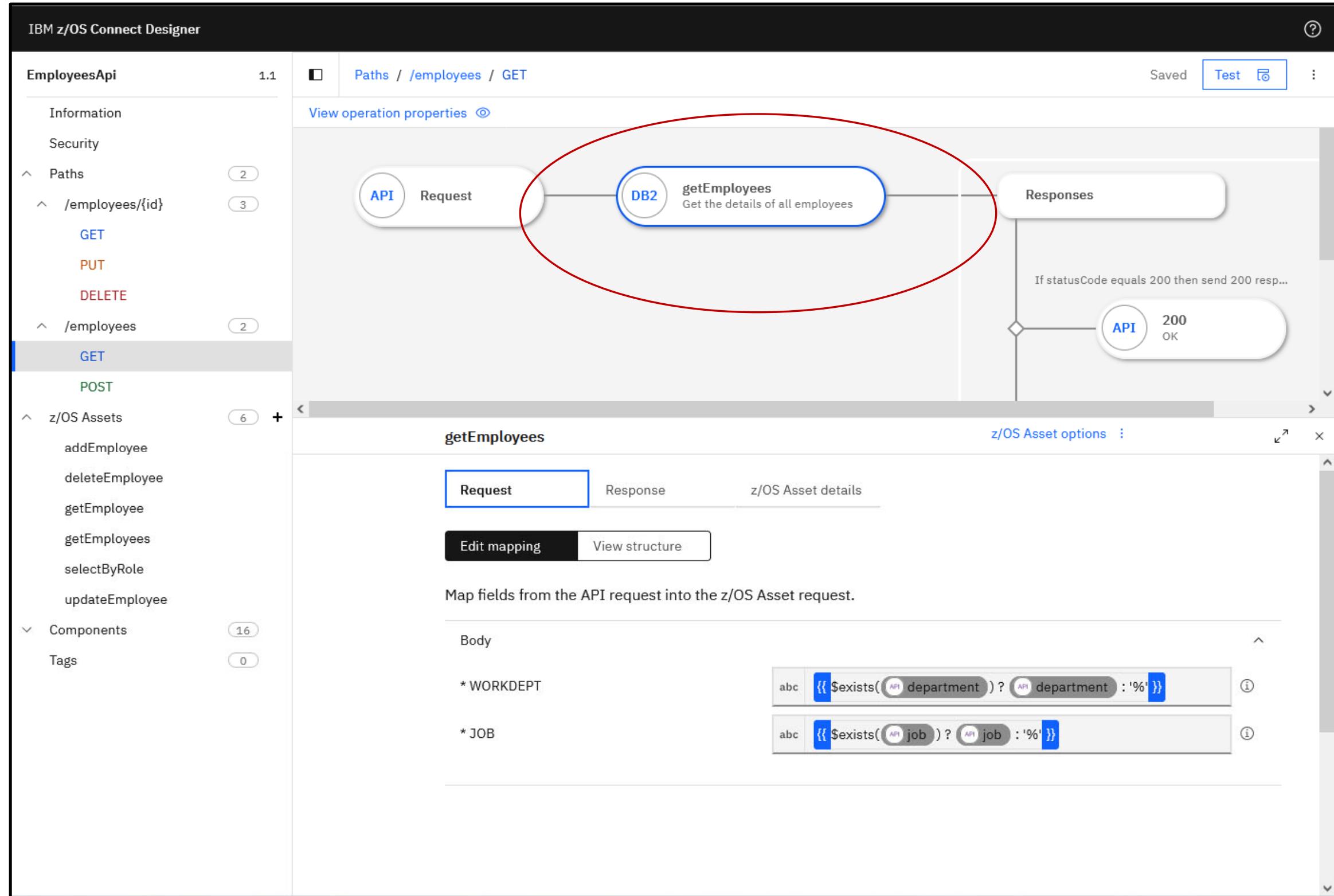
Service name	Version	Collection ID	Path	Description	Status
addEmployee	V1	SYSIBMSERVICE	/services/SYSIBMSERVI...	Add the details of an ind...	Available
deleteEmployee	V1	SYSIBMSERVICE	/services/SYSIBMSERVI...	Remove the details of a...	Available
getEmployee	V1	SYSIBMSERVICE	/services/SYSIBMSERVI...	Get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

Items per page: 10 | 1-10 of 12 items

1 of 2 pages

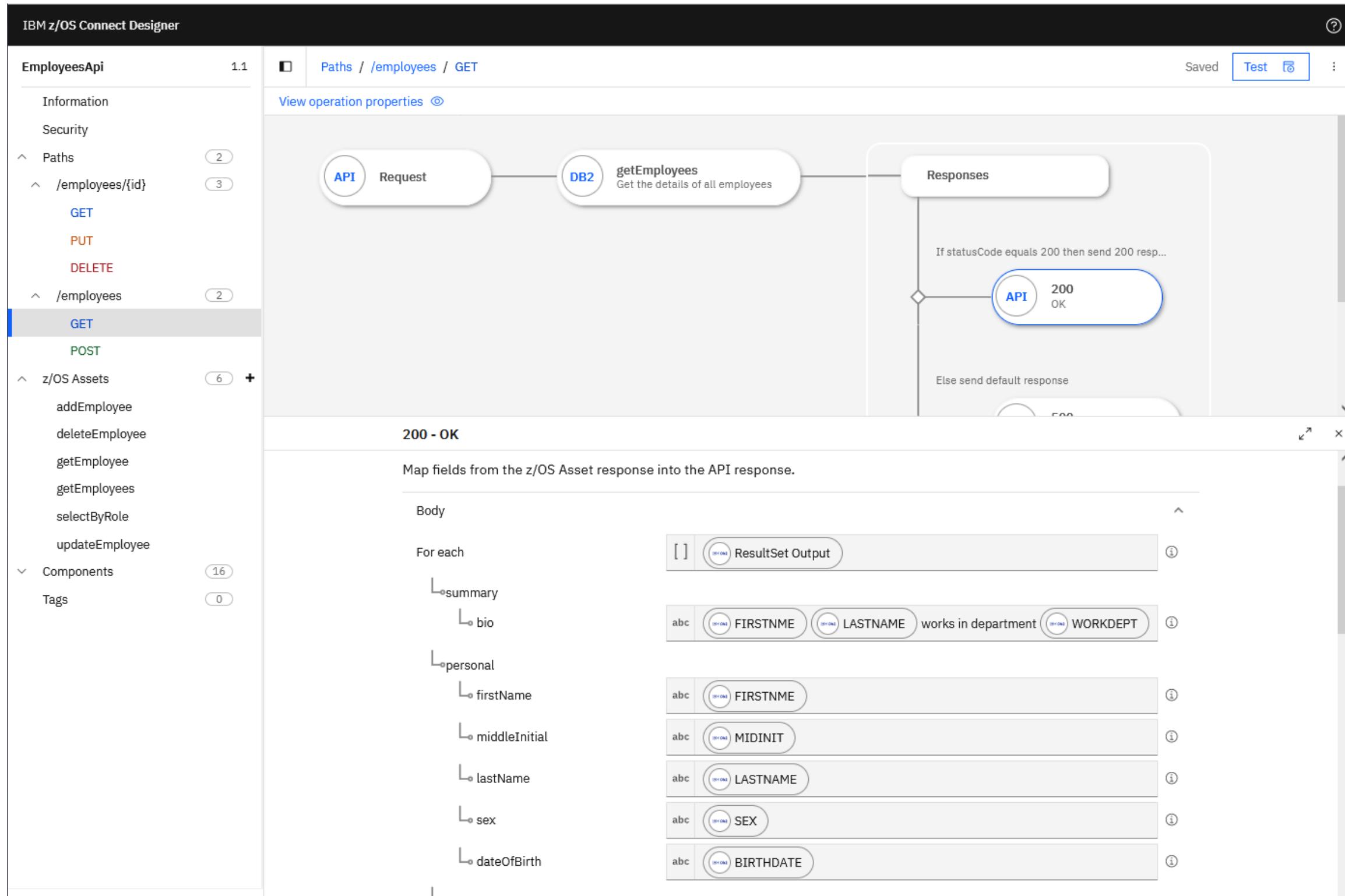
Previous Import

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



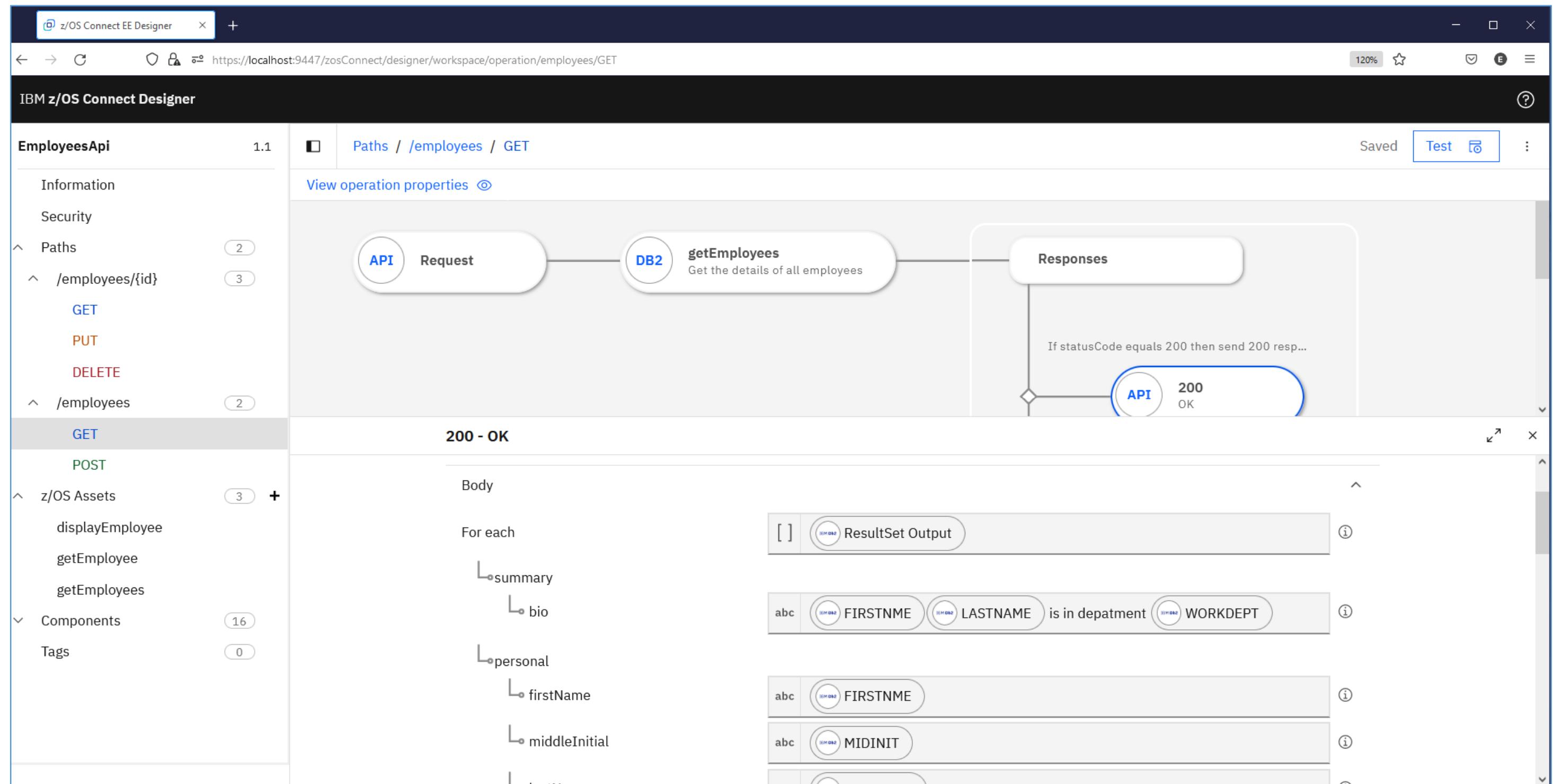


Map the response message (OpenAPI 3)





z/OS Connect Designer for OpenAPI 3 (200)





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 2 /employees/{id} 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.

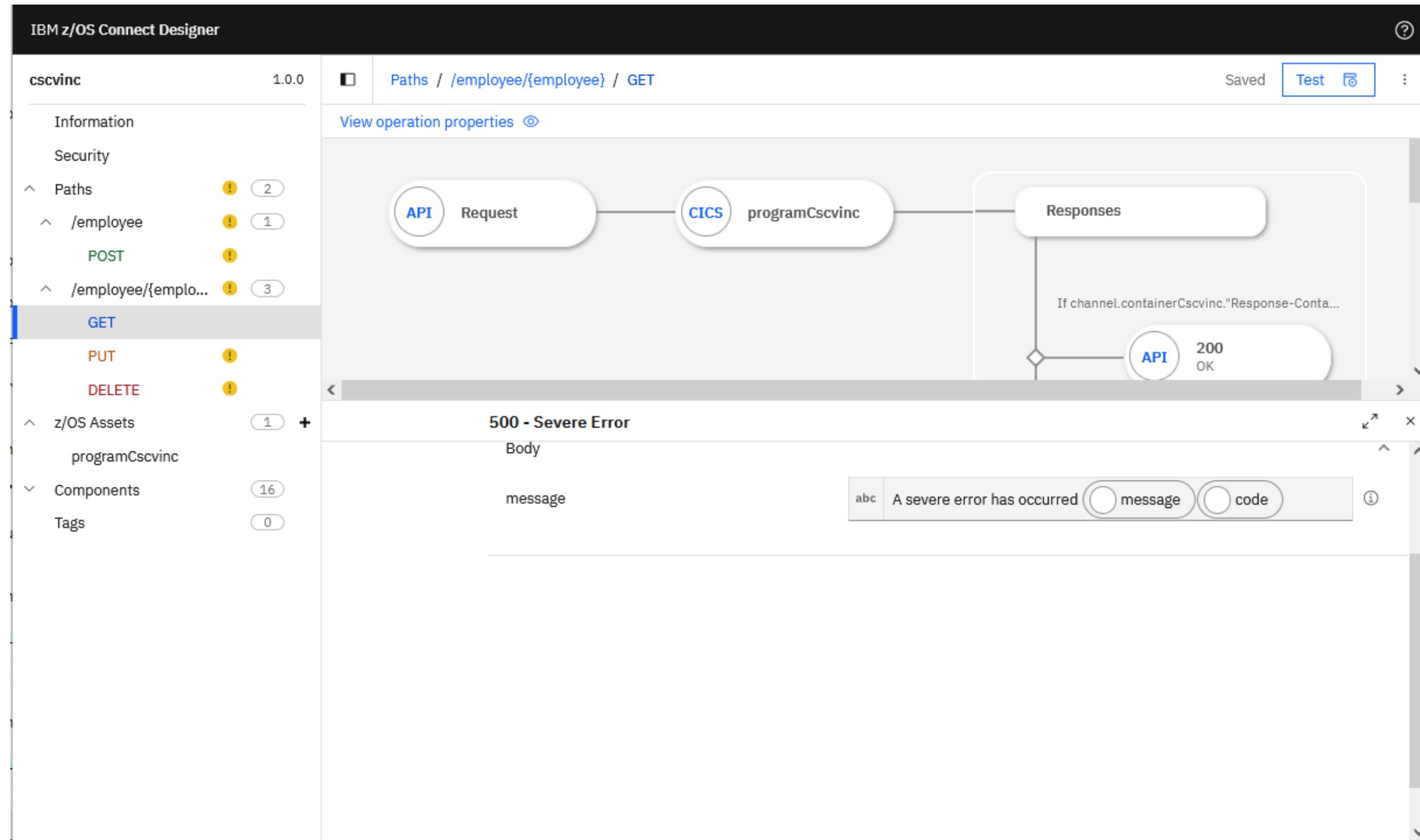
Body

message abc Employee not found

The screenshot shows the IBM z/OS Connect Designer interface for creating an OpenAPI 3 specification. On the left, the 'EmployeesApi' definition is shown with version 1.1. The 'PUT' method is selected under the '/employees/{id}' path. The 'Edit mapping' button is highlighted. A response flowchart is displayed on the right, branching from a decision point based on 'Update Count'. If it's 0, a 404 response is sent; otherwise, a 200 response is sent. A note indicates that if 'Update Count' is 0, a 404 response is sent. Below the flowchart, a '404 - Not Found' section is shown with an 'Edit mapping' button and a 'View structure' button. A note says to map fields from the z/OS Asset response into the API response. Under the 'Body' section, there is a 'message' field containing the value 'Employee not found' with an associated icon.

