



IBM z/OS Connect

Introduction and Overview

Mitch Johnson

mitchj@us.ibm.com

Washington System Center



Agenda

- An Introduction and Overview of using REST API
- Enabling RESTful API to various z/OS resources, e.g.
 - CICS
 - Db2
 - IMS/TM
 - IMS/DB
 - MQ
 - 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 *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.

z/OS Connect Wildfire Github Site <https://ibm.biz/BdPRGD>



The screenshot shows two adjacent GitHub browser tabs. The left tab displays the repository structure for 'zCONNEE-Wildfire-Workshop'. A red oval highlights the 'OpenAPI3' directory under the 'AdminSecurity' folder. The right tab shows the contents of the 'exercises' folder, listing various PDF files related to API development.

File	Description	Last Modified
Developing CICS API Requester Applications.pdf	Add files via upload	2 months ago
Developing IMS API Requester Applications.pdf	Add files via upload	2 months ago
Developing MVS Batch API Requester Applications.pdf	Add files via upload	2 months ago
Developing RESTful APIs for DVM VSAM Services.pdf	Add files via upload	20 days ago
Developing RESTful APIs for DVM VSAMCICS Services.pdf	Add files via upload	20 days ago
Developing RESTful APIs for Db2 REST Services.pdf	Add files via upload	2 months ago
Developing RESTful APIs for HATS REST Services.pdf	Add files via upload	2 months ago
Developing RESTful APIs for IMS Database REST Services....	Add files via upload	2 months ago
Developing RESTful APIs for IMS Transactions.pdf	Add files via upload	2 months ago
Developing RESTful APIs for MQ.pdf	Add files via upload	2 months ago
Developing RESTful APIs for MVS Batch.pdf	Add files via upload	2 months ago
Developing RESTful APIs for a CICS COMMAREA program.pdf	Add files via upload	2 months ago
Developing RESTful APIs for a CICS Container program.pdf	Add files via upload	2 months ago

mitchj@us.ibm.com

- Contact your IBM representative to schedule access to these exercises

© 2018, 2022 IBM Corporation
Slide 4

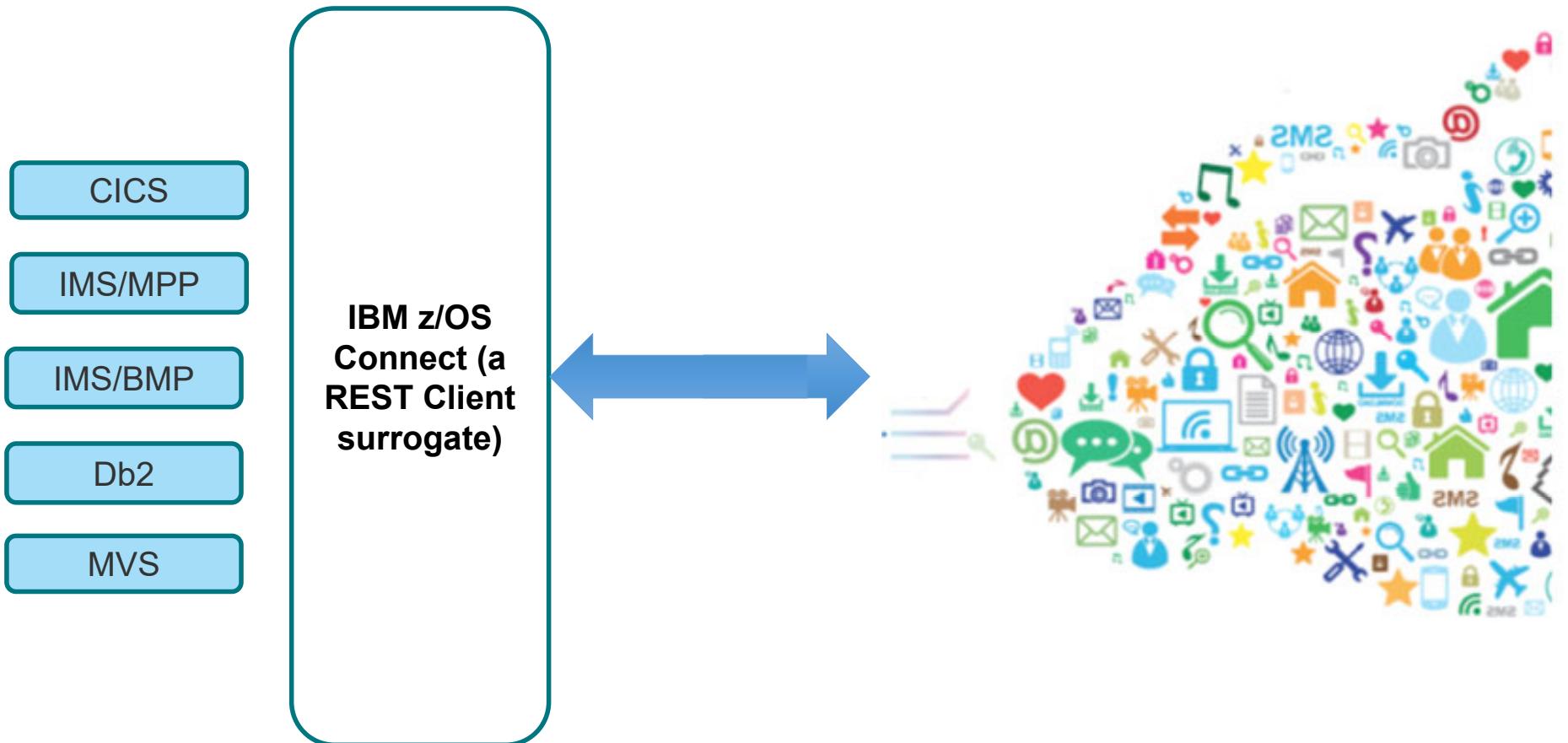
z/OS Connect EE exposes z/OS resources to the “cloud” via RESTful APIs



+ HCL and Rocket Software

*Other Vendors or your own implementation

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



/but_first, what_is_REST?

What makes an API “RESTful”?



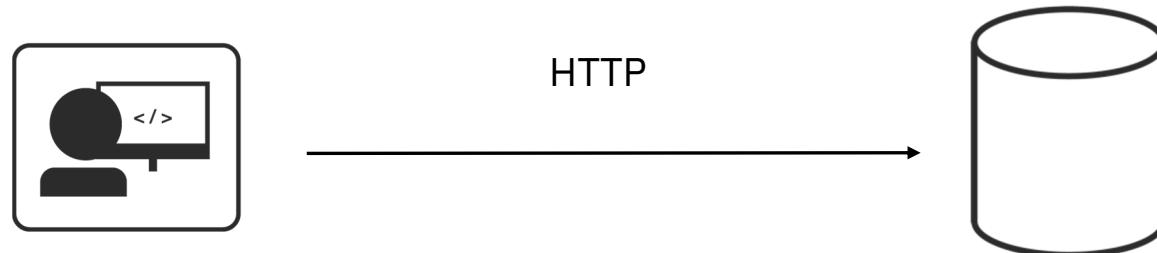
REST is architectural programming style

REST stands 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 REST

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" }



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.

How do we describe and/or document an API?



We use an Open API specification

- It is more than just an API framework



There are a number of tools available to aid consumption:

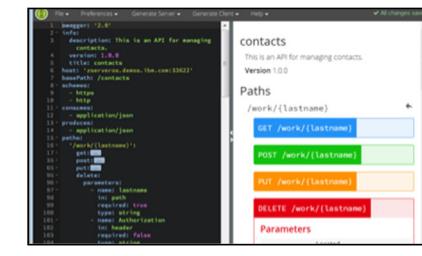
Code Generation* - create stub code to consume APIs from various languages



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.



What is the significance of OpenAPI Specification to z/OS Connect?

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.

- **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 described 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.**



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.

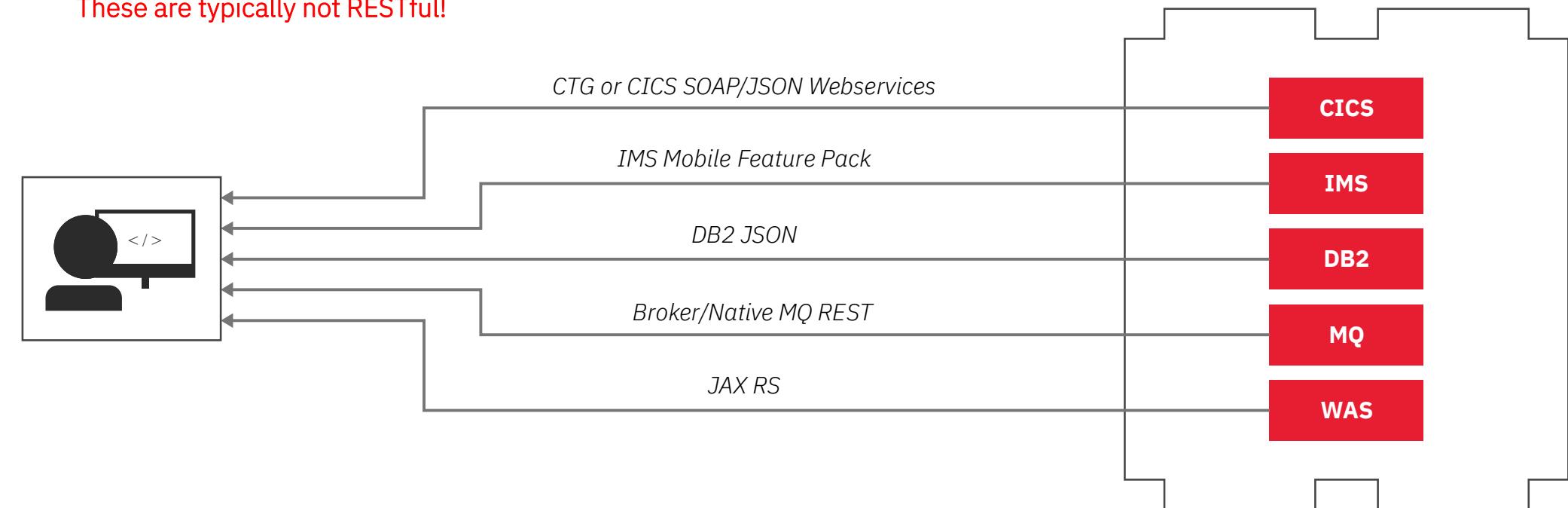


There was support for REST before z/OS Connect but..