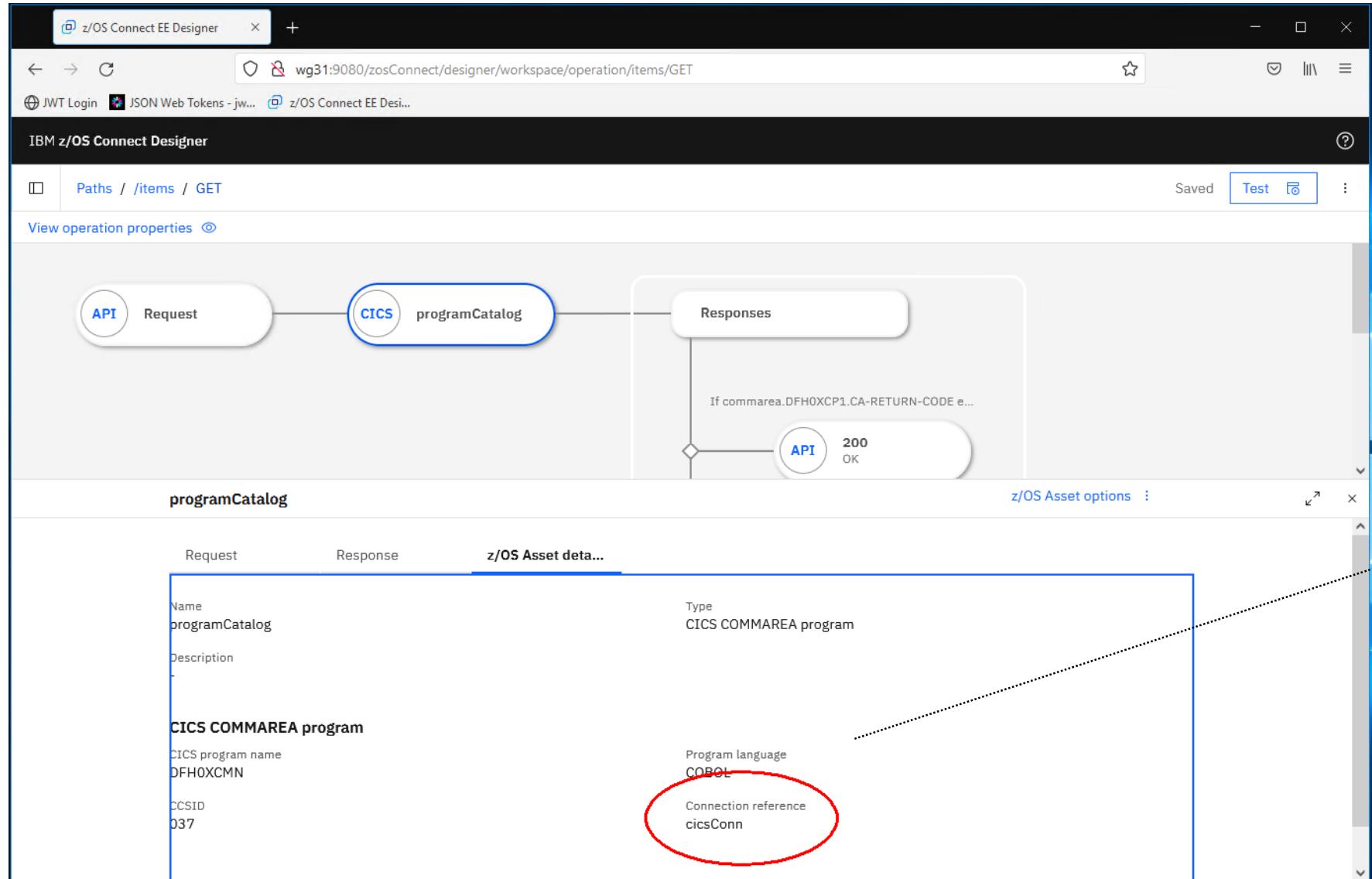


z/OS Connect Designer for OpenAPI 3 (500)





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



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

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

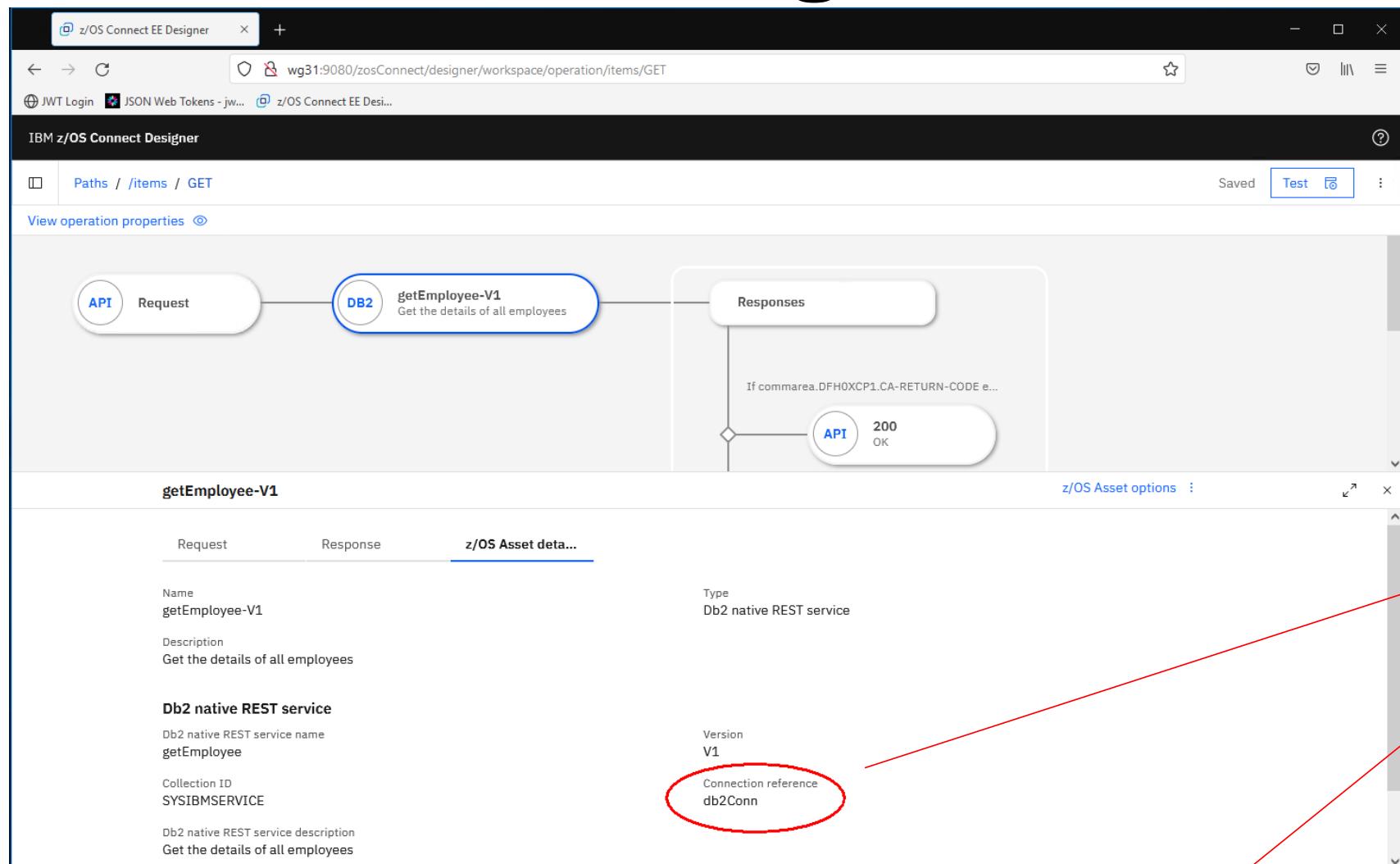
A callout box points to the connection reference in the asset view, explaining its purpose:

Define IPIC connection to CICS using variables defined in bootstrap.properties file

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



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



The screenshot shows the 'Server Config' interface with the 'db2.xml' configuration file open. The 'Design' tab is selected, showing the XML code:

```

<?xml version="1.0" encoding="UTF-8"?>
<server description="Db2 Connections">
  <featureManager>
    <feature>zosconnect:db2-1.0</feature>
  </featureManager>
  <zosconnect_credential user="${DB2_USERNAME}" password="${DB2_PASSWORD}" id="commonCredentials" />
  <zosconnect_db2Connection id="db2Conn" host="${DB2_HOST}" port="${DB2_PORT}" credentialRef="commonCredentials" />
</server>

```

A callout box with a red arrow points to the 'zosconnect_db2Connection' element, containing the text: 'Define connections to Db2 using variables defined in bootstrap.properties file'.

```

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

```

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



EJB roles for z/OS Connect (OpenAPI 3)

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

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

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

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

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

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

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

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



z/OS Server Issues and Considerations – API Deployment

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

Gradle plug-in and its requirements, see [Using Gradle 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"/>
```



What REST test tooling is available?



API Testing 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. To the right of the search bar are icons for cloud, plus, settings, and notifications, along with an Upgrade button. Below the navigation is a list of environments: 'No Environment' is selected. In the main workspace, there are two requests listed: one for 'https://wg31.washington.ibm.com:9453/cscvinc/employee/948478' and another for 'https://mpz3.washington.ibm.com:9443/cscvinc/employee/111111'. The second request is currently selected and has a 'Send' button highlighted. Below the requests, the 'Params' tab is active, showing 'Query Params' with a table:

KEY	VALUE	DESCRIPTION	...	Bulk Edit
...

Below the table, there are tabs for Body, Cookies (1), Headers (8), and Test Results. The 'Body' tab is selected, displaying a JSON response with line numbers:

```
1 {  
2   "cscvincSelectServiceOperationResponse": {  
3     "cscvincContainer": {  
4       "response": {  
5         "CEIBRESP": 0,  
6         "CEIBRESP2": 0,  
7         "USERID": "CICSUSER",  
8         "filea": {  
9           "employeeNumber": "111111",  
10          "name": "C. BAKER",  
11          "address": "OTTAWA, ONTARIO",  
12          "phoneNumber": "51212003",  
13          "date": "26 11 81",  
14          "amount": "$00011.00"  
15        }  
16      }  
17    }  
18  }  
19 }
```

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', 'Trash', and a settings icon.

mitchj@us.ibm.com

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



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

The screenshot shows the IBM REST API Documentation interface. On the left, there's a sidebar with categories like 'Liberty REST APIs', 'cscvinc', 'db2employee', 'filemgr', 'imsPhoneBook', 'jwtIvpDemoApi', 'miniloancics', 'mqapi', and 'phonebook'. The 'cscvinc' section is expanded, showing operations: POST /cscvinc/employee, DELETE /cscvinc/employee/{employee}, GET /cscvinc/employee/{employee}, and PUT /cscvinc/employee/{employee}. The 'db2employee' section is collapsed. The 'filemgr', 'imsPhoneBook', 'jwtIvpDemoApi', 'miniloancics', 'mqapi', and 'phonebook' sections are also collapsed.

On the right, a browser window titled 'IBM REST API Documentation' is open at the URL <https://mpz3.washington.ibm.com:9443/api/explorer/#!/cscvinc/employee/111111>. The browser interface includes a 'Curl' section with a command line, a 'Request URL' field, a 'Response Body' pane showing a JSON object, a 'Response Code' field with '200', and a 'Response Headers' pane.

```
curl -X GET --header 'Accept: application/json' --header 'Authorization: Basic RnJlZDpmcmVk' 'https://mpz3.washington.ibm.com:9443/cscvinc/employee/111111'
```

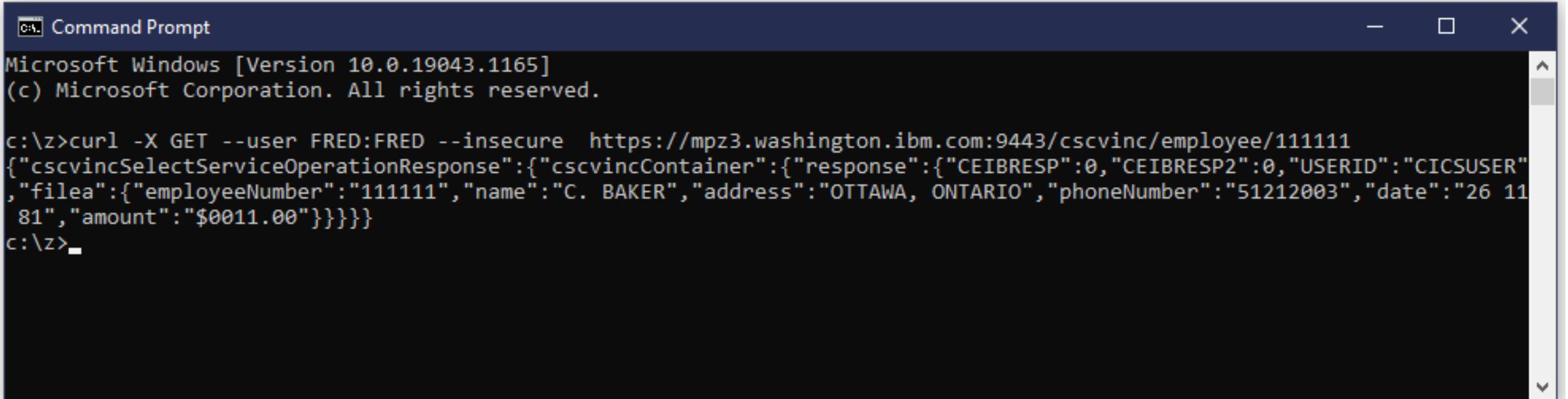
```
{
  "cscvincSelectServiceOperationResponse": {
    "cscvincContainer": {
      "response": {
        "CEIBRESP": 0,
        "CEIBRESP2": 0,
        "USERID": "CICSUSER",
        "filea": {
          "employeeNumber": "111111",
          "name": "C. BAKER",
          "address": "OTTAWA, ONTARIO",
          "phoneNumber": "51212003",
          "date": "26 11 81",
          "amount": "$0011.00"
        }
      }
    }
  }
}
```

```
200
```

```
{
  "content-language": "en-US",
  "content-length": "269",
  "content-type": "application/json; charset=UTF-8",
}
```



API Testing with cURL



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>

MVS JCL Invoking curl using Rocket Software's tooling



```
*****  
/* 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=stop+  
ped > 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
```

Change the status of API to stopped

Delete or remove the API from the server

Deploy the API to the server

Execute the API



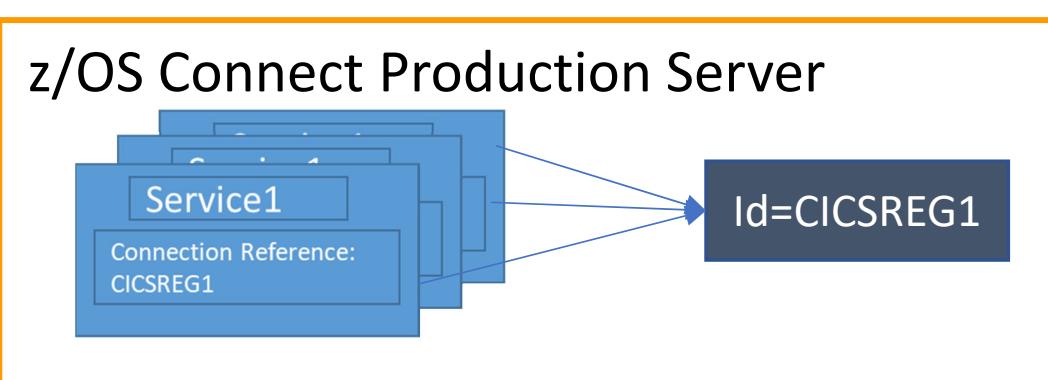
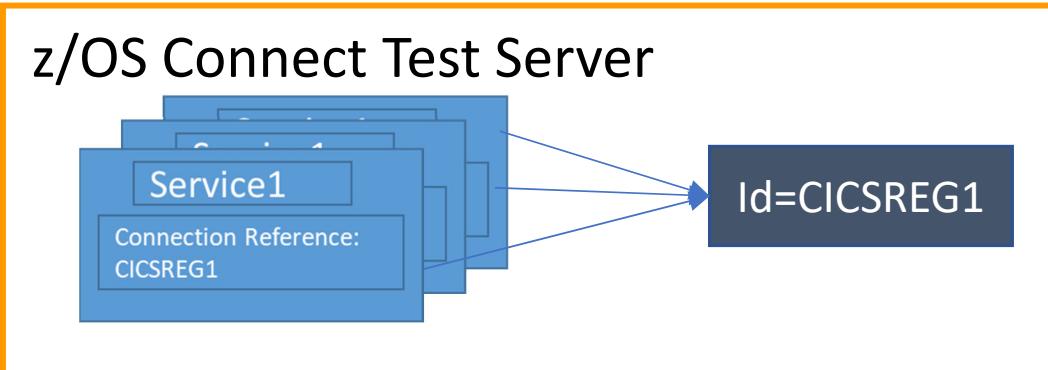
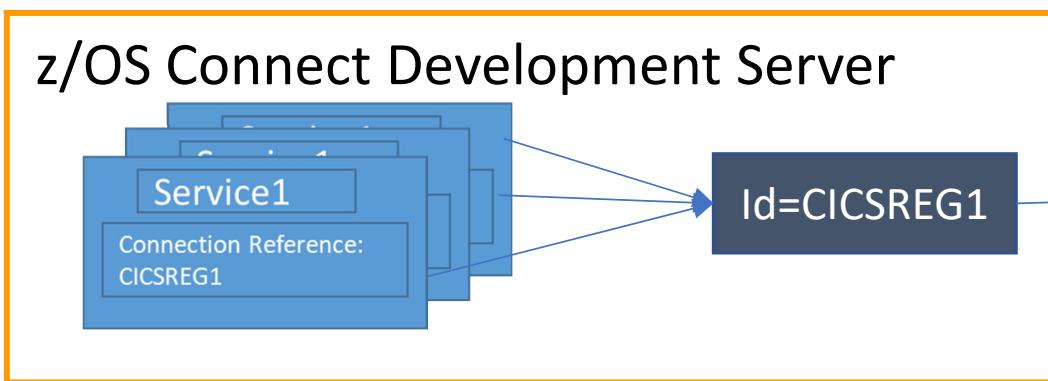
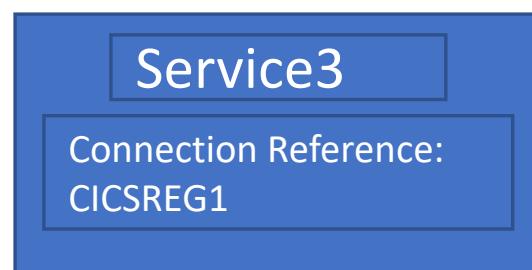
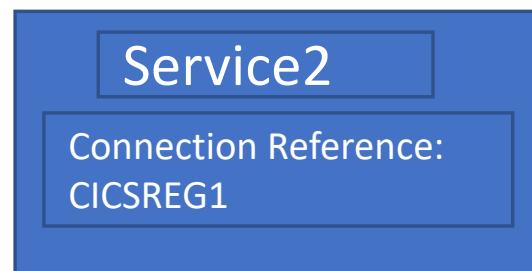
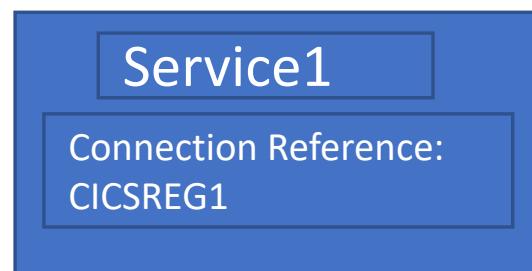
Connection References

Carefully consider the names used for connections



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

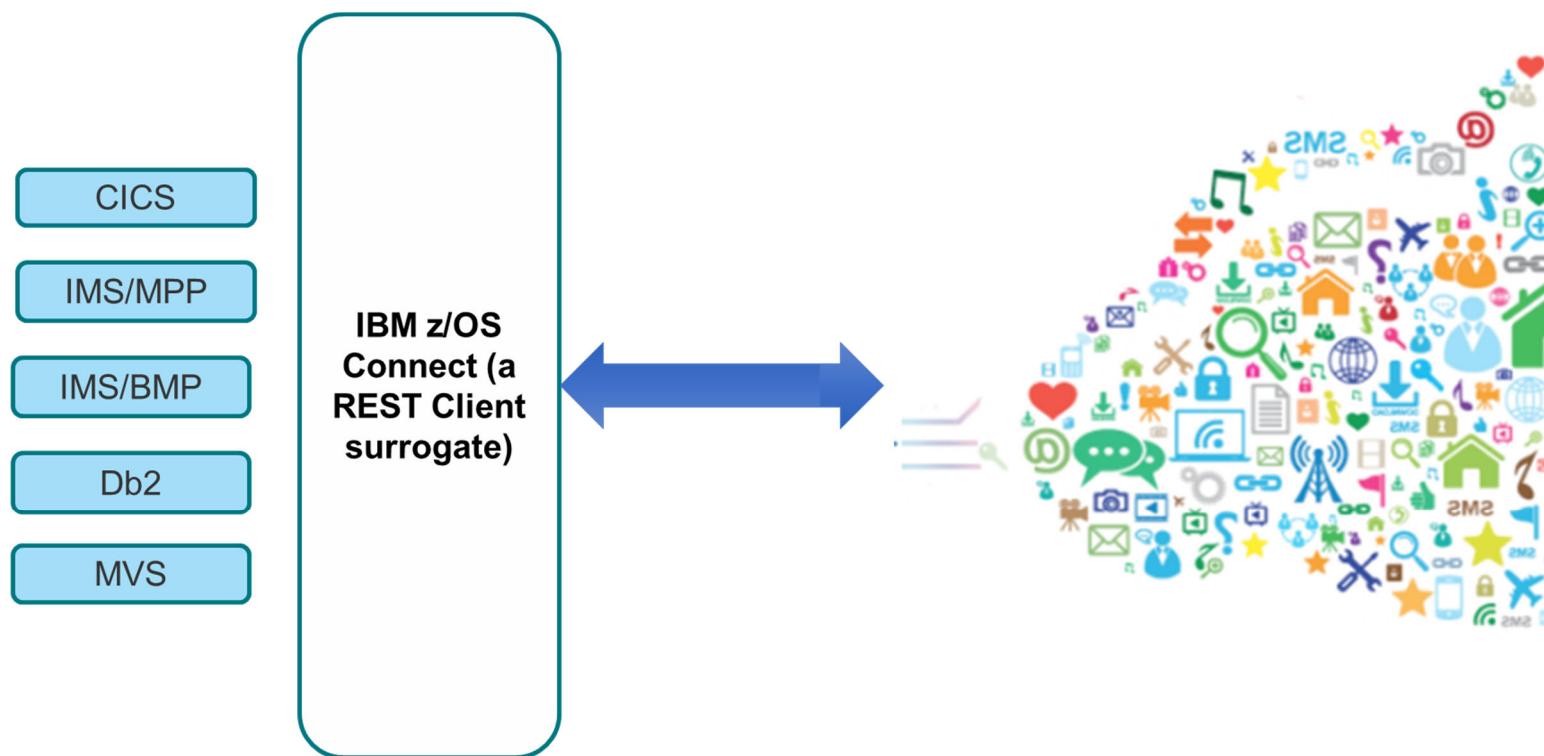
Don't couple service and API requester connection names to specific systems or endpoints





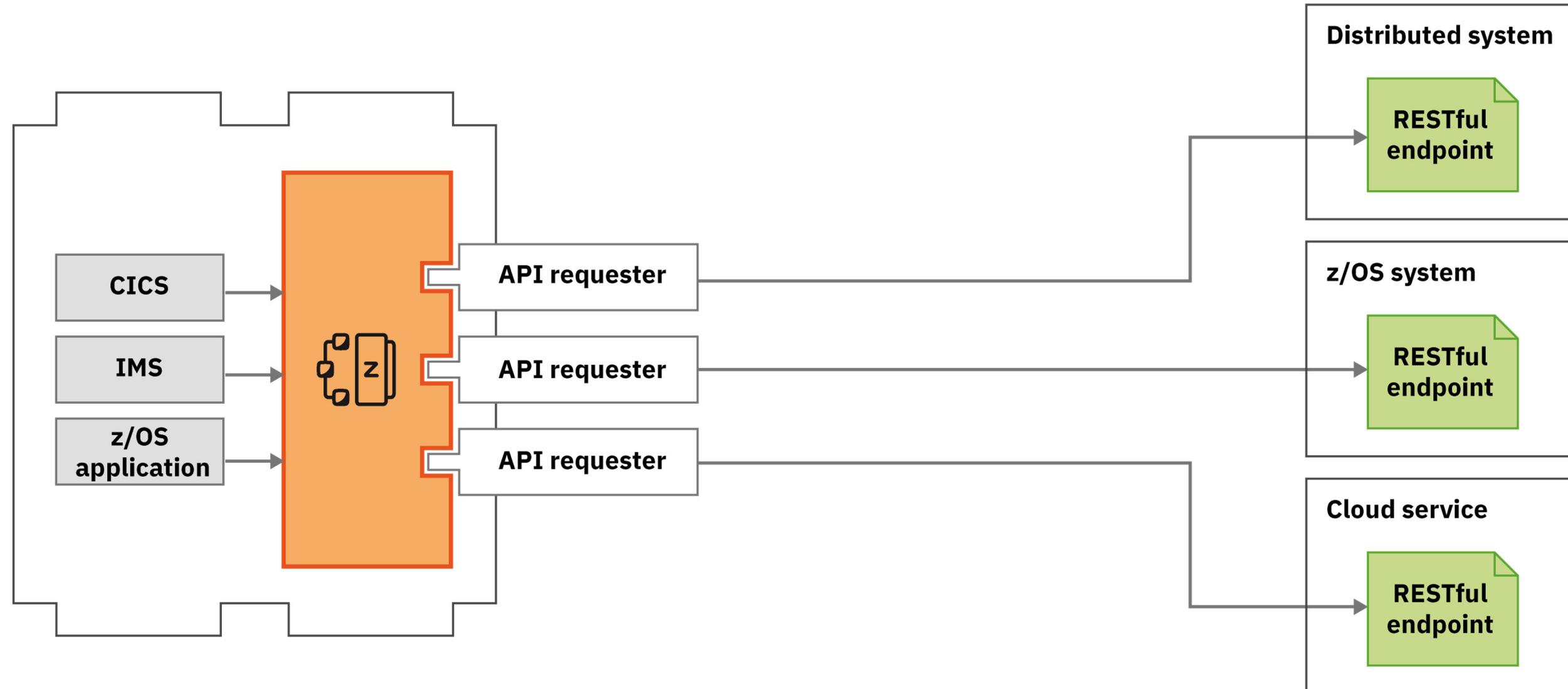
/api_toolkit/apiRequesters

Quick and easy **API mapping**.





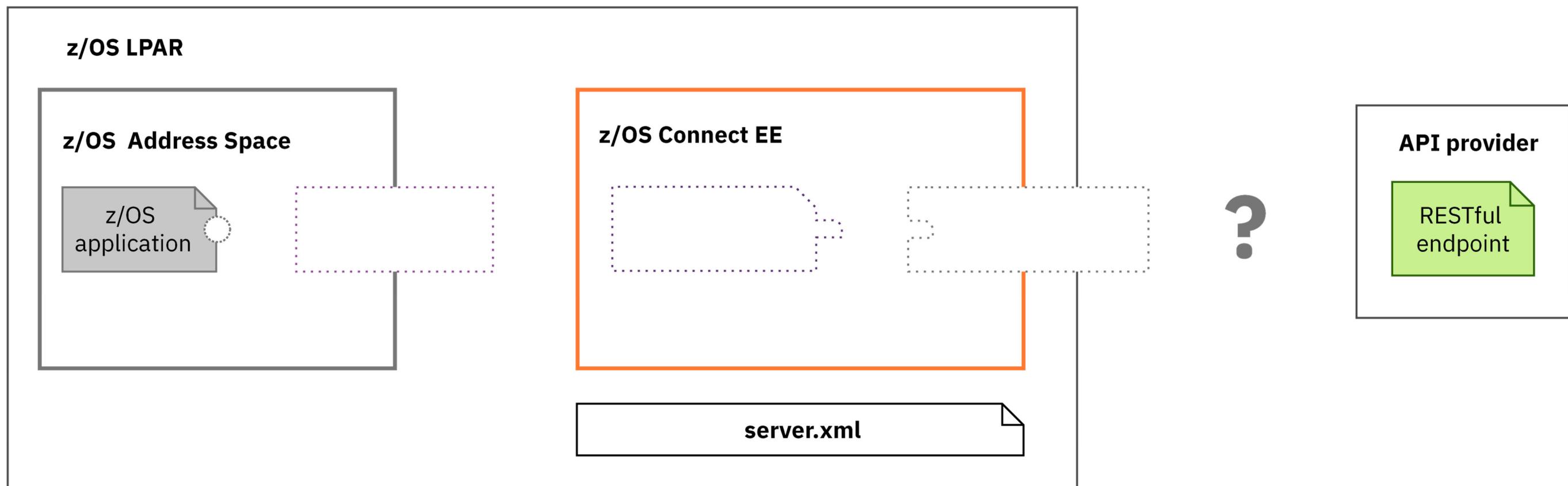
Use API requester to call external APIs from z/OS assets





Steps to calling an external API

Starting point



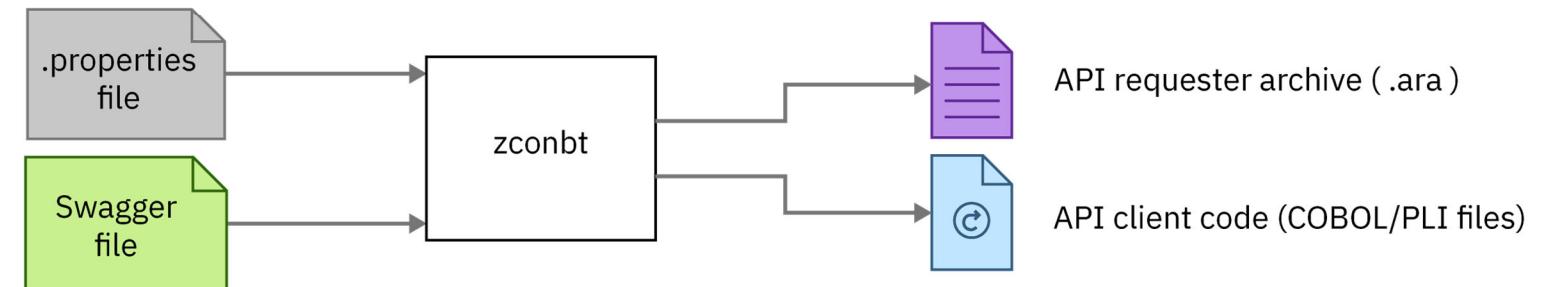


Steps to calling an external API

Generate API requester archive and API client code from Swagger

The screenshot shows a Swagger JSON editor window with the following content:

```
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:
        - "getCscvincSelectService"
      parameters:
        - name: "employee"
          in: "path"
          required: true
          type: "string"
          maxLength: 6
      responses:
        200:
          description: "OK"
        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>



COBOL working storage implications

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

```
/C:/z/apiRequester/ATS/ATSContactX
+-----+
  file:///C:/z/apiRequester/ATS/ATSContactPreferences
  - X
  JSON Raw Data Headers
  Save Copy Collapse All Expand All Filter 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: "122222444001"
          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"
```

```
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 PIC S9999 COMP-5 SYNC.
    15 umi2 PIC X(255).

 12 pin-num PIC S9(9) COMP-5 SYNC.
 12 pin.
    15 pin2-length PIC S9999 COMP-5 SYNC.
    15 pin2 PIC X(255).

 12 firstName-num PIC S9(9) COMP-5 SYNC.
 12 firstName.
    15 firstName2-length PIC S9999 COMP-5 SYNC.
    15 firstName2 PIC X(255).

 12 middleName-num PIC S9(9) COMP-5 SYNC.
 12 middleName.
    15 middleName2-length PIC S9999 COMP-5 SYNC.
    15 middleName2 PIC X(255).

 12 lastName-num PIC S9(9) COMP-5 SYNC.
 12 lastName.
    15 lastName2-length PIC S9999 COMP-5 SYNC.
    15 lastName2 PIC X(255). 
```



Consider adding constraints to the properties

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

The screenshot shows a JSON editor interface with the following JSON structure:

```
{
  "type": "array",
  "items": [
    {
      "type": "array",
      "maxItems": 10,
      "memberCodeableConcept": {
        "description": "Multiple member codes"
      }
    },
    {
      "type": "object"
    }
  ],
  "title": "Member Contacts Request",
  "description": "Read-only request data to search for member contact information."
}
```

Two specific fields have red circles around them:

- `communicationPreferences.items.maxItems: 10`
- `umi.maxLength: 12`

The screenshot shows a Notepad window with the following COBOL-like code:

```
* 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  PIC S9(9) COMP-5
                  15 umi2          PIC X(12).

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

12 pin.           15 pin2-length  PIC S9(9) COMP-5
                  15 pin2          PIC X(255).

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

12 firstName.     15 firstName2-length  PIC S9(9) COMP-5
                  15 firstName2    PIC X(30).

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

12 middleName.    15 middleName2-length  PIC S9(9) COMP-5
                  15 middleName2   PIC X(30).
```

Three specific fields have red circles around them:

- `memberContactsResponse OCCURS 10.`
- `umi2-length` and `umi2`
- `firstName2-length` and `firstName2`

There are also API Requester generation properties available to help

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**.

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	SYNC.