Completely different configuration and management.

Multiple endpoints for developers to call/maintain access to.

These are typically not RESTful!



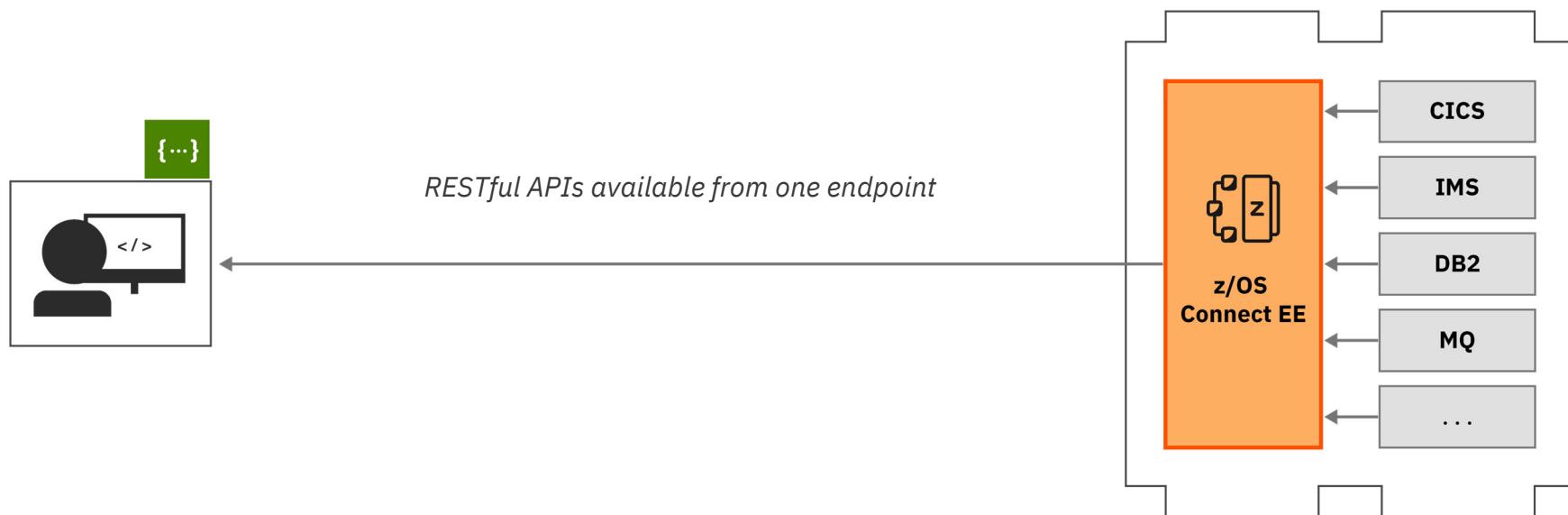


z/OS Connect provides a single-entry point

- 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.



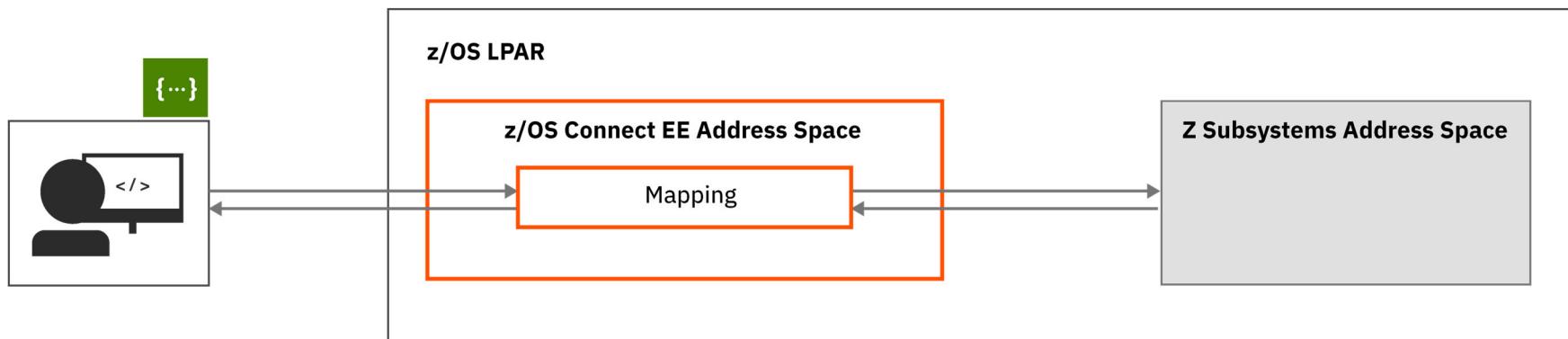


**Other than a RESTful interface,
what else does z/OS Connect provide?**



Data mapping

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

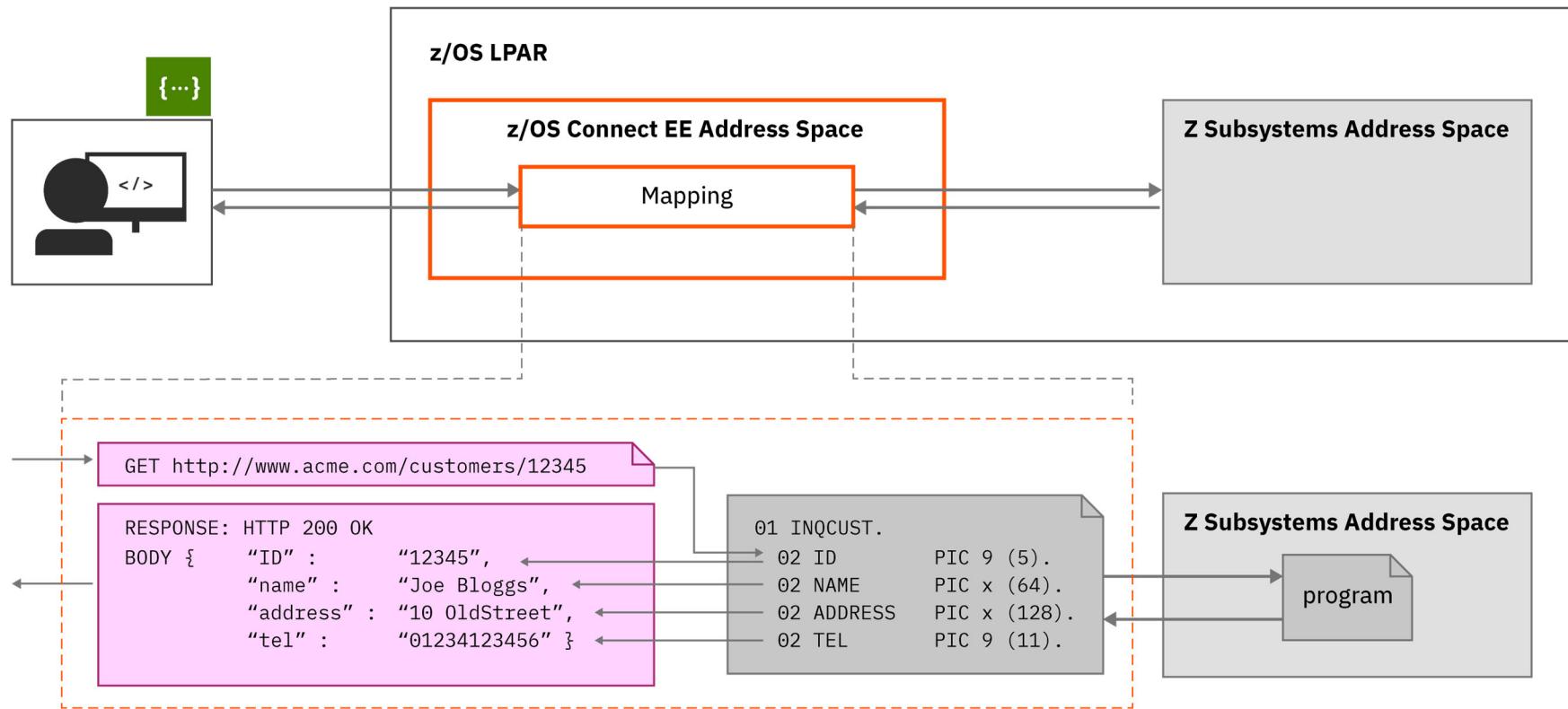


* Most z/OS subsystems depend on information in a serial data format and do not normally work with JSON request/response messages. Examples of non-JSON formats are CICS COMMAREAAs and CONTAINERS, IMS or MQ messages, or records stored in sequential or VSAM data sets. Data mapping and transformation refers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



Data mapping and transformation example

- A closer look





z/OS Connect OpenAPI Tooling

Example of an OPENAPI 2 versus an OPENAPI 3 Response Message



The screenshot shows the Eclipse-based z/OS Connect API Toolkit interface. It displays the response structure for an OPENAPI 2 endpoint named 'GET.employee.{employee}'. The response body is defined as 'cscvincSelectServiceOperationResponse' and contains fields: 'cscvincContainer' (1..1), 'response' (0..1), 'CEIBRESP' (0..1) integer, 'CEIBRESP2' (0..1) integer, 'USERID' (0..1) string, and 'filea' (0..1). Below this, there are sections for 'HTTP Response' and 'HTTP Headers'.

Eclipse based - z/OS Connect API Toolkit

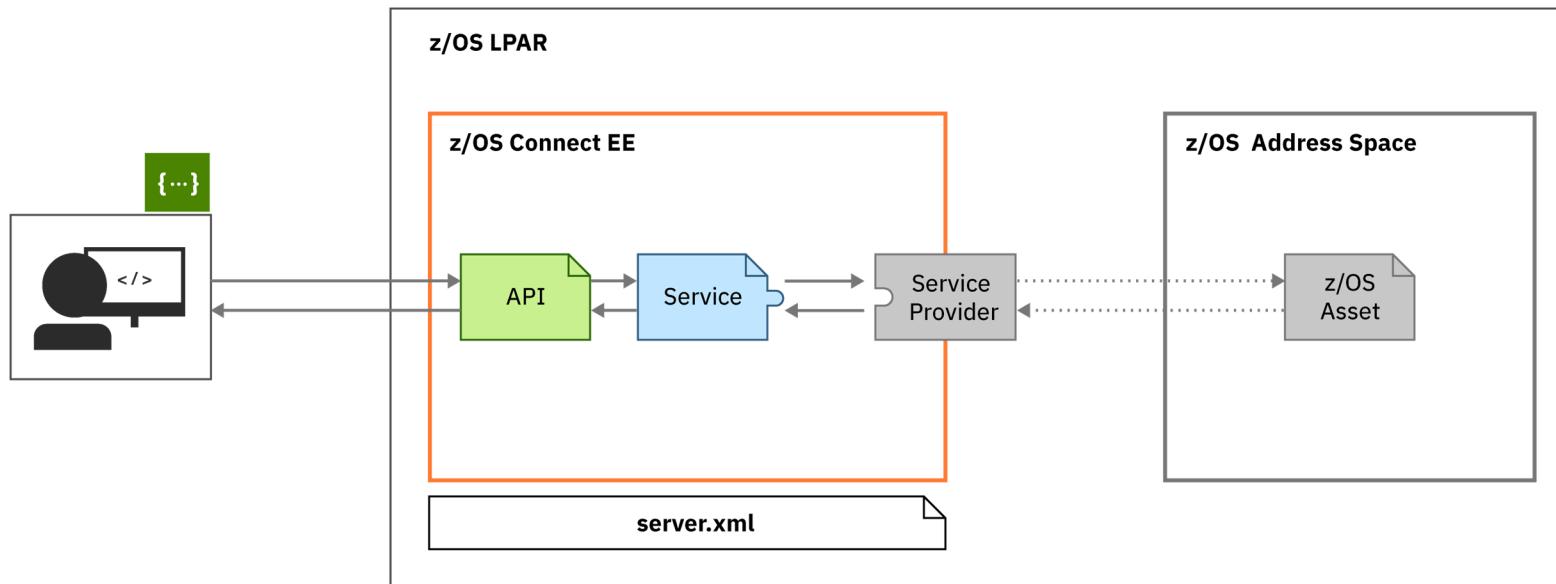
The screenshot shows the z/OS Connect Designer interface. It displays the configuration for an OPENAPI 3 endpoint named 'cscvinc'. The 'Paths' section shows '/employee' (1) and '/employee/{employee}' (3). The 'GET' method is selected. A flowchart shows a decision point 'Else send default response' leading to an 'API 404 Not Found' node. Below this, a '404 - Not Found' section shows mapping rules for the 'Body' field, specifically mapping 'Employee not found -> CICS EIBCODE: CEIBRESP -> EIBRESP2: CEIBRESP2'.

Web browser - z/OS Connect Designer



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)

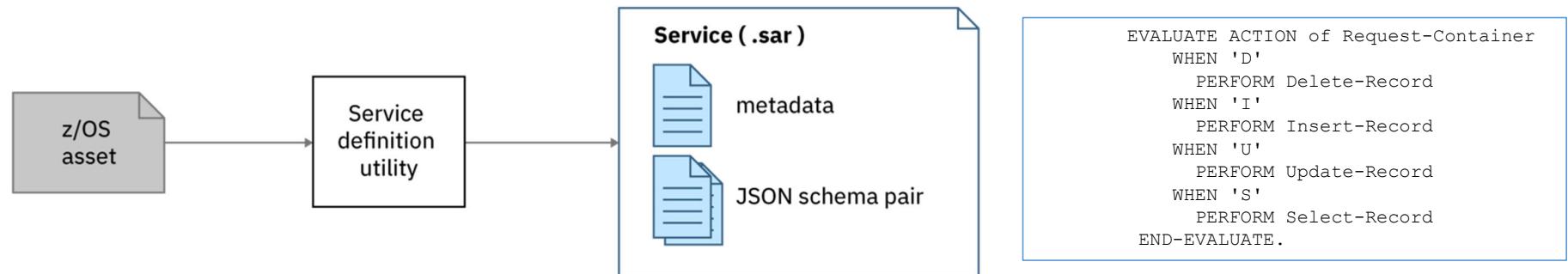


Describing the interaction (service) with the z/OS resource (OpenAPI 2)

Start by creating a service to represent an interaction with the 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



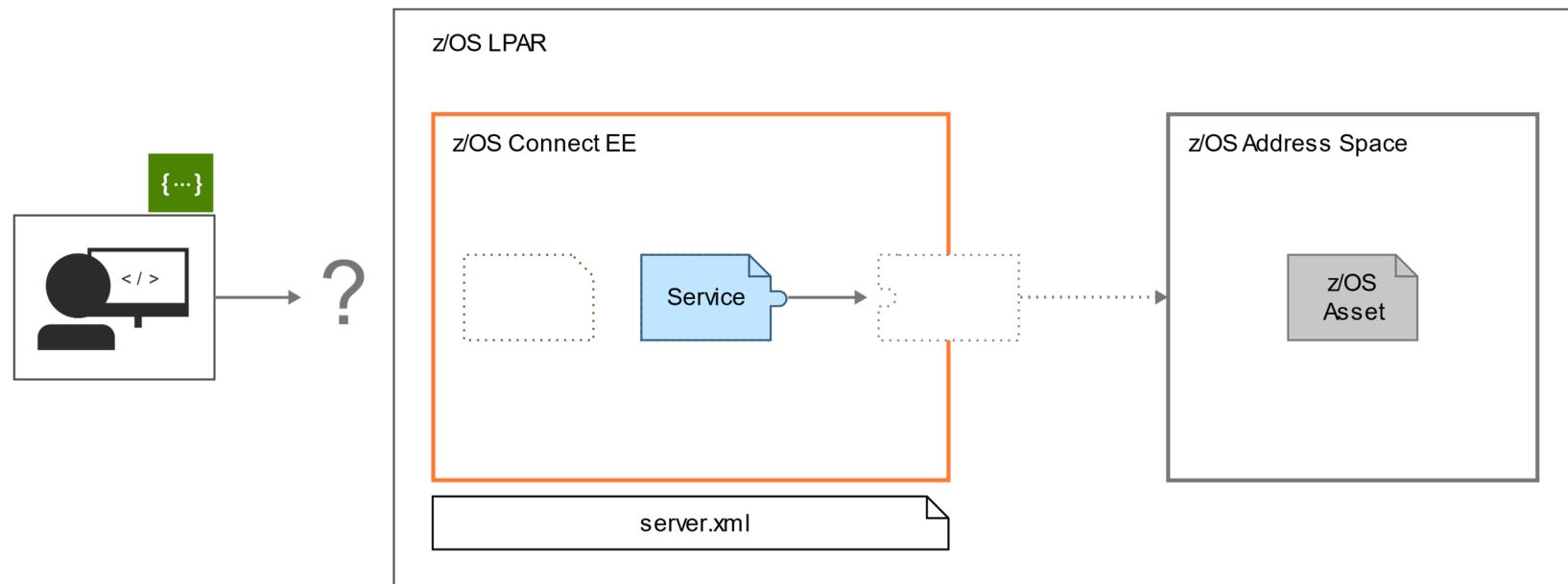
Use a system-appropriate utility to generate a service archive file for the z/OS application

- 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

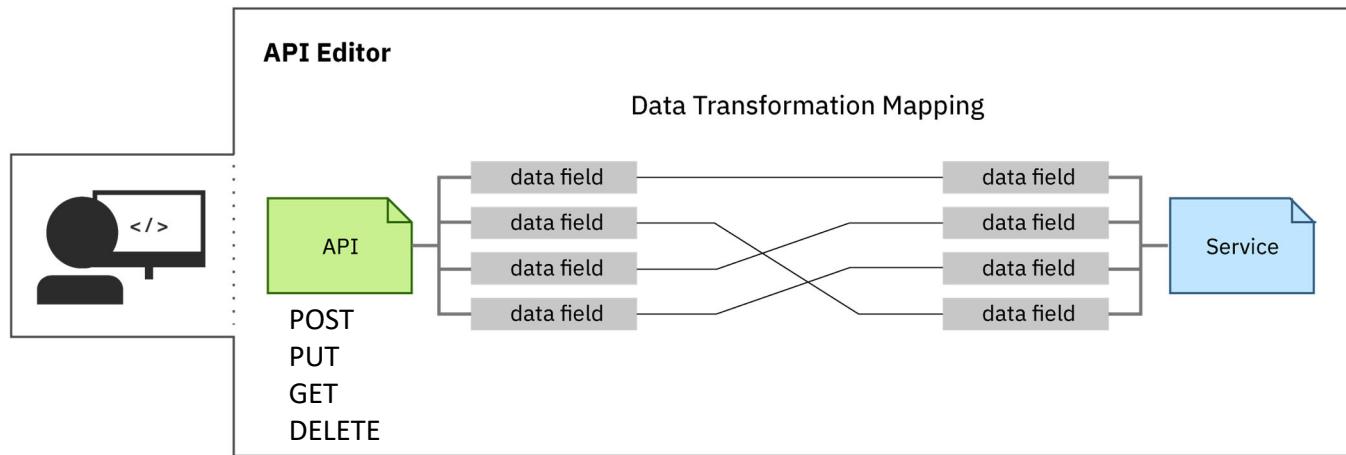


Deploy the service archive file generated in **Step 1** using the right-click deploy in **the API toolkit**.