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

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

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



The number of specific entries can be ambiguous

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

```
BROWSE      USER1.ZCEE.SOURCE(PUTAPI) - 01.05      Line 000000062 Col 001 080
Command ===>      Scroll ===> PAGE
      MAINLINE SECTION.

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

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

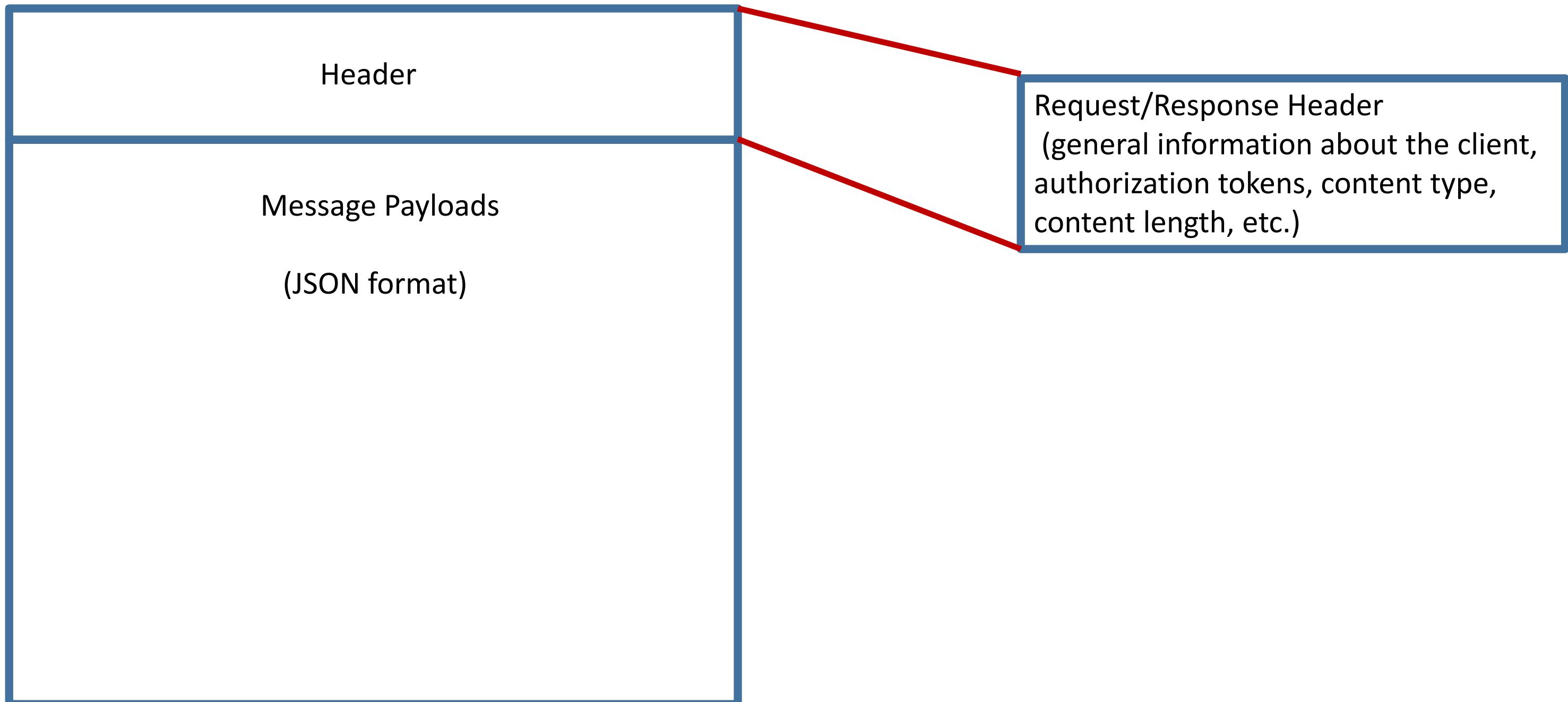
      MOVE numb of PARM-DATA TO employee IN PUT-REQUEST.
      MOVE LENGTH of employee in PUT-REQUEST to
            employee-length IN PUT-REQUEST.

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

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



Request and Response Message Layout





Providing an API key to the request and the application

The application can provide the authentication credentials required by the API.

Via a HTTP header

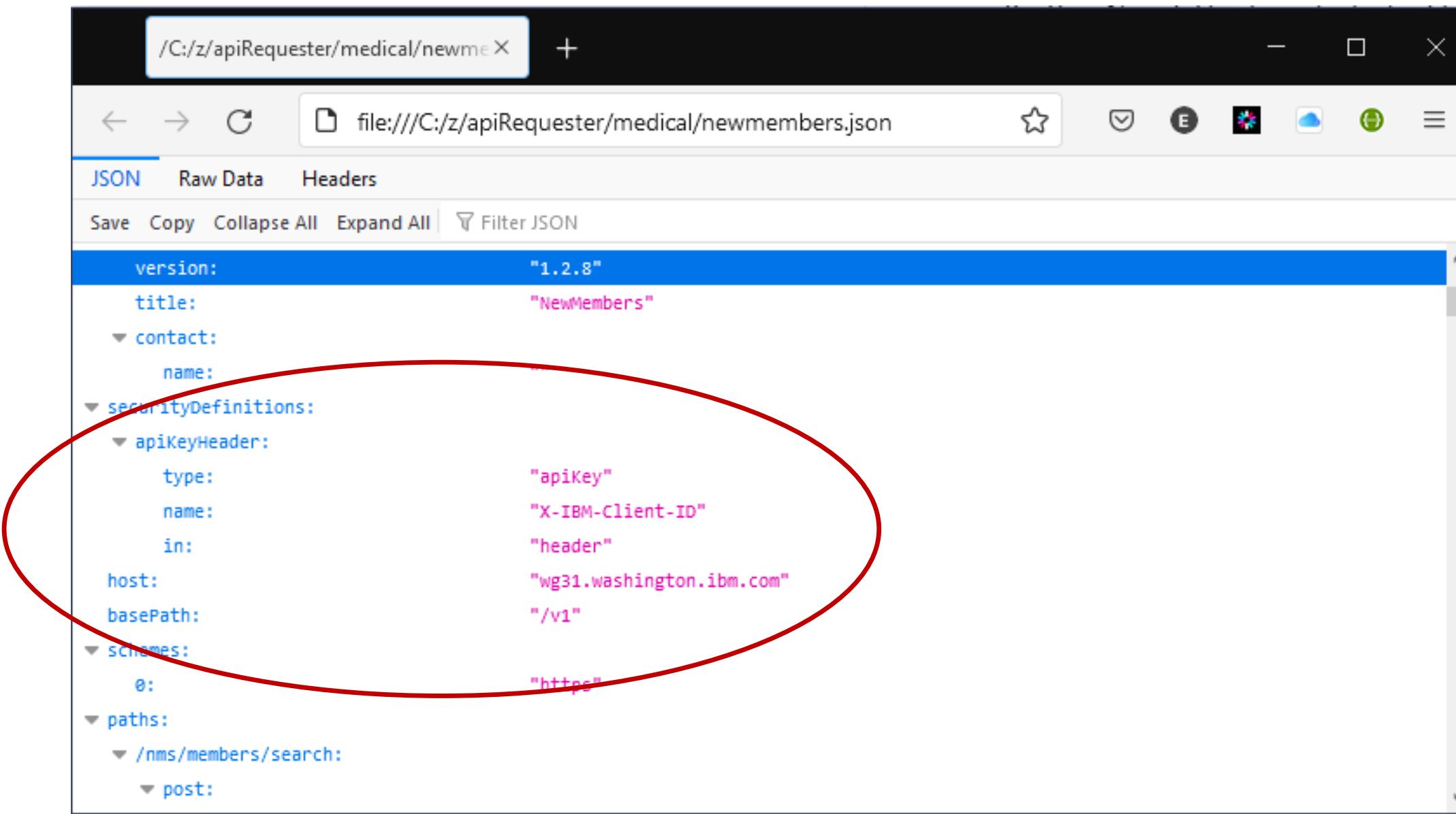
GET /something HTTP/1.1

X-API-Key: abcdef12345

Or via a query parameter

GET /something?api_key=abcdef12345

When provided in the specification document as shown below or . . .



```
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:
```



Or by using 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.

The screenshot shows a Windows Notepad window titled "cscvinc.properties". The content of the file is as follows:

```
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
```

The status bar at the bottom of the Notepad window displays: Ln 8, Col 19 | 100% | Unix (LF) | UTF-8

Support for an application to add an API key to the request



Either way, adds code to the request copy book which can be initialized by the application

The image shows two windows from the mpz3 terminal application. The left window displays assembly language code with several lines circled in red. The right window displays the corresponding source code in P-Code, also with some lines circled in red.

Left Window (Assembly Language):

```
*      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-ClientTransactio-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-ClientSessionI-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.
```

Right Window (Source Code):

```
EDIT      USER1.ZCEE.SOURCE(GETAPIEN) - 01.01
Command ==> Columns 00001 00072
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      * Set up the data for the API Requester call
000092      *
000093      MOVE employee of PARM-DATA TO employee IN GET-REQUEST.
000094      MOVE LENGTH of employee in GET-REQUEST to
000095      employee-length IN GET-REQUEST.
000096
000097      *-----*
000098      * Initialize API Requester PTRs & LENs
000099      *
000100      * Use pointer and length to specify the location of
000101      * request and response segment.
000102      * This procedure is general and necessary.
000103      SET BAQ-REQUEST-PTR TO ADDRESS OF GET-REQUEST.
000104      MOVE LENGTH OF GET-REQUEST TO BAQ-REQUEST-LEN.
000105      SET BAQ-RESPONSE-PTR TO ADDRESS OF GET-RESPONSE.
000106      MOVE LENGTH OF GET-RESPONSE TO BAQ-RESPONSE-LEN.
000107
000108      *-----*
```

Bottom status bar: Connected to remote server/host mpz3 using lu/pool MPZ30019 and port 23



Additional Swagger header properties

The application can also set values for additional header properties required by the API

```
"/C:/z/apiRequester/ATS/decision-service-swagger" +  
- □ X  
← → C file:///C:/z/apiRequester/ATS/decision-service-swagger ★  
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: "ROMEDSV02.0"  
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:
```

```
File Edit Format View Help  
06 ReqHeaders.  
09 x-hmhs-keyId-length  
09 x-hmhs-keyId  
09 ConsumerID-length  
09 ConsumerID  
09 ContextID-length  
09 ContextID  
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  
Ln 130, Col 18 100% Windows (CRLF) UTF-8
```



Steps to calling an external API

Using `zconbt` to generate API requester archive and API client code from 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: getCscvincSelectService, basePath: /cscvincapi, relativePath: /employee/{employee}, method: GET
- request data structure : CSC00Q01
- response data structure : CSC00P01
- api info file : CSC00I01

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

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

operationId: deleteCscvincDeleteService, 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  
//SYSTSPRT DD SYSOUT=*  
//SYSERR DD SYSOUT=*  
//STDOUT DD SYSOUT=*  
//SYSTSIN DD *,SYMBOLS=EXECSYS  
BPXBATCH SH +  
  export WORKDIR=&WORKDIR; +  
  export ZCONDIR=&ZCONDIR; +  
  cd $WORKDIR; +  
  $ZCONDIR/zconbt.zos -p cscvinc.properties -f=cscvinc.ara; +  
  cp -v $WORKDIR/syslib/* "/*'JOHNSON.ZCONBT.COPYLIB'"
```

cscvinc.properties

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

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



Tech-Tip: Copy books naming convention

- 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.

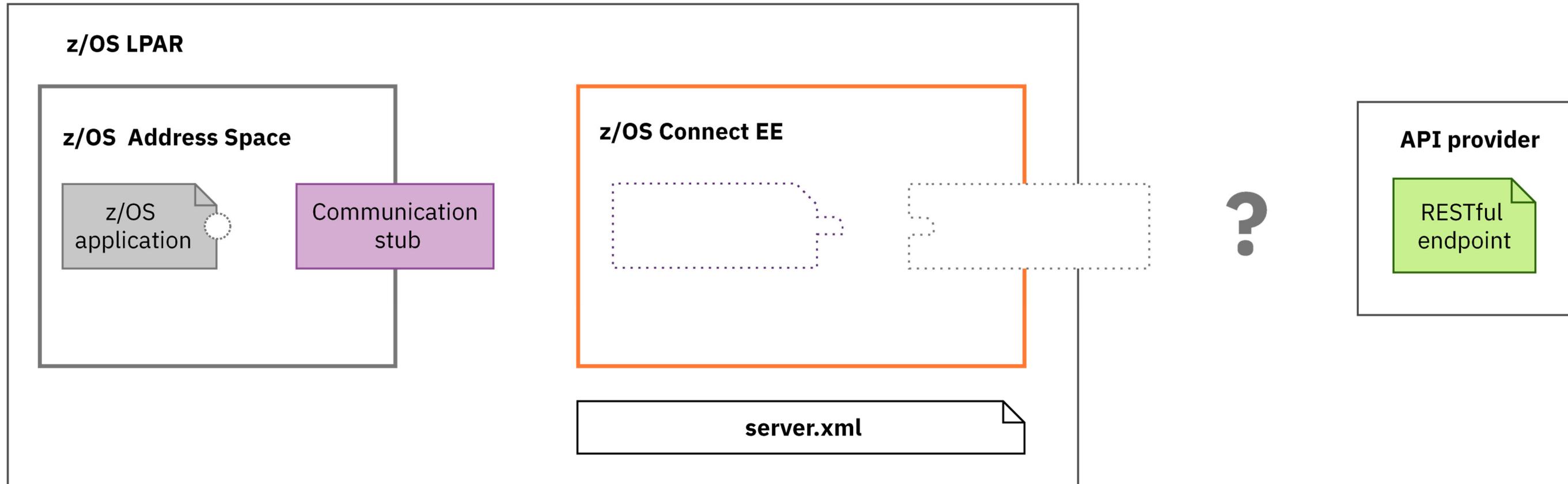
```
* Request and Response
 01 GET-REQUEST.
    10 FILLER
      01 GET-RESPONSE.
        COPY MQ000R01 SUPPRESS.
* Structure with the API information
 01 GET-INFO-OPER1.
    COPY MQ000I01 SUPPRESS.
```

- 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



Steps for involving an external API from a COBOL program

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



Configure a communication stub.

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

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



Steps to calling an external API

Include the generated copy books in a COBOL program

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

  * Copy API Requester required copybook
  COPY BAQRINFO.

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

  01 API_RESPONSE.
    COPY CSC02P01.

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

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

API-REQUEST

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

API-RESPONSE

```
CSC00I01 ✎ CSC00Q01 ✎ CSC00P01 ✎
  *
  * ++++++
  06 ResBody.
  09 cscvincSelectServiceOp-num  PIC S9(9) COMP-5 SYNC.
  09 cscvincSelectServiceOperatio.
    12 Container1.
    15 RESPONSE-CONTAINER2-num  PIC S9(9) COMP-5
  SYNC.
```

API-INFO-OPER1

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



Steps to calling an external API

Add a call to the communication stub passing pointers to working storage of the copy books

The diagram illustrates the steps to calling an external API through a communication stub, showing code snippets from four windows:

- GETAPI**: The main program window containing assembly language code. It highlights:
 - * Set up the data for the API Requester call
 - * Initialize API Requester PTRs & LENs
 - * Call the communication stub
 - * The BAQ-RETURN-CODE field in 'BAQRINFO' indicates whether this
- CSC00I01**: A copy book window showing the definition of the communication stub parameters. It highlights:
 - 03 BAQ-APINAME VALUE 'cscvincapi_1.0.0'. PIC X(255)
 - 03 BAQ-APINAME-LEN PIC S9(9) COMP-5 SYNC
 - 03 BAQ-APIPATH VALUE '%2Fcscvincapi%2Femployee%2F%7Bemployee%7D'.
 - 03 BAQ-APIPATH-LEN PIC S9(9) COMP-5 SYNC
 - 03 BAQ-APIMETHOD VALUE 'GET'. PIC X(255)
 - 03 BAQ-APIMETHOD-LEN PIC S9(9) COMP-5 SYNC
- CSC00Q01**: A copy book window showing the schema definition for the response. It highlights:
 - * JSON schema keyword 'minLength' value: '0'.
 - * JSON schema keyword 'maxLength' value: '6'.
 - * This field contains a varying length array of characters or binary data.
 - 09 employee-length PIC S9999 COMP-5 SYNC.
 - 09 employee PIC X(6).
- CSC00P01**: A copy book window showing the detailed structure of the response body. It highlights:
 - * RespBody.
 - 09 cscvincSelectServiceOp-num PIC S9(9) COMP-5 SYNC.
 - 09 cscvincSelectServiceOperatio. 12 Container1.
 - 15 RESPONSE-CONTAINER2-num PIC S9(9) COMP-5 SYNC.



Steps to calling an external API

Access the results

```
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.
  WHEN BAQ-ERROR-IN-API
```

```
mpz3
File Edit Settings View Communication Actions Window Help
Menu Utilities Compilers Help
BROWSE ZCEE30.SBAQCOB(BAQRINFO)
Command ==>
Line 0000000066 Col 001 080
Scroll ==> PAGE

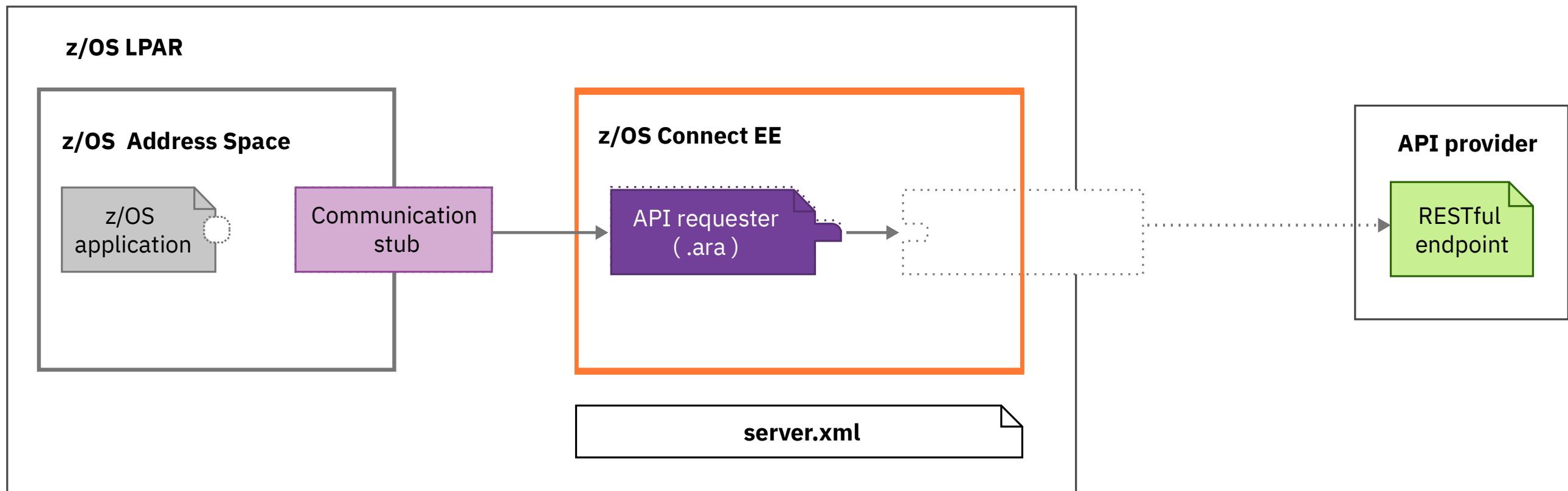
01 BAQ-RESPONSE-INFO,
03 BAQ-RESPONSE-INFO-COMP-LEVEL PIC S9(9) COMP-5 SYNC VALUE 0.
03 BAQ-STUB-NAME PIC X(8).
03 BAQ-RETURN-CODE PIC S9(9) COMP-5 SYNC.
  88 BAQ-SUCCESS          VALUE 0.
  88 BAQ-ERROR-IN-API    VALUE 1.
  88 BAQ-ERROR-IN-ZCEE   VALUE 2.
  88 BAQ-ERROR-IN-STUB  VALUE 3.
  88 BAQ-ERROR-NO-RESPONSE 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 *****

MA B
Connected to remote server/host mpz3 using lu/pool MPZ30021 and port 23
18 / 058
```