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

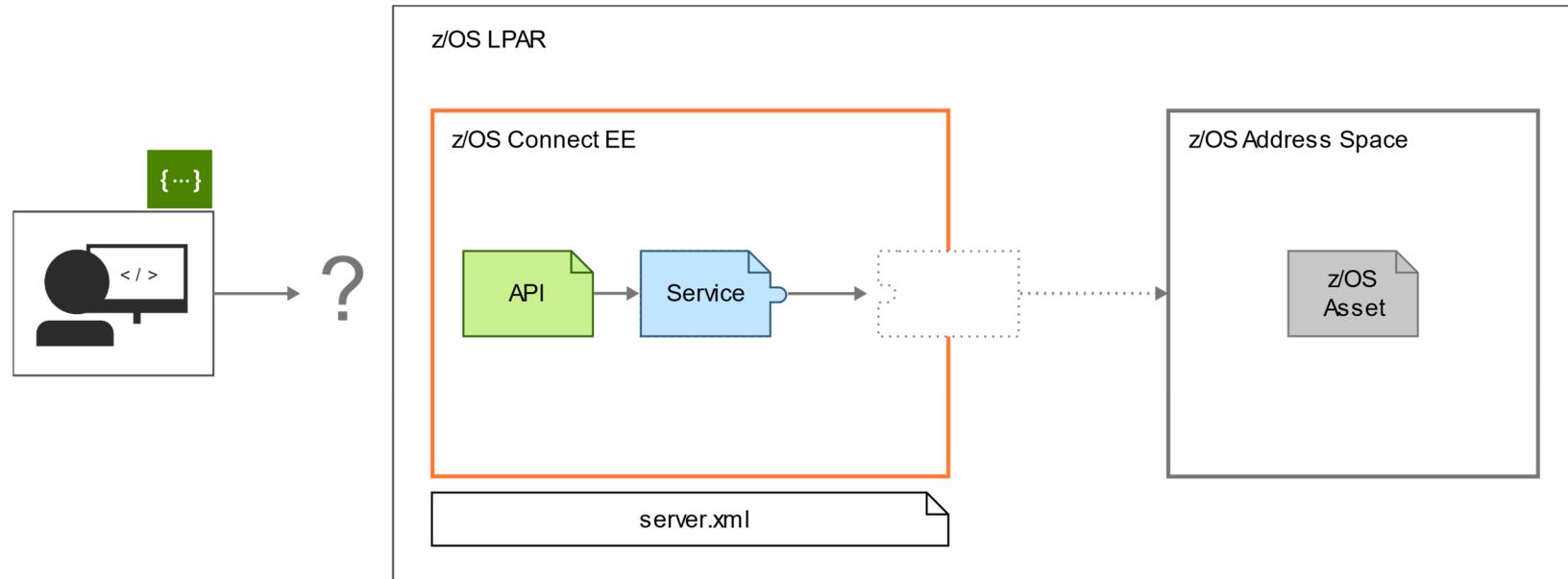


- Import the service archive file into the **API toolkit** and start designing the RESTful API.
- Provides additional data mapping
- Use the editor to describe the API and how it maps to underlying services.



Deploy the API (Open API 2)

Deploy the API archive file

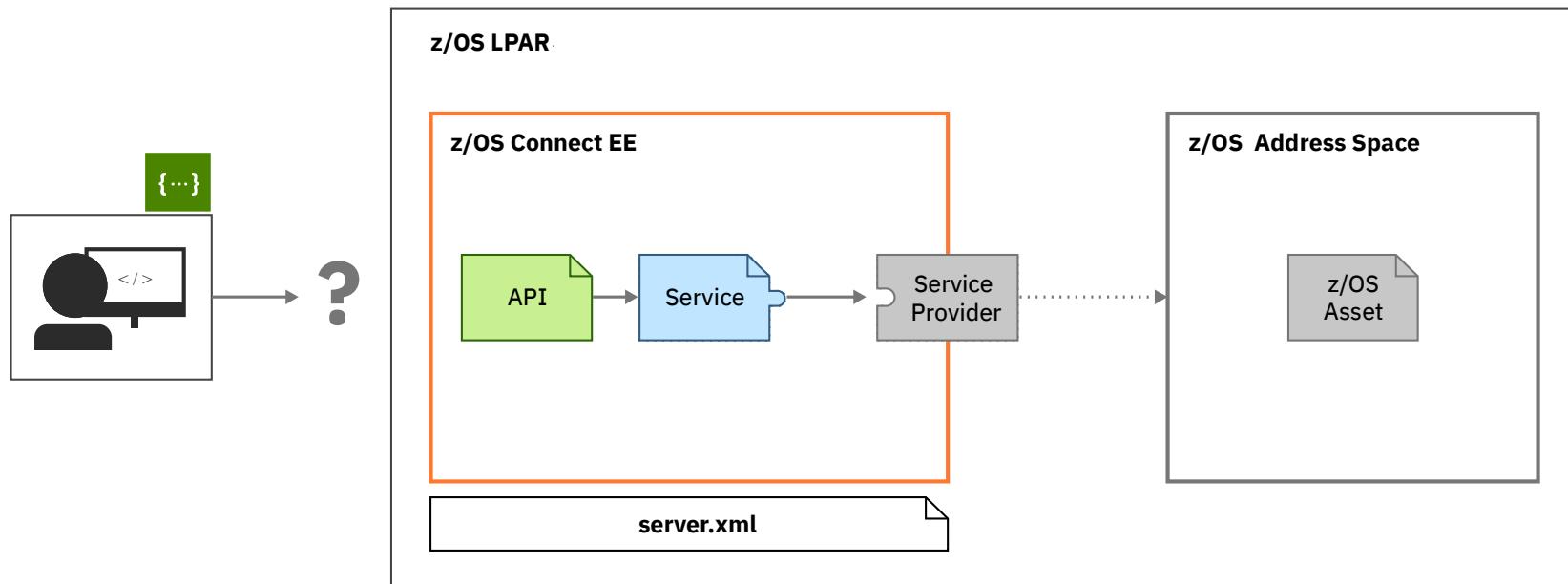


Deploy your API using the right-click deploy in **the API toolkit**



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

Configure the service provider

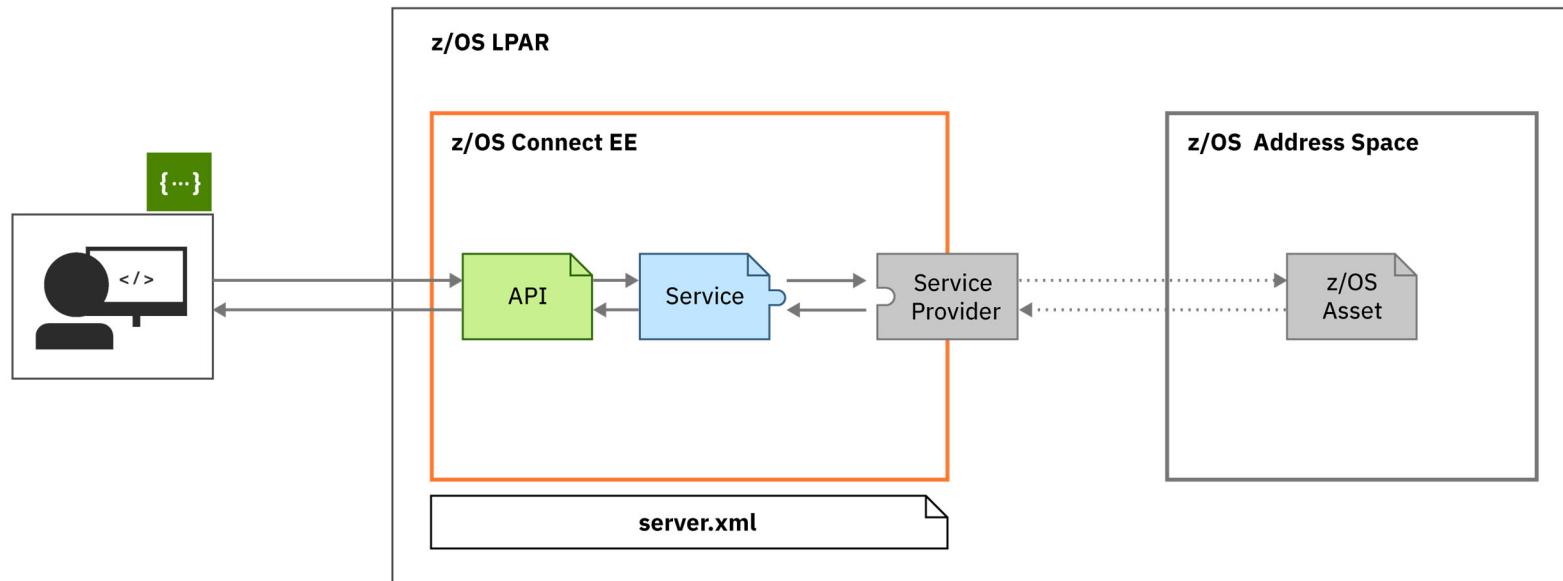


Configure the system-appropriate service provider to connect to your backend system in your **server.xml**.



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

Done

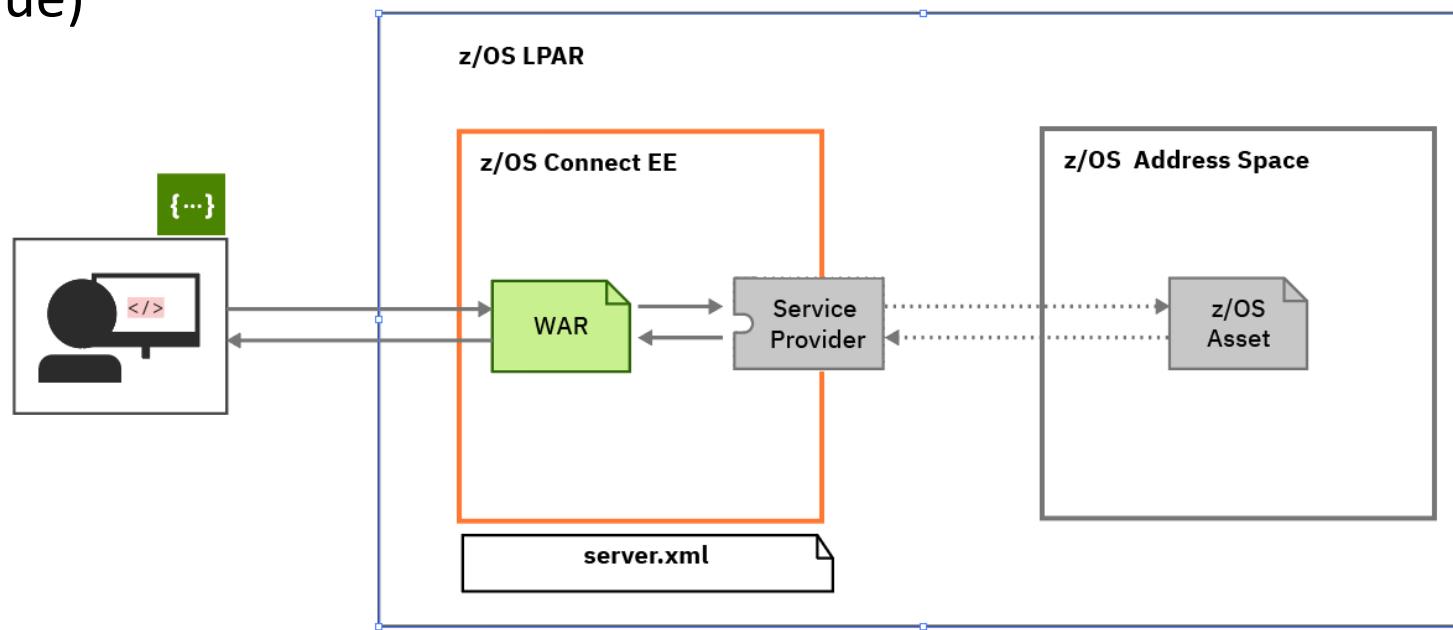


- The API is ready to be consumed and requires no knowledge that a z/OS resource is being accessed
- The Service provides meta data specific to the z/OS Asset (e.g., CICS program, MQ queue manager, etc.)
- The Service Provider is tightly coupled to a specific instance of a resource (e.g., host and port,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



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)



Results: the client code is totally unaware of the z/OS infrastructure

CICS

```

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

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

```

Problems @ Javadoc Declaration Console Coverage

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

URL: https://wg31.washington.ibm.com:9453/cscvinc/employee/22222

USERID: CICSUMER

CEIBRESP0: 0

CEIBRESP2: 0

name: DR E. GRIFFITHS

employeeNumber: 22222

amount: \$0022.00

address: FRANKFURT, GERMANY

phoneNumber: 20034151

date: 26 11 81

Response Message: {"cscvincSelectServiceOperationResponse": {"cscvincContainer": {"response": {"CEIBRESP0": "0", "USERID": "CICSUMER", "CEIBRESP2": "0", "name": "DR E. GRIFFITHS", "employeeNumber": "22222", "amount": "\$0022.00", "address": "FRANKFURT, GERMANY", "phoneNumber": "20034151", "date": "26 11 81"}}}

Db2

```

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

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

```

Problems @ Javadoc Declaration Console Coverage

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

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

Employee Number: 000010

First Name : CHRISTINE

Last Name: HAAS

Middle Initial: I

Phone Number: 3204

IMS

```

53
54
55
56
57
58
59
60
61

URL url = new URL("https://wg31.washington.ibm.com:9453/phonebook/contacts/" + args[1]);
System.out.println("URL: " + url);
HttpURLConnection conn = (HttpURLConnection) url.openConnection();
conn.setRequestMethod("GET");
conn.setRequestProperty("Content-Type", "application/json");
byte[] bytesEncoded = Base64.encodeBase64(args[0].getBytes());
conn.addRequestProperty("Authorization", "Basic " + new String(bytesEncoded));