Steps to calling an external API

Deploy API requester (.ara) archive



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



Tech-Tip-Deploying API requester archive files

- Use API requester archive as request message and use HTTP POST
- Use URI path /zosConnect/apiRequesters
- Postman or cURL

The screenshot shows the Postman application interface. A POST request is being prepared to the URL <https://wg31.washington.ibm.com:9483/zosConnect/apiRequesters>. The 'Body' tab is selected, and the 'binary' option is chosen. A file named 'filea.ara' is attached. The response status is 201 Created, and the JSON response body is displayed:

```
1 {"  
2   "name": "filea_2.0.0",  
3   "version": "2.0.0",  
4   "description": "",  
5   "status": "Started",  
6   "apiRequesterUrl": "https://wg31.washington.ibm.com:9483/zosConnect/apiRequesters/filea\_2.0.0",  
7   "connection": "fileaAPI"  
8 }
```

Command:

```
curl --data-binary @filea.ara  
--header "Content-Type: application/zip"  
https://mpxm:9453/zosConnect/apiRequesters
```

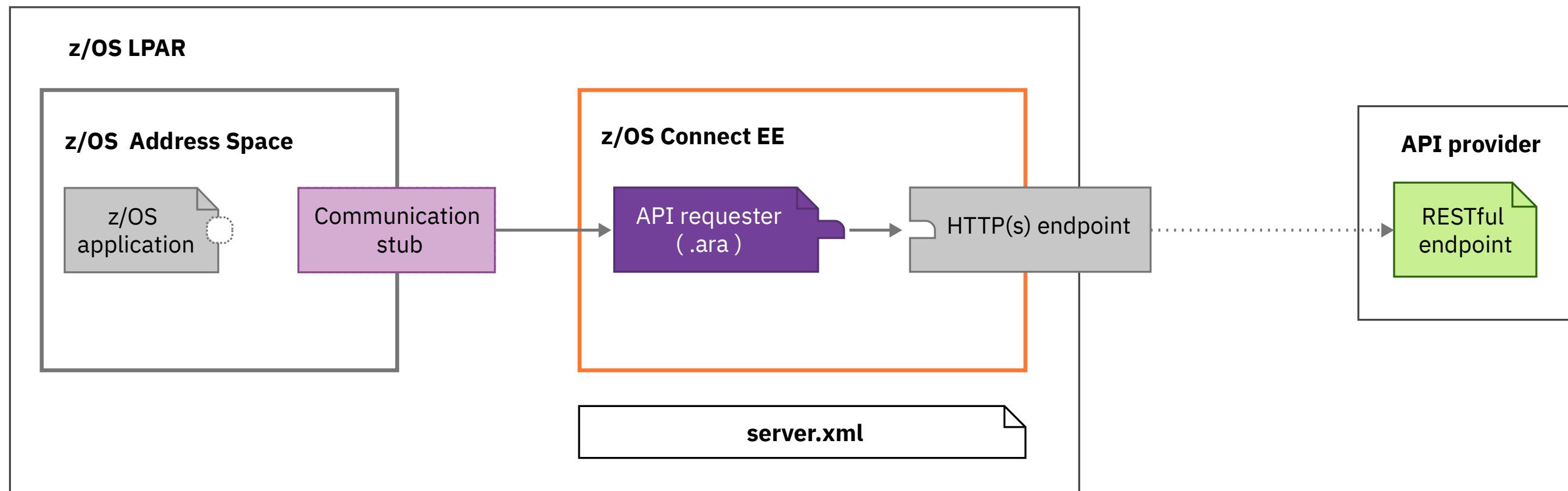
Results:

```
{"name": "filea_2.0.0", "version": "2.0.0", "description": "",  
"status": "Started", "apiRequesterUrl": "https://wg31.washington.ibm.com:9483/zosConnect/apiRequesters/filea_2.0.0", "connection": "fileaAPI"}
```



Steps to calling an external API

Configure HTTP(S) endpoint configuration element



Configure the connection between z/OS Connect EE and the external API.

i ibm.biz/zosconnect-configure-endpoint-connection



Steps to calling an external API

Update the server XML configuration for the endpoint

The screenshot shows a Swagger JSON editor with the following content:

```
swagger: "2.0"
info:
  description: ""
  version: "1.0.0"
  title: "cscvinc"
  host: "localhost:8080"
  basePath: "/cscvinc"
schemes:
  0: "https"
  1: "http"
consumes:
  0: "application/json"
produces:
  0: "application/json"
paths:
  /employee:
    post:
```

Below the editor is a DB2 CLIST window titled "CSC02I01" containing the following data:

Column	Value
03 BAQ-APINAME	PIC X(255) VALUE 'cscvinc_1.0.0'.
03 BAQ-APINAME-LEN	PIC S9(9) COMP-5 SYNC VALUE 13.
03 BAQ-APIPATH	PIC X(255) VALUE '/cscvinc/employee/{numb}'.
03 BAQ-APIPATH-LEN	PIC S9(9) COMP-5 SYNC VALUE 24.
03 BAQ-APIMETHOD	PIC X(255) VALUE 'GET'.
03 BAQ-APIMETHOD-LEN	PIC S9(9) COMP-5 SYNC VALUE 3.

cscvinc.properties
connectionRef=cscvincAPI

The screenshot shows the "Server Config" interface with the file "apiRequesterHTTPS.xml" open. A red oval highlights the XML code for the endpoint connection:

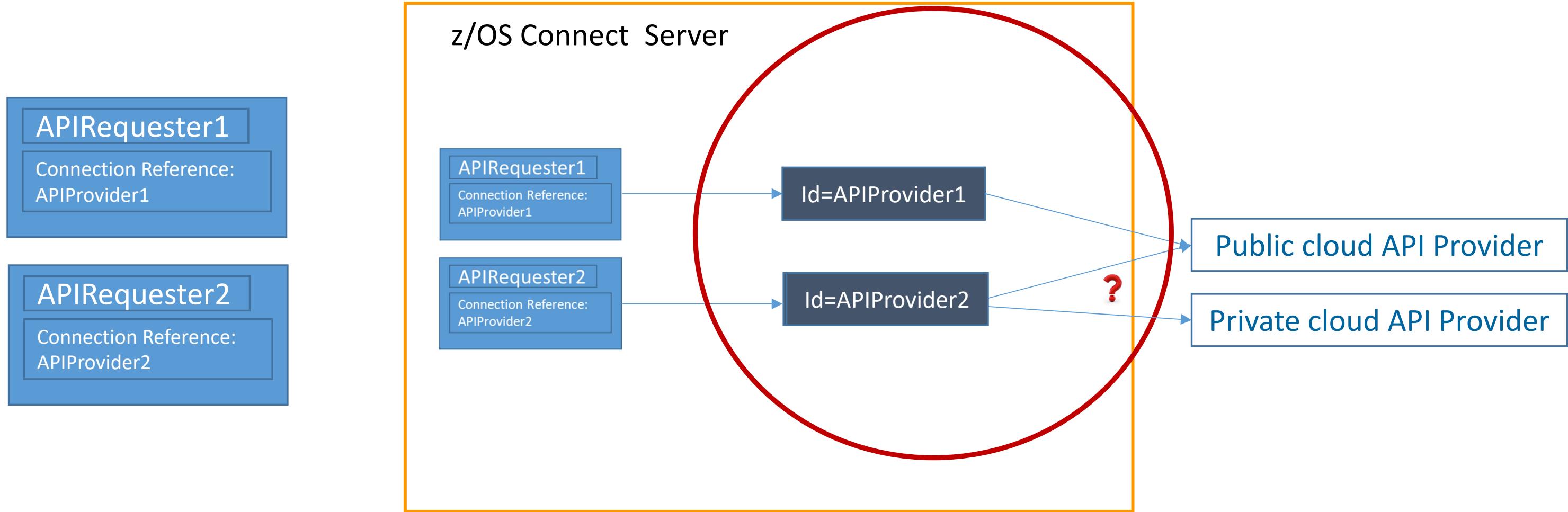
```
<zosconnect_endpointConnection id="cscvincAPI"
  host="https://dvipa.washington.ibm.com"
  port="9443"
  authenticationConfigRef="mySAFAuth"
  connectionTimeout="10s"
  receiveTimeout="40s" />
```

A red arrow points from the highlighted XML code to the URL "http://dvipa.washington.ibm.com:9443/cscvincapi/employee/{numb}" at the bottom right.



Use naming conventions for connection references

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

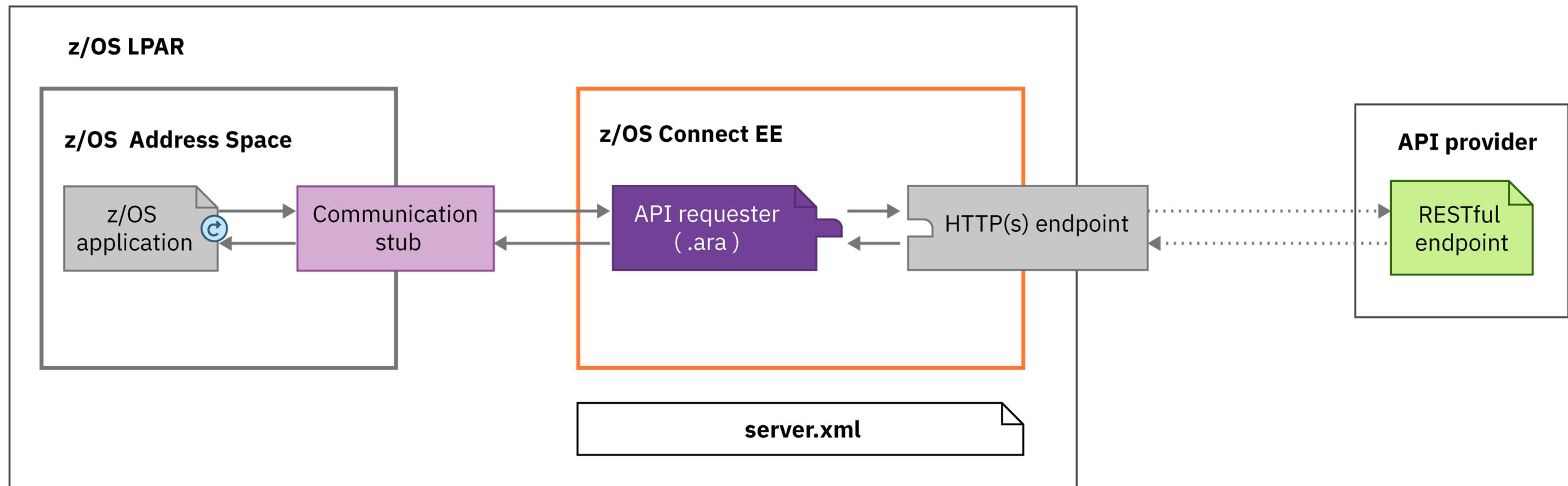


```
<zosconnect_apiRequesters>
  requireAuth="true|false"
  <apiRequester name="cscvincapi_1.0.0"
    connectionRef="APIProvider2"
  </zosconnect_apiRequesters>
```



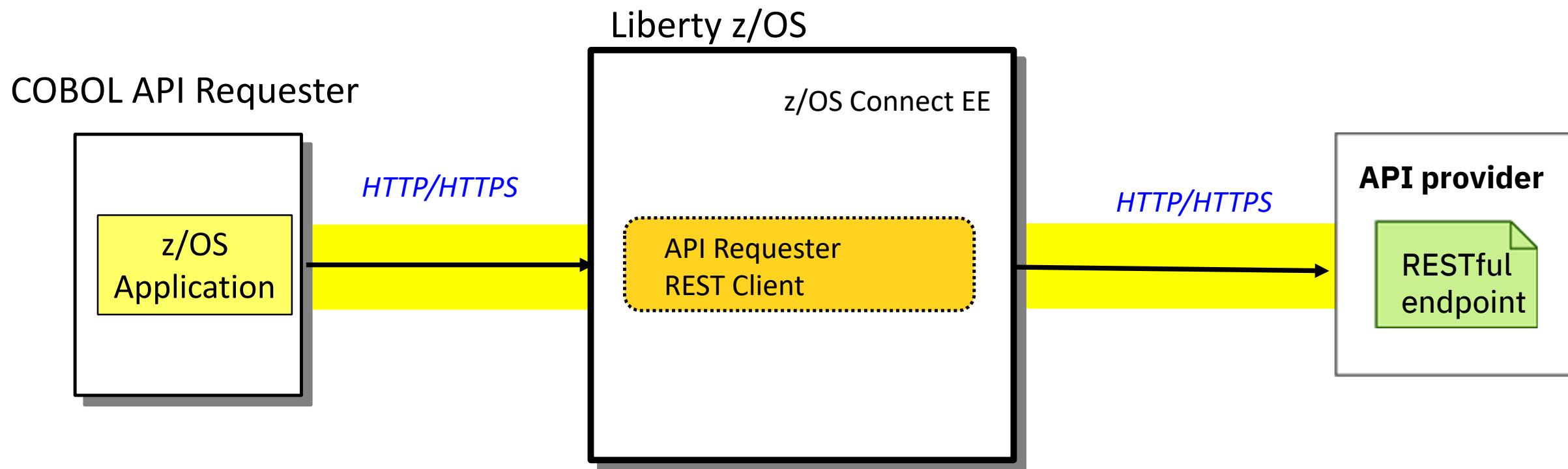
Steps to calling an external API

Done





API requester to API Provider connection overview



MVS Batch and IMS HTTP connection details provided by:

- Environment Variables (BAQURI, BAQPURT)
 - Via JCL
 - LE Options (CEEROPTS)
 - Programmatically (CEEENV)
- HTTP or HTTPS

CICS HTTP connection details provided by:

- CICS URIMAP resource (default BAQURIMP)
 - HOST
 - PORT
 - SCHEME (HTTP/HTTPS)



Configure connections to the z/OS API requester server

Default CICS URI MAP*

```

WG31 - 3270
File Edit Settings View Communication Actions Window Help
I URIMAP
RESULT - OVERTYPE TO MODIFY
  UriMap(BAQURIMP)
  Usage(Client)
  EnableStatus( Enabled )
  AvailStatus(Notapplic)
  Scheme(Http)
  RedirectType( None )
  TcpipService()
  Port(09120)
  Host(wg31.washington.ibm.com:9120)
  Path(/)
  AnalyzerStat(Noanalyzer)
  Hosttype(Hostname)
  Ipresolved(0.0.0.0)
  Ipfamily(Unknown)
  Socketclose(000030)
  Sockpoolsize(000000)
  Transaction()
+ Converter()

SYSID=CICS APPLID=CICS53Z
TIME: 10.38.37 DATE: 02/14/22
PF 1 HELP 2 HEX 3 END      5 VAR      7 SBH 8 SFH      10 SB 11 SF
01/012
MA D
Connected to remote server/host wg31a using lu/pool TCP00120 and port 23
Adobe PDF on Documents\*.pdf

```

* V3.0.37 added support for a CICS application to specify or request a specific URIMAP resource the using BAQ-ZCON-SERVER-URI variable in BAQRINFO

LE Environment Variables

```

//DELTAPI EXEC PGM=DELTAPI, PARM='323232'
//STEPLIB DD DISP=SHR, DSN=USER1.ZCEE.LOADLIB
//          DD DISP=SHR, DSN=ZCEE30.SBAQLIB
//SYSOUT DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//CEEOPTS DD *
  POSIX(ON),
ENVAR("BAQURI=wg31.washington.ibm.com",
"BAQPORT=9120")

```

```

mpz3
File Edit Settings View Communication Actions Window Help
Menu Utilities Compilers Help
BROWSE ZCEE30.SBAQC0B(BAQRINFO) Line 000000010 Col 001 080
Command ==>
* (C) Copyright IBM Corp. 2017, 2021
* US Government Users Restricted Rights - Use, duplication or
* disclosure restricted by GSA ADP Schedule Contract with
* IBM Corp.
***** This file contains the generated language structure(s) for ****
* Request and Response Info
*****
* BAQ-REQUEST-INFO-COMP-LEVEL permitted values
* VALUE
* 0 Base support
* 1 Added support for BAQ-OAUTH
* 2 Added support for BAQ-TOKEN (JWT)
* 3 Added support for setting z/OS Connect EE server URI
* 4 Added support for BAQ-OAUTH-EXT
*****
01 BAQ-REQUEST-INFO.
03 BAQ-REQUEST-INFO-COMP-LEVEL PIC S9(9) COMP-5 SYNC VALUE 4.
03 BAQ-REQUEST-INFO-USER
  05 BAQ-OAUTH.
    07 BAQ-OAUTH-USERNAME PIC X(256).
    07 BAQ-OAUTH-USERNAME-LEN PIC S9(9) COMP-5 SYNC
    VALUE 0.
    07 BAQ-OAUTH-PASSWORD PIC X(256).
    07 BAQ-OAUTH-PASSWORD-LEN PIC S9(9) COMP-5 SYNC
MA A
Connected to remote server/host mpz3 using lu/pool MPZ30044 and port 23

```



Runtime Environment variables

Use these runtime environment variables when connecting to a z/OS Connect server

BAQPASSWORD - Specifies the password, in clear text, for the specified BAQUSERNAME to be authenticated with the z/OS Connect server. The username and password that are used for basic authentication, when SSL mutual authentication is not enabled.