```

Problems @ Javadoc Declaration Console Coverage

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

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

lastName: LAST1

firstName: FIRST1

zipCode: D01/R01

extension: 8-111-1111

message: ENTRY WAS DISPLAYED

HTTP code: 200

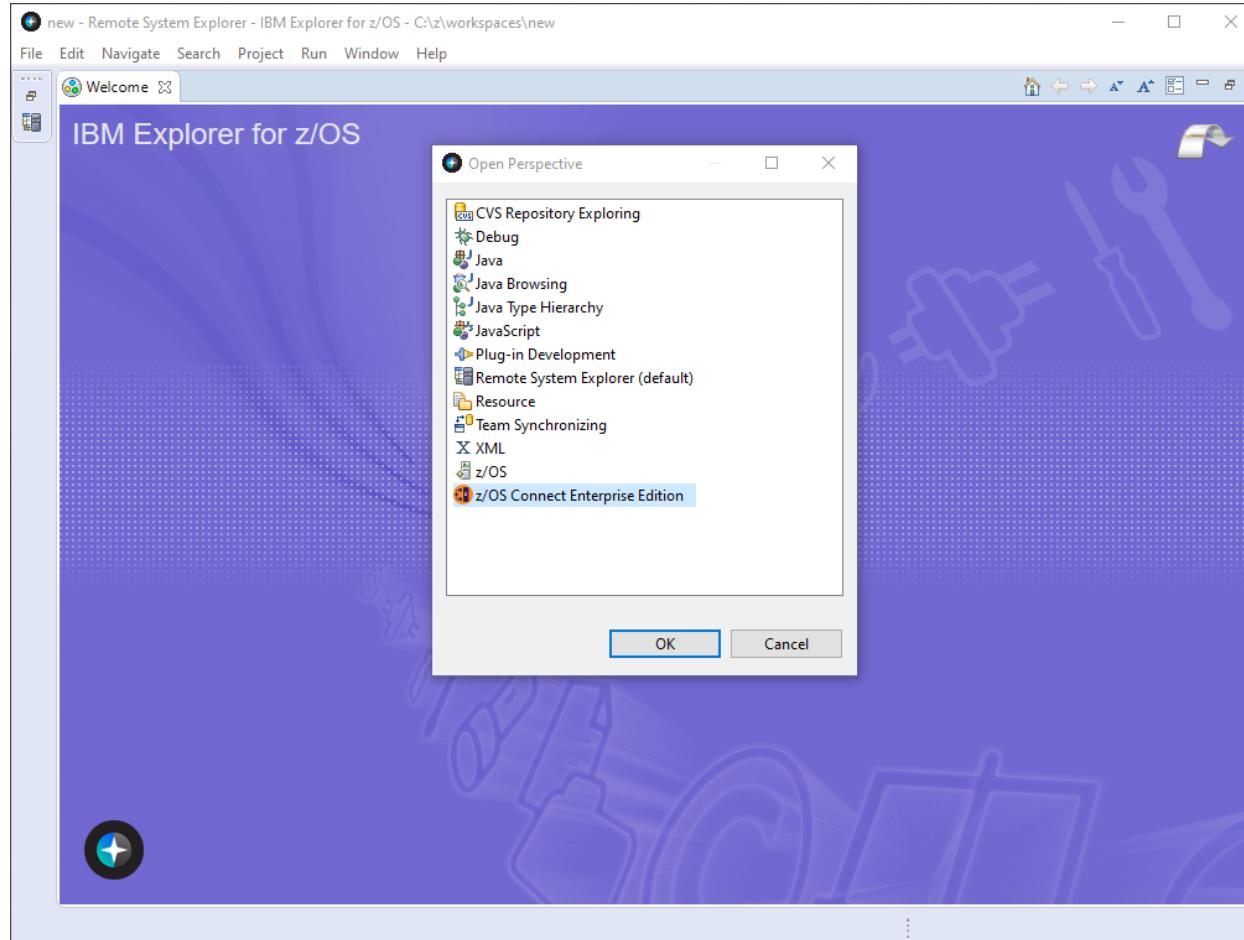


/api_toolkit/services

Simple service creation for OpenAPI 2 APIs



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

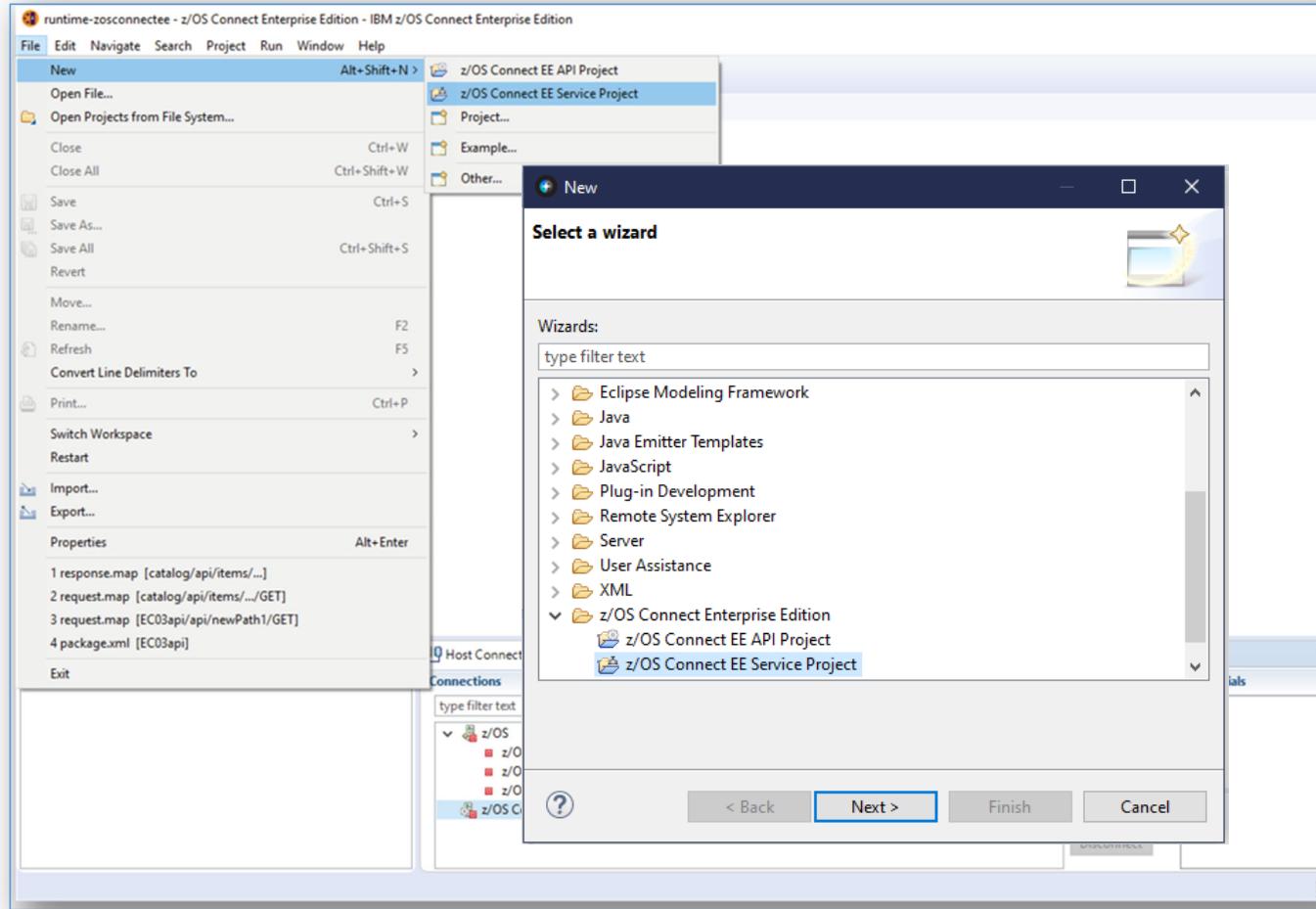


Use the **API toolkit** to create services through Eclipse-based tooling.

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



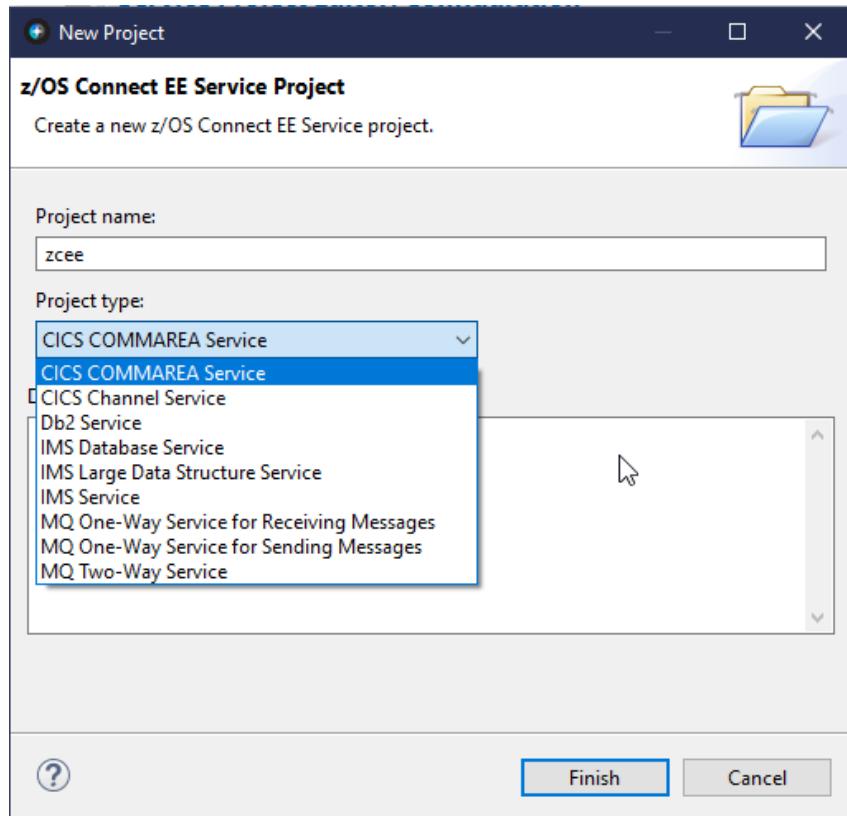
Use the **API toolkit** to create services through Eclipse-based tooling.

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



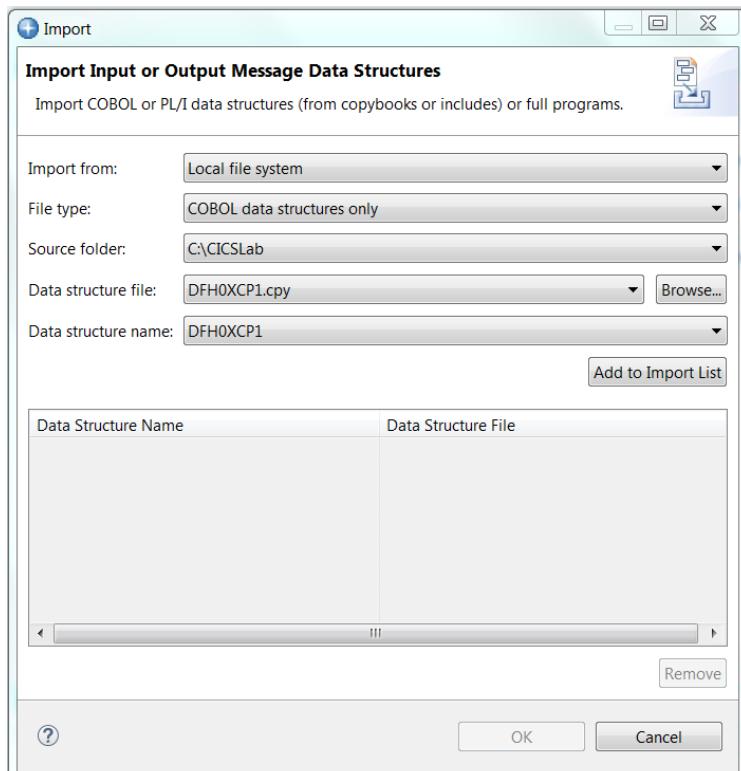
A common interface for service creation, irrespective of back-end subsystem.

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



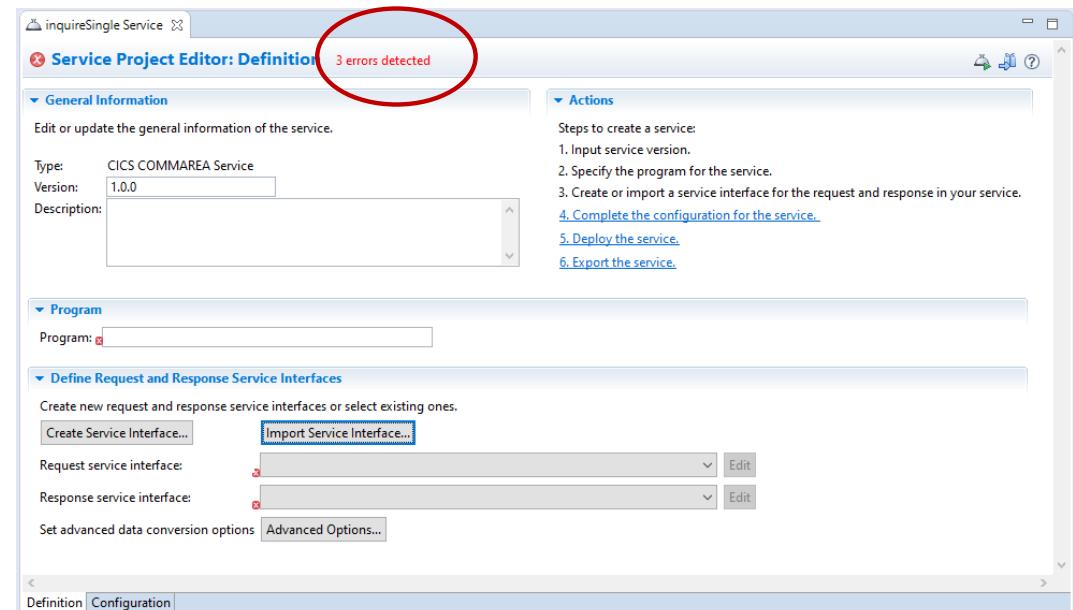
API toolkit – Creating Services for CICS, IMS TM and MQ

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



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

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





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

Allows editing a request service interface definition

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

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

The screenshot shows a software interface titled "Service Interface Definition". It displays a table of fields with columns: Fields, Include, Interface rename, Default Field Value, Data Type, Field Length, and Start Byte. The table lists several fields under the "DFHXCPI" structure, including CA_REQUEST_ID, CA_RETURN_CODE, CA_RESPONSE_MESSAGE, CA_REQUEST_SPECIFIC, CA_INQUIRE_REQUEST, CA_INQUIRE_SINGLE, CA_ITEM_REF_REQ, CA_SINGLE_ITEM, and CA_ORDER_REQUEST. A red circle highlights the "Default Field Value" column for CA_REQUEST_ID, which is set to "01INQS". A red box highlights the "Interface rename" column for CA_INQUIRE_SINGLE, which is set to "inquireSingle". The "Data Type" column shows various types like CHAR, DECIMAL, STRUCT, and CHAR. The "Field Length" and "Start Byte" columns provide specific details for each field.

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



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

And editing a response message service interface definition

*inquireSingleResponse

Service Interface Definition

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

Search:

Fields	Include	Interface rename	Default Field Value	Data Type	Field Length	Start Byte
COMAREA	<input type="checkbox"/>					
DFH0XCP1	<input checked="" type="checkbox"/>					
CA_REQUEST_ID	<input checked="" type="checkbox"/>	CA_REQUEST_ID		CHAR	6	1
CA_RETURN_CODE	<input checked="" type="checkbox"/>	returnCode		DECIMAL	2	7
CA_RESPONSE_MESSAGE	<input checked="" type="checkbox"/>	responseMessage		CHAR	79	9
CA_REQUEST_SPECIFIC (Redefines CA_INQUIRE_REQUEST)	<input type="checkbox"/>	CA_REQUEST_SPECIFIC		CHAR	911	88
CA_INQUIRE_REQUEST redefines CA_INQUIRE_SINGLE	<input type="checkbox"/>	CA_INQUIRE_REQUEST		STRUCT	911	88
CA_INQUIRE_SINGLE redefines CA_INQUIRE_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 checked="" type="checkbox"/>	singleItem		STRUCT	60	99
CA_SNGL_ITEM_REF	<input checked="" type="checkbox"/>	itemReference		DECIMAL	4	99
CA_SNGL_DESCRIPTION	<input checked="" type="checkbox"/>	description		CHAR	40	103
CA_SNGL_DEPARTMENT	<input checked="" type="checkbox"/>	department		DECIMAL	3	143
CA_SNGL_COST	<input checked="" type="checkbox"/>	cost		CHAR	6	146
IN_SNGL_STOCK	<input 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_INQUIRE_SINGLE	<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 the Service Interface Editor with two tabs: "cscvincSelectService Service" and "cscvincSelectRequest". The "cscvincSelectRequest" tab is active. It displays a table of fields with columns: Fields, Include, Interface Rename, Default Field Value, Data Type, and Field Length. A red circle highlights the "Default Field Value" column for the "ACTION" row, which is set to "S".

The screenshot shows the Service Project Editor: Definition. Under the "Program" section, the program name is "CSCVINC", highlighted with a red circle. The "Actions" section lists steps for creating a service, and the "Definition" tab is selected.

```
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.
```

mitchj@us.ibm.com

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



Accessing a CICS program – Transaction ID Usage

cscvincSelectService Service

Service Project Editor: Configuration

Required Configuration

Enter the required configuration for this service.

Coded character set identifier (CCSID):

Connection reference:

Optional Configuration

Enter the optional configuration for this service.

Transaction ID:

Transaction ID usage:

Bidi configuration reference:

Use context containers:

Context containers HTTP headers:

Add another

Definition **Configuration**

EIB_ONLY

```

WG31 - 3270
File Edit Settings View Communication Actions Window Help
TRANSACTION: CSMI PROGRAM: DFHMIRS TASK: 0008501 APPLID: CICS53Z DISPLAY: 00
STATUS: PROGRAM INITIATION

EIBTIME = 104730
EIBDATE = 0122050
EIBTRNID = 'CSMI'
EIBTRSKN = 8501
EIBTRMID = '/RBX'

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

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

WG31 - 3270
File Edit Settings View Communication Actions Window Help
TRANSACTION: MIJO PROGRAM: DFHMIRS TASK: 0008476 APPLID: CICS53Z DISPLAY: 00
STATUS: ABOUT TO EXECUTE COMMAND

EXEC CICS LINK PROGRAM
PROGRAM ('CSCVINC')
SYNCONRETURN
CHANNEL ('Channel')
NOHANDLE

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

WG31 - 3270
File Edit Settings View Communication Actions Window Help
TRANSACTION: MIJO PROGRAM: CSCVINC TASK: 0008837 APPLID: CICS53Z
STATUS: PROGRAM INITIATION

EIBTIME = 181730
EIBDATE = 0122051
EIBTRNID = 'MIJO'
EIBTRSKN = 8837
EIBTRMID = '/nrx'

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

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

```

EIB_AND_MIRROR

```

WG31 - 3270
File Edit Settings View Communication Actions Window Help
TRANSACTION: MIJO PROGRAM: DFHMIRS DFH
STATUS: PROGRAM INITIATION

EIBTIME = 109914
EIBDATE = 0122050
EIBTRNID = 'MIJO'
EIBTRSKN = 8492
EIBTRMID = '/RBX'

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

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

WG31 - 3270
File Edit Settings View Communication Actions Window Help
TRANSACTION: MIJO PROGRAM: DFHMIRS DFH
STATUS: ABOUT TO EXECUTE COMMAND

EXEC CICS LINK PROGRAM
PROGRAM ('CSCVINC')
SYNCONRETURN
TRANSID ('MIJO')
CHANNEL ('Channel')
NOHANDLE

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

WG31 - 3270
File Edit Settings View Communication Actions Window Help
TRANSACTION: MIJO PROGRAM: CSCVINC TASK: 0008949 APPLID: CICS53Z
STATUS: PROGRAM INITIATION

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

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

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

```

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

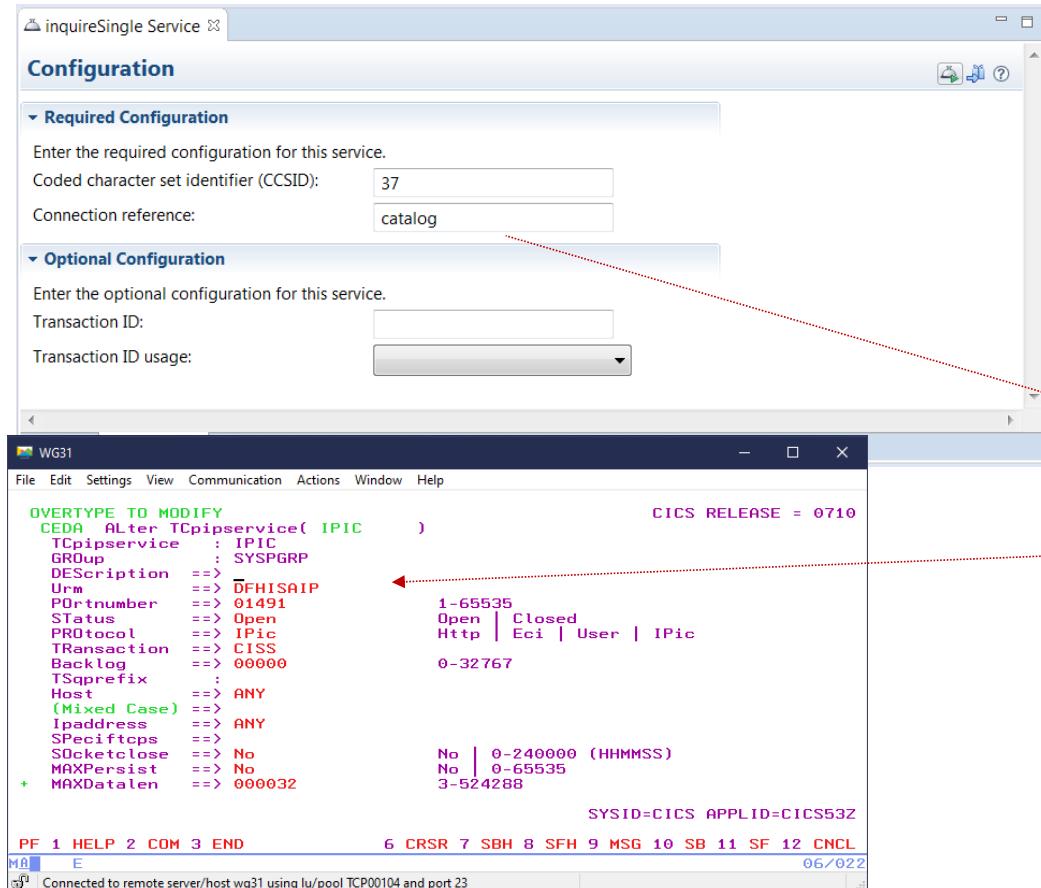
mitchj@us.ibm.com

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



Accessing a CICS program using IPIC

The server.xml file is the key configuration file:



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

catalog.xml

Design Source

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

Define IPIC connection to CICS



API toolkit – Creating Services for IMS

Creating a “GET” service interface request definition

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

ivtnoDisplayRequest

Service Interface Editor

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

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

The service developer creates distinct services for each function.

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

ivtnoDisplayService Service

Service Project Editor: Configuration

Required Configuration

Enter the required configuration for this service.

Connection profile: **IMSCONN**

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

Optional Configuration

Enter the optional configuration for this service.