BAQPORT - Specifies the port number for the z/OS Connect server.

BAQTIMEOUT - An optional 4-byte integer to set a timeout value in seconds for waiting for an API response. Valid range is 1 - 2,678,400 seconds. The default timeout value is 10 seconds.

BAQURI - Specifies either an IPv4 or IPV6 address, or a hostname of the host where the z/OS Connect server resides.

BAQUSERNAME - Specifies the username for connections if basic authentication is used.

BAQVERBOSE - An optional value to turn on verbose messages to assist debugging of runtime and configuration issues. Valid values are **OFF**, **ON**, **ERROR**, **AUDIT** and **ALL**. See URL <https://www.ibm.com/docs/en/zos-connect/zosconnect/3.0?topic=car-configuring-other-zos-applications-access-zos-connect-api-calls> for more information.



Basic authentication – 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: A PassTicket provides an alternative to a password



- A PassTicket is generated by or for a client by using a secured sign-on key (whose value is masked or encrypted) to encrypt a valid *RACF identity* combined with the *application name* of the targeted resource. Also embedded in the PassTicket is a time stamp (based on the current Universal Coordinated Time (UCT)) which sets the time when the PassTicket will expire (usually 10 minutes).
- Access to PassTickets is managed using the RACF PTKTDATA class.
- For z/OS Connect, a RACF PassTicket can be used for basic authentication when connecting from any REST client on any platform to a z/OS Liberty server and for requests from a z/OS Connect server accessing IMS and Db2.
- ***PassTickets do not have to be generated on z/OS using RACF services.*** IBM has published the algorithm used to generate a PassTickets, see manual *z/OS Security Server RACF Macros and Interfaces, SA23-2288-40*. *Github has examples using Java, Python and other example are available on other sites.*

```
<safRegistry id="saf" />
  <safAuthorization racRouteLog="ASIS" />
  <safCredentials unauthenticatedUser="WSGUEST"
    profilePrefix="BBGZDEFLT" />
```



Tech/Tip: Generating PassTickets on z/OS

- On z/OS, a COBOL user application can generate a pass tickets by calling RACF service IRRSPK00:

```

77 COMM-STUB-PGM-NAME          PIC X(8) VALUE 'BAQCSTUB'.
77 PTKT-STUB-PGM-NAME          PIC X(8) VALUE 'ATSPTKTC'.

*-----*
***** L I N K A G E   S E C T I O N *****
*-----*
LINKAGE SECTION.

*-----*
* P R O C E D U R E S
*-----*
PROCEDURE DIVISION using PARM-BUFFER.

*-----*
MAINLINE SECTION.

*-----*
* Common code
*-----*
* initialize working storage variables
INITIALIZE GET-REQUEST.
INITIALIZE GET-RESPONSE.
CALL PTKT-STUB-PGM-NAME.

```

JOHNSON.PASSTCKT.SOURCE (ATSPTKTC)

```

*-----*
* Build IRRSPK00 parameters
*-----*
MOVE 0 to ws-length
MOVE LENGTH OF identity to identity-length.
INSPECT FUNCTION REVERSE (identity)
      TALLYING ws-length FOR ALL SPACES.
SUBTRACT ws-length FROM identity-length.
MOVE 0 to ws-length
MOVE LENGTH OF applid to applid-length.
INSPECT FUNCTION REVERSE (applid)
      TALLYING ws-length FOR ALL SPACES.
SUBTRACT ws-length FROM applid-length.
MOVE 8 to passTicket-length.
MOVE 'NOTICKET' to passTicket.
MOVE X'0003' to irr-functionCode.
MOVE X'00000001' to irr-ticketOptions.
SET irr-ticketOptions-ptr to ADDRESS OF irr-ticketOptions.
*-----*
* Call RACF service IRRSPK00 to obtain a pass ticket based
*   on identity and applid
*-----*
PERFORM CALL-RACF.
IF irr-safrc NOT = zero then
  DISPLAY "SAF_return_code:      " irr-safrc
  DISPLAY "RACF_return_code:     " irr-racfrc
  DISPLAY "RACF_reason_code:    " irr-racfrsn
End-if
. . .
*-----*
* Call IRRSPK00 requesting a pass ticket
*-----*
CALL-RACF.
  CALL W-IRRSPK00 USING irr-workarea,
    IRR-ALET, irr-safrc,
    IRR-ALET, irr-racfrc,
    IRR-ALET, irr-racfrsn,
    IRR-ALET, irr-functionCode,
    irr-optionWord,
    IRR-PASSTICKET,
    irr-ticketOptions-ptr,
    IRR-IDENTITY,
    IRR-APPLID

```



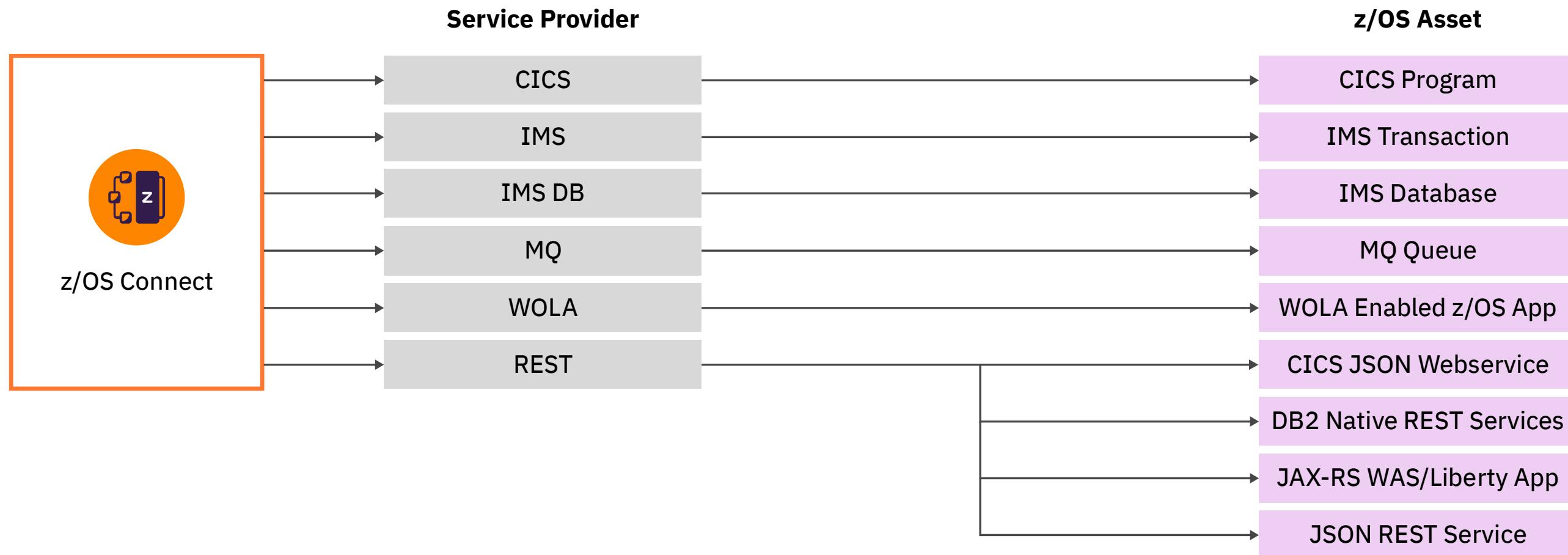
/miscellaneousTopics

performance, high availability, Liberty



What assets can z/OS Connect EE map to?

And which service provider could I use?

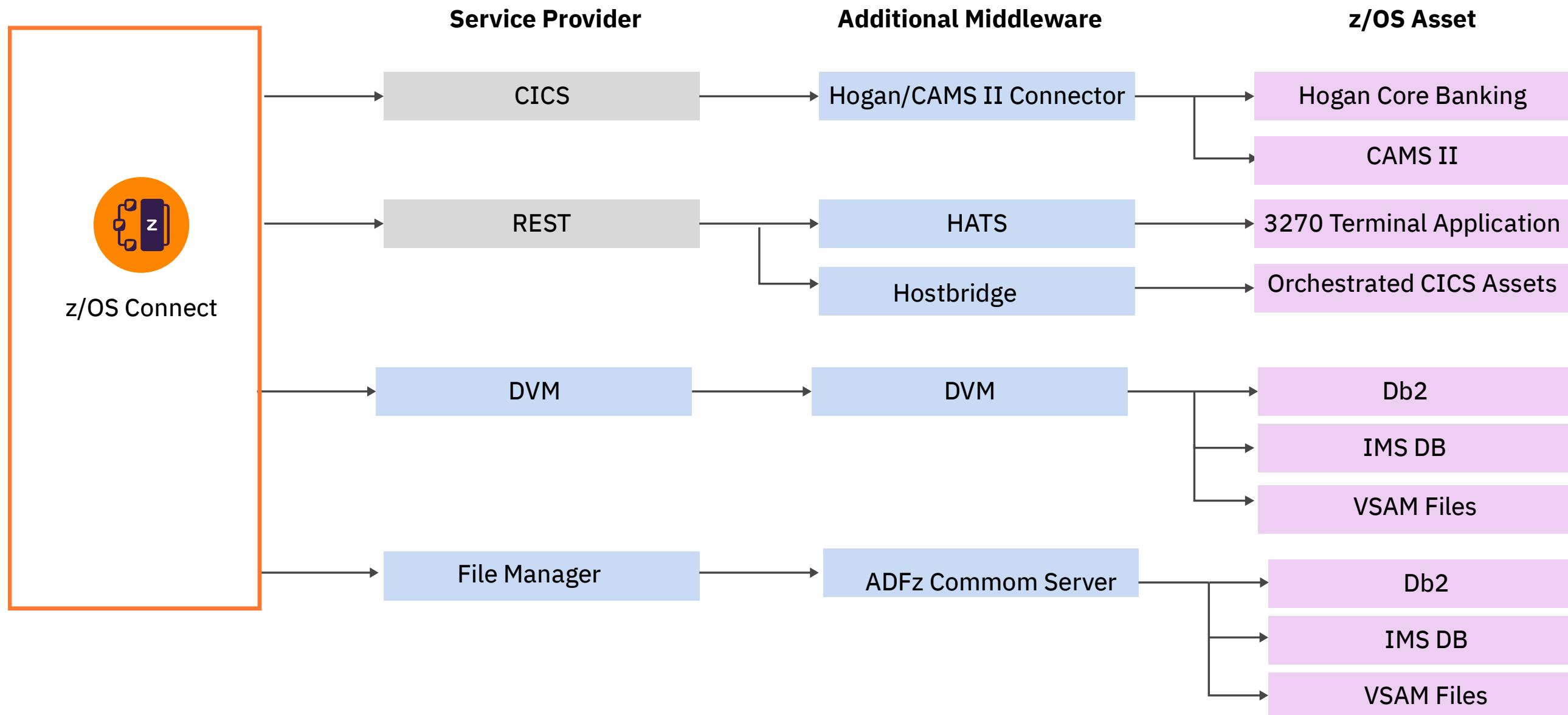


The core **service providers** included with z/OS Connect EE provide API access to a wide range of z/OS assets.



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

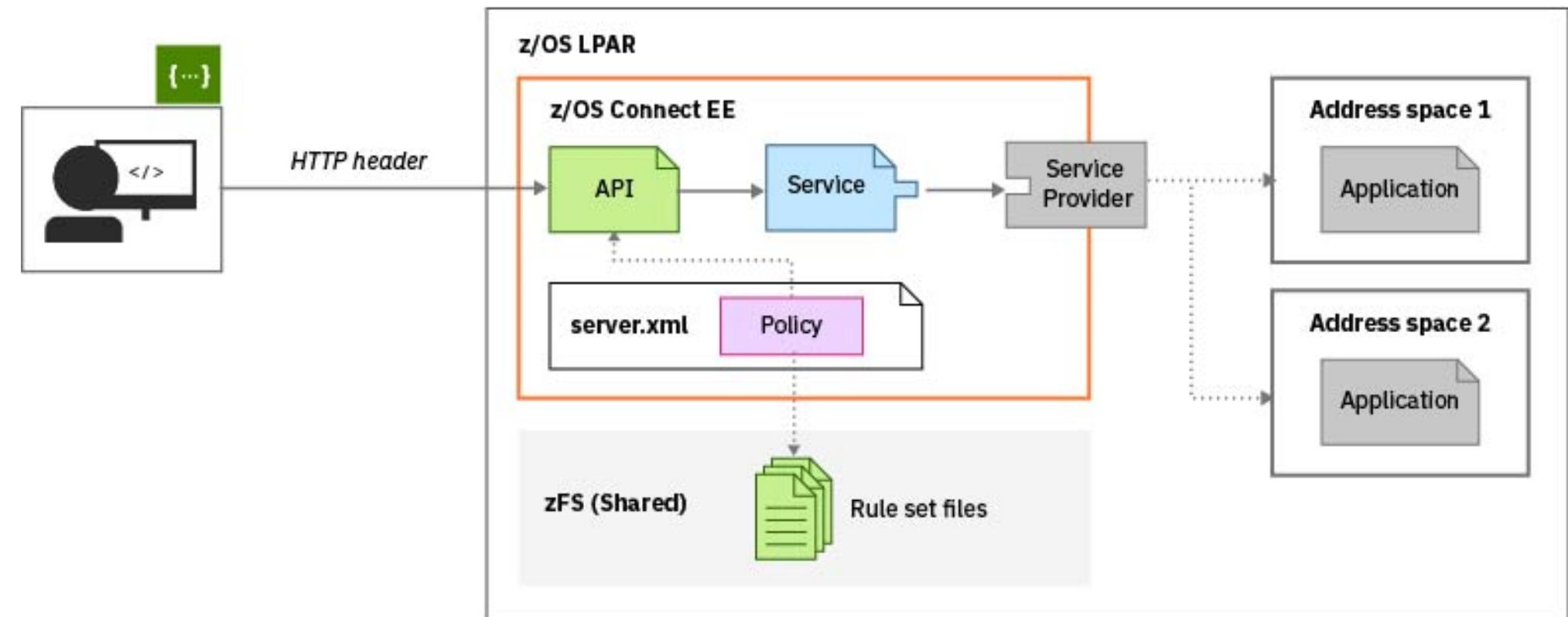
- imsConnectionRef
- imsInteractionRef
- imsInteractionTimeout
- imsItermOverrideName
- imsTranCode
- imsTranExpiration

Db2 attributes

- db2ConnectionRef
- db2CollectionID

MQ attributes

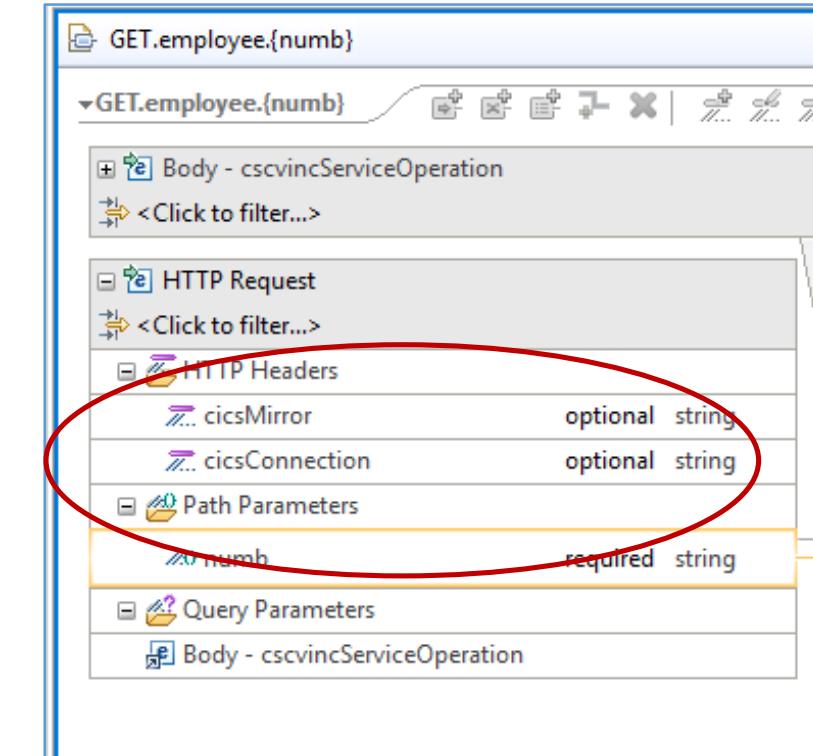
- mqConnectionFactory
- mqDestination
- mqReplyDestination





A sample API Policies for CICS

```
<ruleset name="CICS rules">
  <rule name="csmi-rule">
    <conditions>
      <header name="cicsMirror" value="CSMI,MIJO"/> *
    </conditions>
    <actions>
      <set property="cicsTransId" value="${cicsMirror}"/>
    </actions>
  </rule>
  <rule name="connection-rule">
    <conditions>
      <header name="cicsConnection"
             value="cscvinc,cics92,cics93"/>
    </conditions>
    <actions>
      <set property="cicsConnectionRef" value="${cicsConnection}">
    </actions>
  </rule>
</ruleset>
```



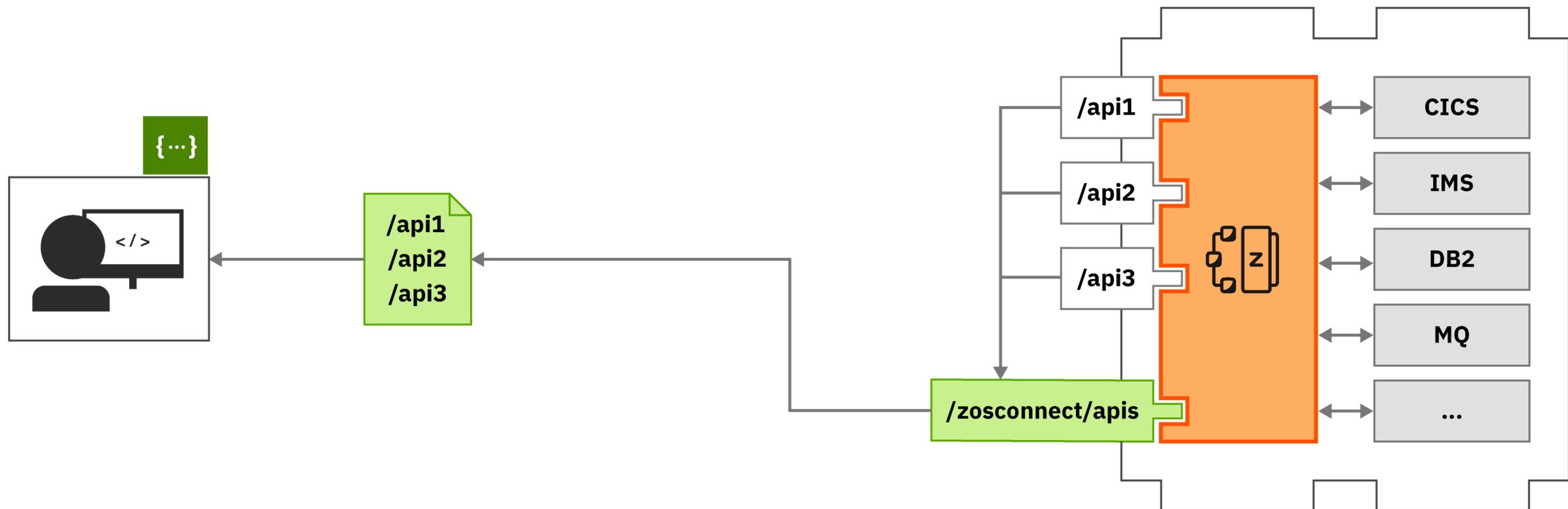
Curl

```
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)



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

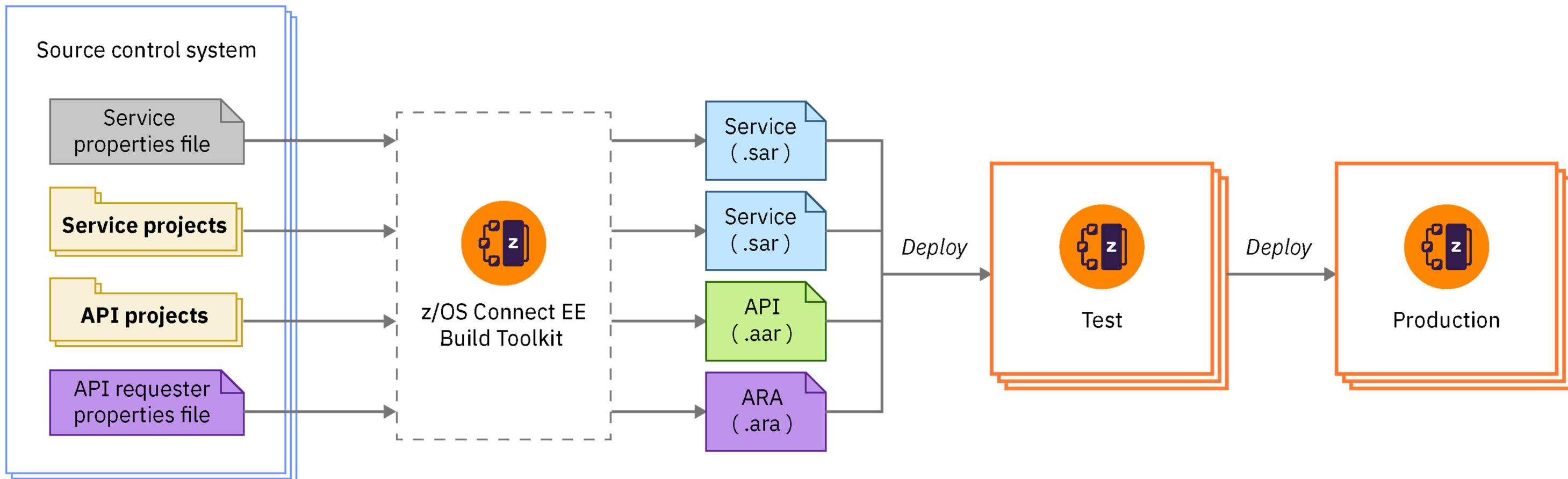
GET	/apiRequesters	Returns a list of all the deployed z/OS Connect API Requesters
POST	/apiRequesters	Deploys a new API Requester into z/OS Connect and invoke an API Requester call
DELETE	/apiRequesters/{apiRequesterName}	Undeploys an API Requester from z/OS Connect
GET	/apiRequesters/{apiRequesterName}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



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



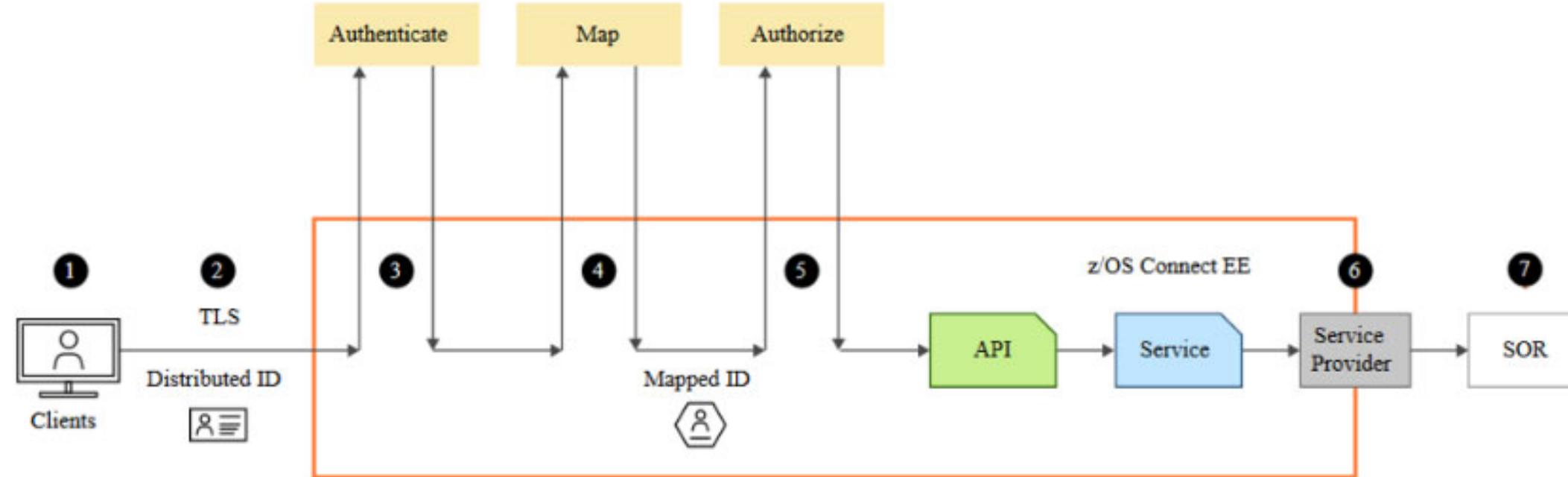


/security

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



OPENAPI 3 roles

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

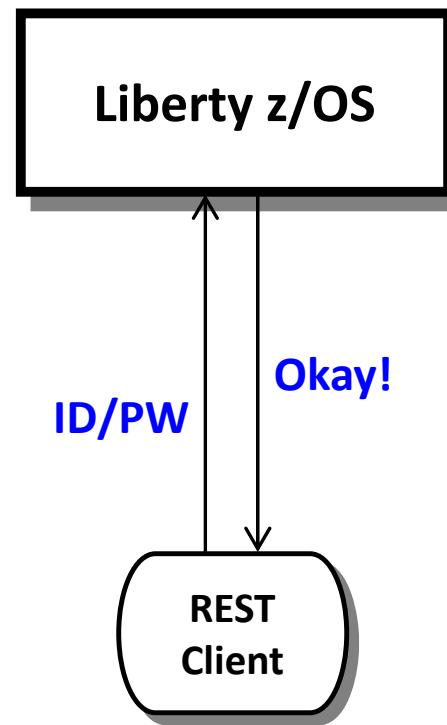
<authorization-roles id="Manager">
  <security-role name="Manager">
    <group name="managerGroup"/>
  </security-role>
</authorization-roles>
<authorization-roles id="Staff">
  <security-role name="Staff">
    <group name="staffGroup"/>
  </security-role>
</authorization-roles>
```



API Provider Authentication

Several different ways this can be accomplished:

Basic Authentication



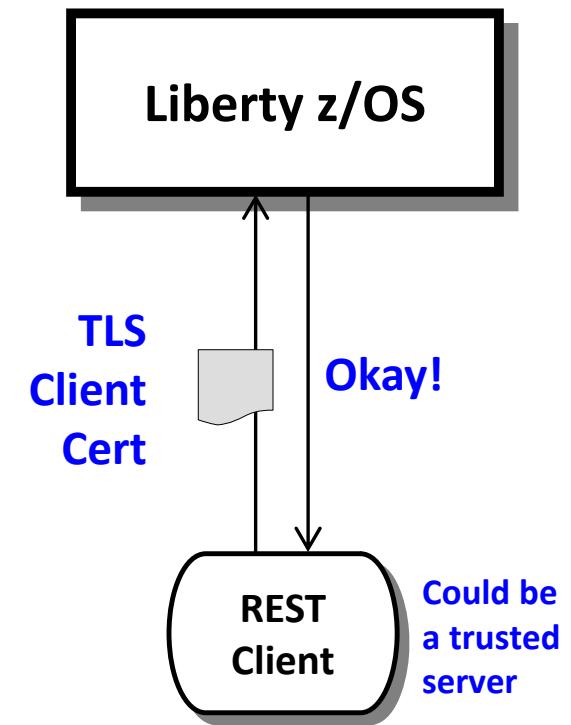
Server prompts for ID/PW

Client supplies ID/PW or
ID/Passticket

Server checks registry:

- Basic (server.xml)
- LDAP
- SAF

Client Certificate



Server prompts for cert.

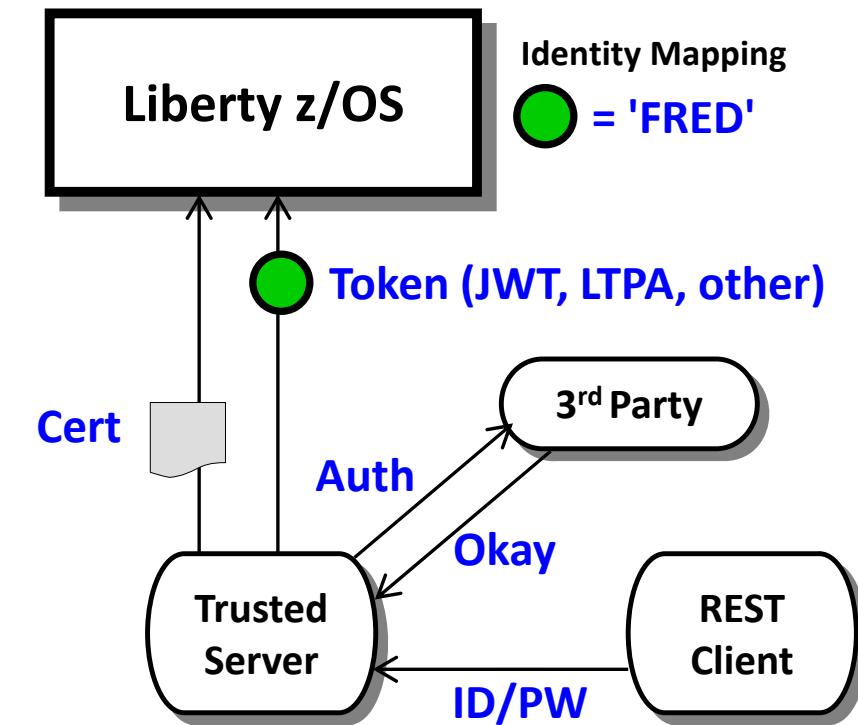
Client supplies certificate

Server validates cert and
maps to an identity

Registry options:

- LDAP
- SAF

Third Party Authentication



Client authenticates to 3rd party sever

Client receives a trusted 3rd party token

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

Registry options:

- LDAP
- SAF



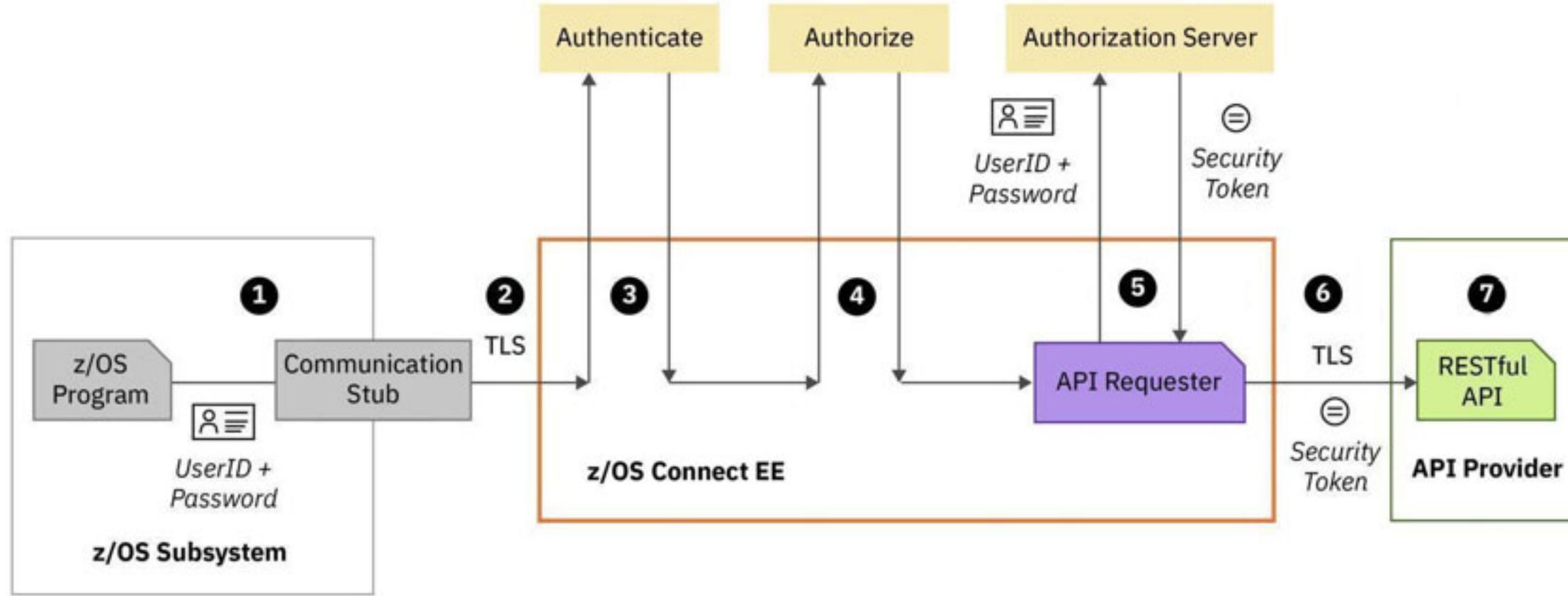
Third Party Authentication Examples

Screenshot of the UPS Sign Up page. At the top, there's a yellow banner stating "UPS is open for business: Service impacts related to Coronavirus ...More". Below the banner, the UPS logo is on the left, followed by "Sign up / Log in" and a search bar. A "Feedback" button is located on the right side of the page. The main section is titled "Sign Up" and includes a link "Already have an ID? Log in". It says "Use one of these sites." and lists "Google", "Facebook", "Amazon", "Apple", and "Twitter" with their respective icons. Below these, it says "Or enter your own information." and has fields for "Name *", "Email *", "User ID *", "Password *", and "Phone". A "Show" link is next to the password field.

Screenshot of the myNCDMV Sign In page. The background features a scenic view of autumn foliage. The page has "Log In" and "Sign Up" tabs at the top. The "Log In" tab is selected, leading to a "Log In to myNCDMV" section. It includes fields for "Email Address" (with "name@example.com" entered) and "Password" (with "*****" shown). There's a "Remember Me" checkbox, a "Log In" button, a "Forgot Password" link, and a "Continue as Guest" link. Below these, there are links for "Continue with Apple", "Continue with Facebook", and "Continue with Google". A notice for public computer users states: "NOTICE FOR PUBLIC COMPUTER USERS - If you sign in with Google, Apple, or Facebook you are also signing into that account on this computer. Remember to sign out when you're done." The page is powered by "payit".



Typical z/OS Connect EE API Requester security flow



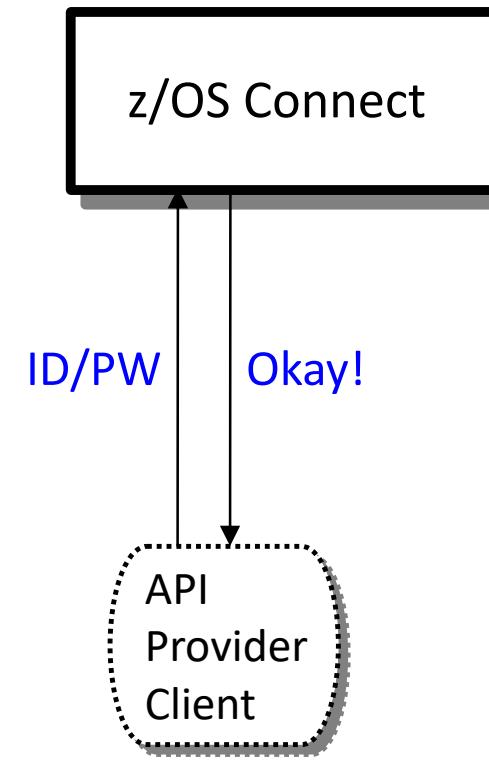
1. A user ID and password can be used for basic authentication by the z/OS Connect EE server
2. Connection between the CICS, IMS, or z/OS application and the z/OS Connect EE server can use TLS
3. Authenticate the CICS, IMS, or z/OS application.
4. Authorize the authenticated user ID to connect to z/OS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



z/OS Application to z/OS Connect API Requester

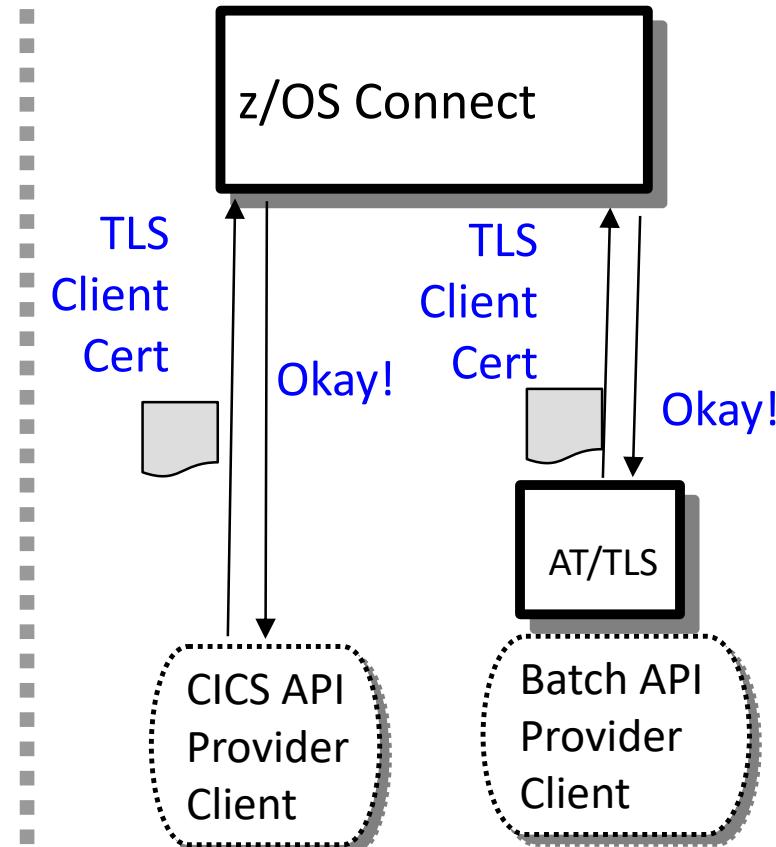
Two options for providing credentials for authentication

Basic Authentication



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

Client Certificate



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

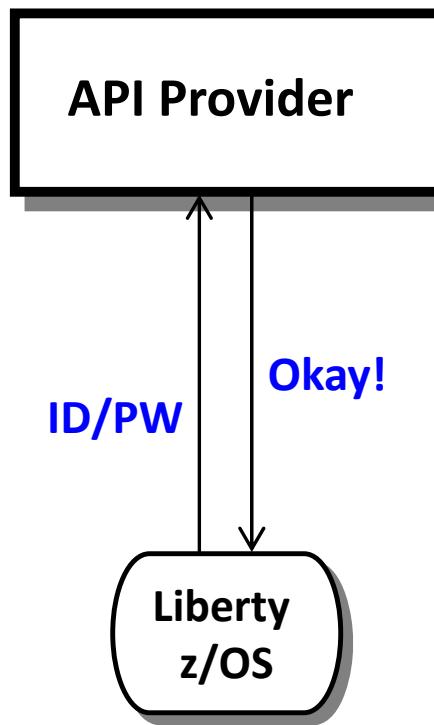
**CICS or AT/TLS supplies a
client certificate**



API Requester - API Provider Authentication

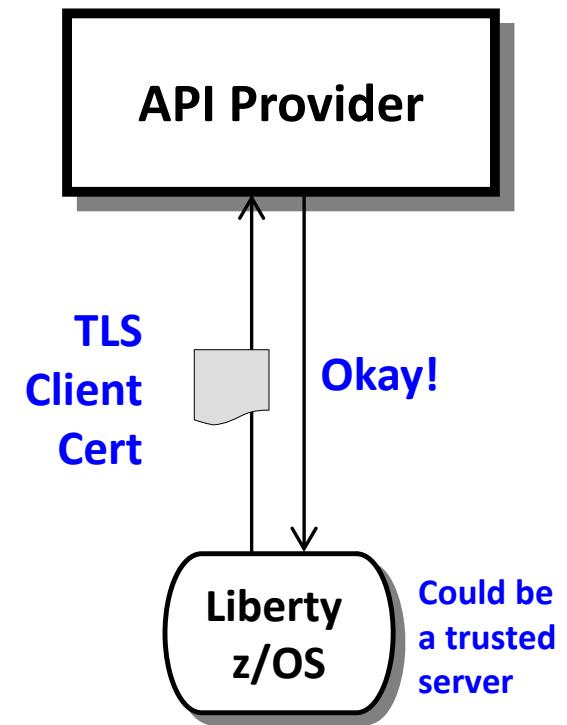
Several different ways this can be accomplished:

Basic Authentication



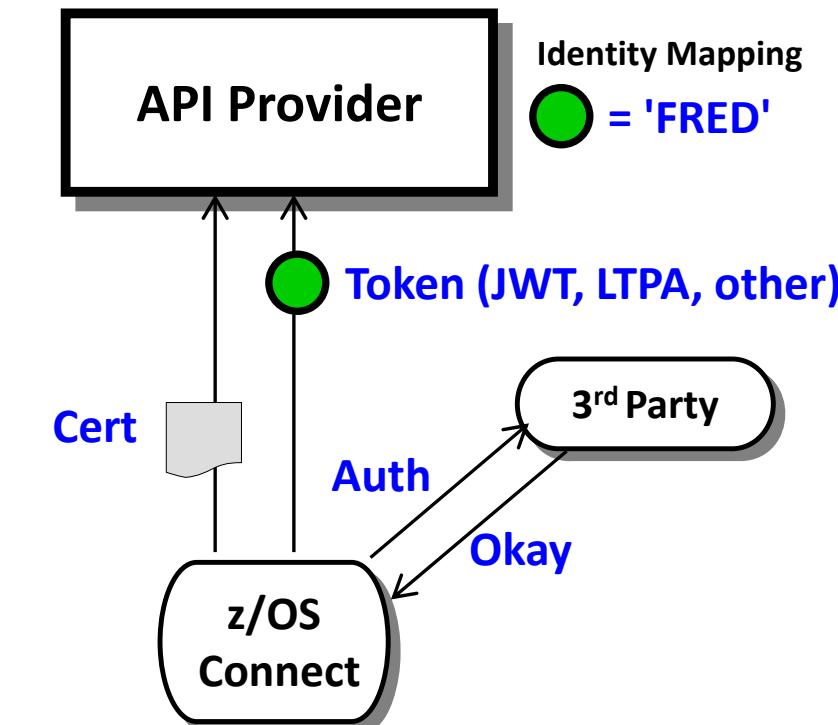
zCEE supplies ID/PW or
ID/Passticket

Client Certificate



Server prompts for certificate
zCEE supplies certificate

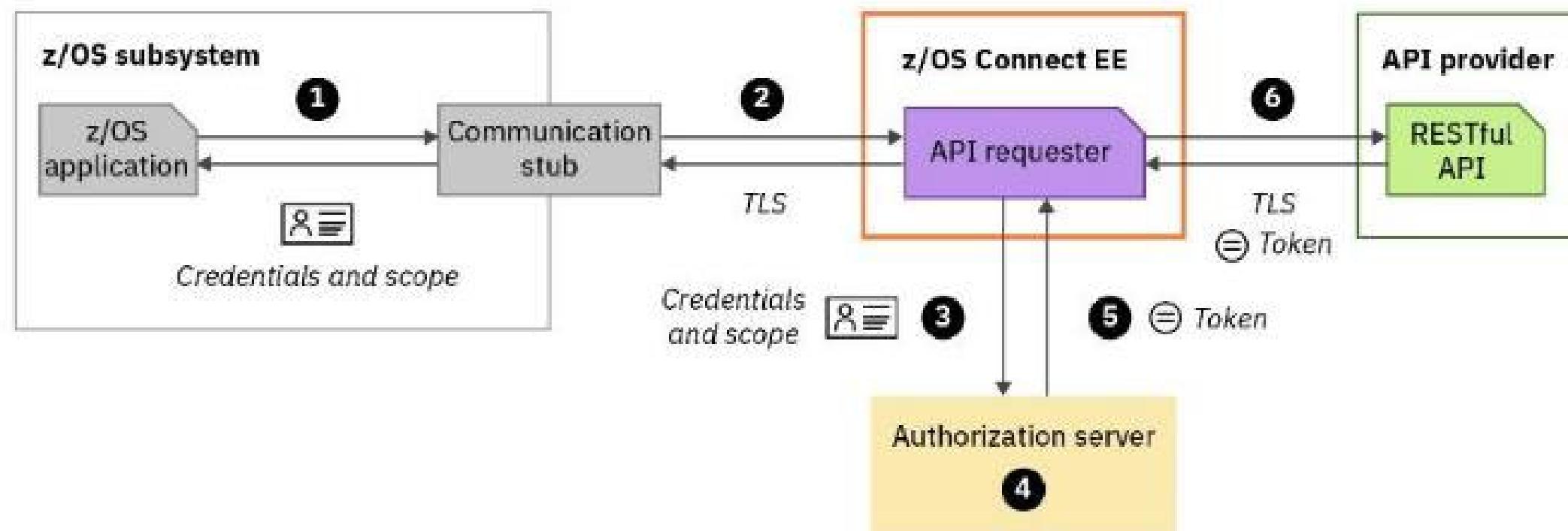
Third Party Authentication



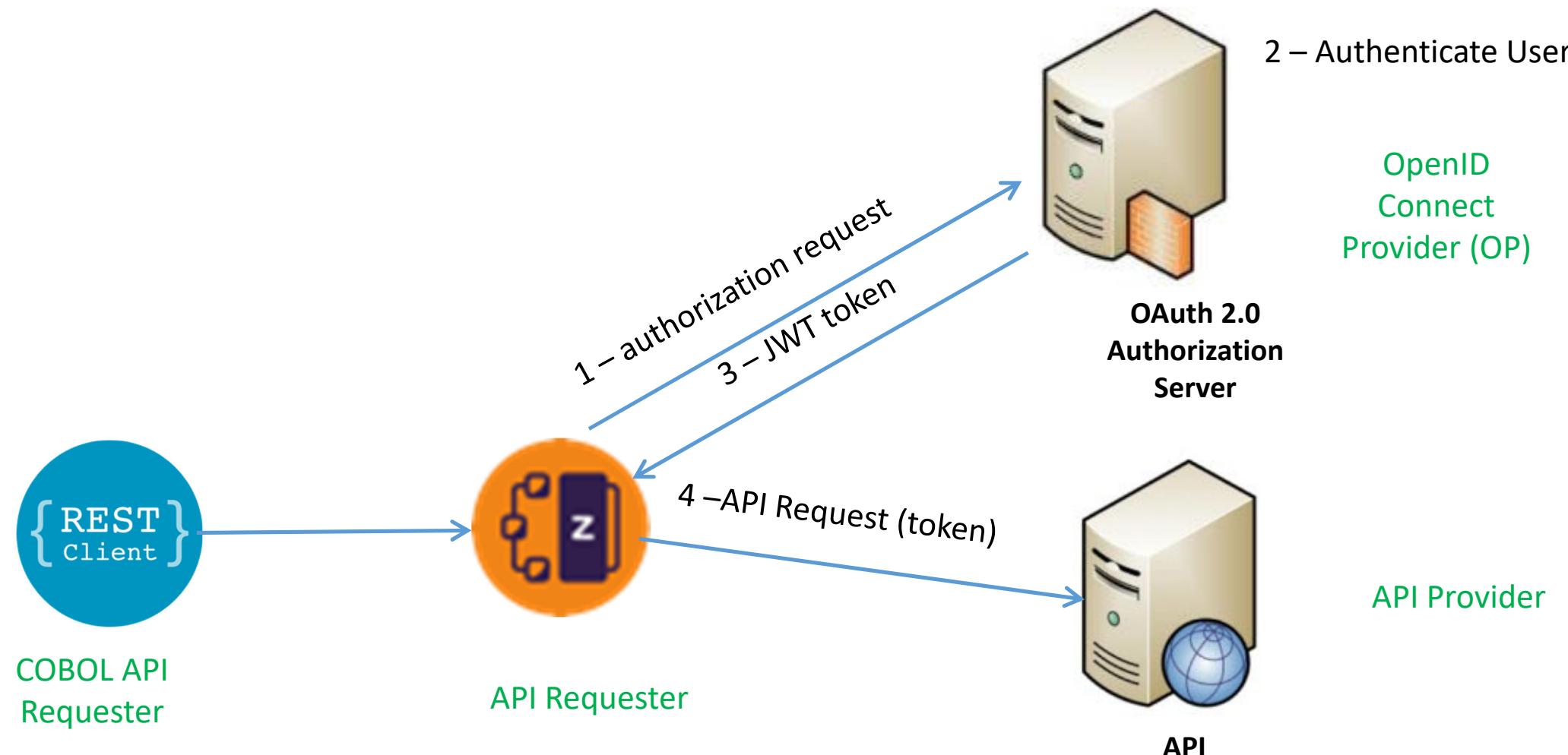
zCEE authenticates to 3rd party sever
zCEE receives a trusted 3rd party token
Token flows to API Provider



Calling an API with OAuth 2.0 support



z/OS Connect OAuth Flow for API requester



Grant Types:

- client_credentials
- password



Configuring OAuth support – BAQRINFO copy book

```
05 BAQ-OAUTH.  
 07 BAQ-OAUTH-USERNAME          PIC X(256).  
 07 BAQ-OAUTH-USERNAME-LEN      PIC S9(9) COMP-5 SYNC VALUE 0.  
 07 BAQ-OAUTH-PASSWORD          PIC X(256).  
 07 BAQ-OAUTH-PASSWORD-LEN      PIC S9(9) COMP-5 SYNC VALUE 0.  
 07 BAQ-OAUTH-CLIENTID          PIC X(256).  
 07 BAQ-OAUTH-CLIENTID-LEN      PIC S9(9) COMP-5 SYNC VALUE 0.  
 07 BAQ-OAUTH-CLIENT-SECRET    PIC X(256).  
 07 BAQ-OAUTH-CLIENT-SECRET-LEN PIC S9(9) COMP-5 SYNC VALUE 0.  
 07 BAQ-OAUTH-SCOPE-PTR        USAGE POINTER.  
 07 BAQ-OAUTH-SCOPE-LEN        PIC S9(9) COMP-5 SYNC.
```

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

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.

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

<https://ibm.biz/zCEEWorkshopMaterial>



The screenshot shows the GitHub repository 'ibm-wsc/zCONNEE-Wildfire-Workshop' on the 'master' branch. The repository has 1 branch and 0 tags. The 'Code' button is highlighted. The commit history shows 457 commits by emitchj. A red oval highlights the 'OpenAPI2' and 'OpenAPI3' directories. Other visible files include APIRequesters, AdminSecurity, COBOL Samples, XML Samples, README.md, ZCADMIN - zOS Connect ..., ZCEESEC - zOS Connect Se..., ZCINTRO - Introduction to..., ZCREQUEST - Introduction..., zOS Connect EE V3 Advan..., and zOS Connect EE V3 Gettin... .

The screenshot shows the GitHub repository 'zCONNEE-Wildfire-Workshop' with two branches: 'OpenAPI2' and 'OpenAPI3'. The 'OpenAPI2' branch contains files for 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.pdf, Developing RESTful APIs for Db2 REST Services.pdf, Developing RESTful APIs for HATS REST Service.pdf, Developing RESTful APIs for IMS DVM Services.pdf, Developing RESTful APIs for IMS Database RES.pdf, and Developing RESTful APIs for IMS Transactions.pdf. The 'OpenAPI3' branch contains files for Developing RESTful APIs for Db2 REST Services.pdf and Developing RESTful APIs for a CICS Program.pdf.

- 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> 
Subject: Got a minute? Two questions on your IBM Z Washington Systems Center experience

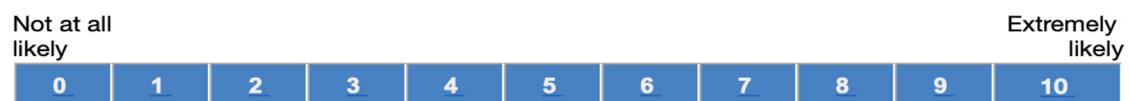


Dear |

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

You can begin the survey by answering this question.

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



Sincerely,

IBM Advocacy Team

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



Thank you for listening and your questions

And thank you for completing the Medallia survey