IMS destination override:

Program name:

IMS Connections and Interactions (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

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, PORTTMOT=6000) , ODBMTMOT=6000)
```

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>
```



API toolkit – Creating Services for IMS DB

Creating a service project from the IMS Catalog

The screenshot shows the 'Service Project Editor: Definition' window for a service named 'selectByName Service'. The 'General Information' panel shows 'Type: IMS Database Service', 'Version: 1.0.0', and an empty 'Description' field. The 'Actions' panel lists steps: 1. Input service version, 2. Specify the SQL command for the service, 3. Specify Database Connection Properties and Generate Service Interface, 4. Complete the configuration for the service, 5. Deploy the service, and 6. Export the service. The 'Enter or import SQL Command' panel contains the SQL query: 'SELECT FIRSTNAME, ZIPCODE, PHONENBR, A1111111 FROM ATSVPA.A1111111 WHERE A1111111=?'. This query is circled in red. The 'Specify Database Connection Properties and Generate Service Interface' panel includes fields for 'Database Connection: wg31:5555' and 'Database Name: DFSIVPA', with a 'Generate Service Interface...' button. At the bottom are tabs for 'Definition' and '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=DFSIVD1,DEVICE=3380,SIZE=2048  
SEGM NAME=A1111111,PARENT=0,BYTES=40,RULES=(LLV,LAST),  
PTR=(TB,CTR)  
FIELD NAME=(A1111111,SEQ,U),BYTES=010,START=00001,TYPE=C  
FIELD NAME=FIRSTNAME,BYTES=010,START=00011,TYPE=C  
FIELD NAME=PHONENBR,BYTES=010,START=00021,TYPE=C  
FIELD NAME=ZIPCODE,BYTES=7,START=00031,TYPE=C  
LCHILD NAME=(A1,IVPDB1I),POINTER=INDX,RULES=LAST  
DBDGEN  
FINISH  
END
```



API toolkit – Creating Services for IMS DB

The Toolkit allows editing a service interface definitions*

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

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

*Using a slightly different process



IMS Connection Factory in the server XML

Service Project Editor: Configuration

Required Configuration

Enter the required configuration for this service.

Connection profile:

ConnectionFactory

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

IMS Connect HWSCFG

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



API toolkit – Creating Services for MQ

Creating a “POST” service interface definition

The screenshot displays two windows from the API toolkit:

- Service Interface Editor:** Shows a table of fields for a service interface named "minilonServiceRequest". The table includes columns for Fields, Include, Interface Rename, Default Field Value, and Data Type. A red box highlights the "Interface Rename" column for the "loan application" row, which contains the value "loan application". Another red box highlights the "Default Field Value" column for the same row, which contains the value "F". A circled red box highlights the "Include" column for the "NAME" field, which has a checked checkbox.
- Service Project Editor: Configuration:** Shows configuration settings for a service named "twoway Service". The "Required Configuration" section is expanded. A red circle highlights the "Request destination JNDI name" field, which contains "jms/requestQueue". Other fields include "Connection factory JNDI name: jms/qmgrCf", "Reply destination JNDI name: jms/replyQueue", "Wait interval: 3000", "MQMD format: MQSTR", "Coded character set identifier (CCSID): 37", "Is message persistent: false", "Reply selection: msgIDToCorrelID", and "Expiry: -1".

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



Using JMS to access MQ (One-Way)

mqGetService Service

Service Project Editor: Configuration

Required Configuration

Enter the required configuration for this service.

Connection factory JNDI name: jms/qmgrCf

Destination JNDI name: jms/default

Coded character set identifier (CCSID): 37

Optional Configuration

Enter the optional configuration for this service.

Wait interval:

Message selector:

Definition Configuration

mqClient.xml

Read only Close

Design Source

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



Using JMS to access MQ (Two-Way)

*twoWay Service X

Service Project Editor: Configuration

Required Configuration

Enter the required configuration for this service.

Connection factory JNDI name:

Request destination JNDI name:

Reply destination JNDI name:

Wait interval:

MQMD format:

Coded character set identifier (CCSID):

Is message persistent:

Reply selection:

Expiry:

Definition Configuration

mq.xml

Read only Close

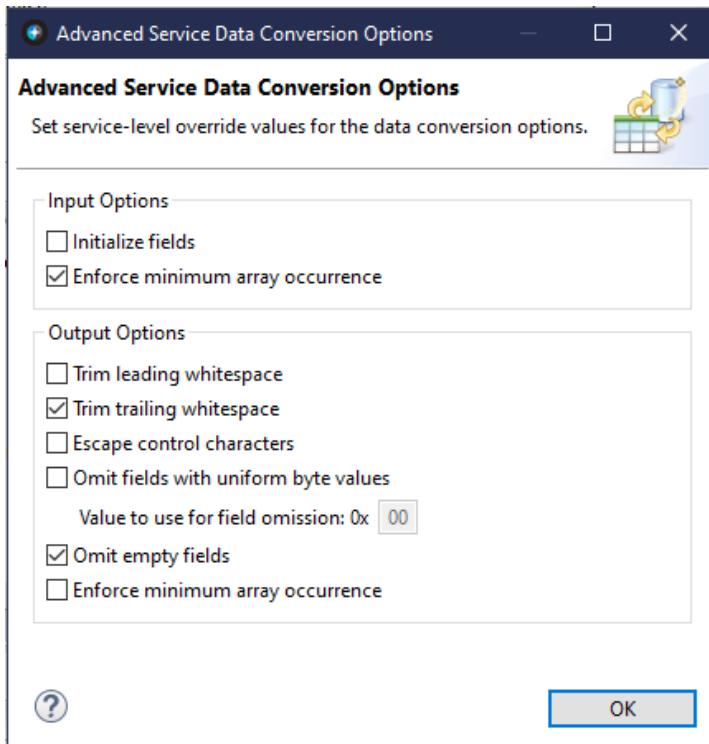
Design Source

```
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.MQZEE.QUEUE"
30     CCSID="37"/>
31 </jmsQueue>
32
33 <jmsQueue id="requestQueue" jndiName="jms/request">
34   <properties.wmqJms
35     baseQueueName="ZCONN2.TRIGGER.REQUEST"
36     targetClient="MQ"
37     CCSID="37"/>
38 </jmsQueue>
39
40 <jmsQueue id="replyQueue" jndiName="jms/replyQueue">
41   <properties.wmqJms
42     baseQueueName="ZCONN2.TRIGGER.RESPONSE"
43     targetClient="MQ"
44     CCSID="37"/>
45 </jmsQueue>
46
47
```

mitchj@us.ibm.com



API toolkit – Advanced Data Conversion Options



Request Messages:

- Initialize fields
- Enforce minimum array occurrence

Response Messages:

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



API toolkit – Creating Services for Db2

Creating a service project from Db2 REST service

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

Import Db2 service from service manager

Db2 service manager connection: wg31:2446

Type to search...

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

Definition Configuration

Import Cancel

*selectEmployee Service

Service Project Editor: Definition

General Information

Edit or update the general information of the service.

Type: Db2 Service
Version: 1.0.0
Description:

Actions

Steps to create a service:

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

Define Db2 service

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

Import from Db2 service manager...

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

Import from local machine...

The service developer retrieves details about the Db2 REST services

Note there is no service interface editor available

Accessing a Db2 REST service resource



Screenshot of the Service Project Editor: Configuration window for the "selectEmployee Service".

The left pane shows the configuration details:

- 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

The right pane shows the XML configuration file "db2pass.xml" with the "Source" tab selected:

```
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
```

Red arrows point from the "db2conn" connection reference in the left pane to the "basicAuthRef" attribute in the XML code, and from the "WG31.WASHINGTON.IBM.COM" host entry in the left pane to the "host" attribute in the XML code.



API toolkit – Services Editor

Server connection and Services deployment

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

The screenshot shows the API toolkit interface. On the left, the 'Host Connections' view is open, displaying a list of server connections. A context menu is open over the 'z/OS Connect EE' connection, with options like 'Deploy Service to z/OS Connect EE Server' and 'Export z/OS Connect EE Service Archive'. To the right, two dialog boxes are displayed: 'Deploy Service to z/OS Connect EE Server Result' and 'Export Service Package'. The 'Deploy Service' dialog shows deployment results for a service named 'cscvincDeleteService' (Version 1.0.0) to 'z/OS Connect EE Server: wg31:9453', indicating success. The 'Export Service Package' dialog allows selecting a workspace or local file system for exporting a service package named 'cscvincDeleteService.sar'.

Deploy Service to z/OS Connect EE Server Result

z/OS Connect EE Server: wg31:9453

Deployment results:

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

All services were deployed successfully.

OK

Export Service Package

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

Export service package to:

Workspace

Local file system

Folder: /Services

File name: cscvincDeleteService.sar

Overwrite service package file

OK Cancel



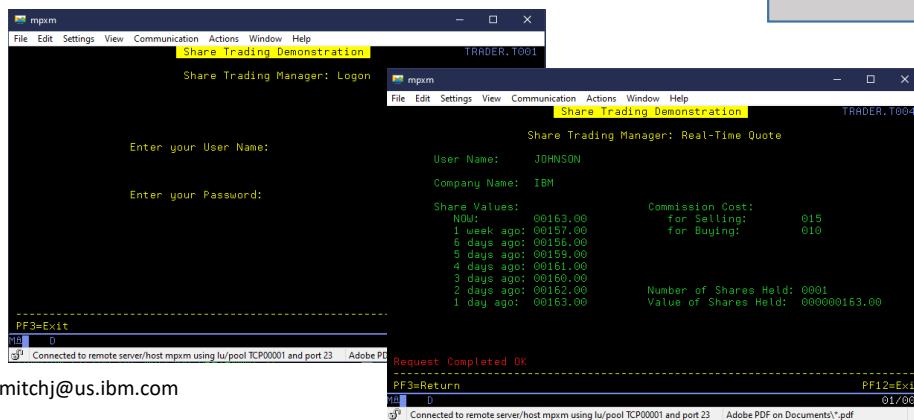
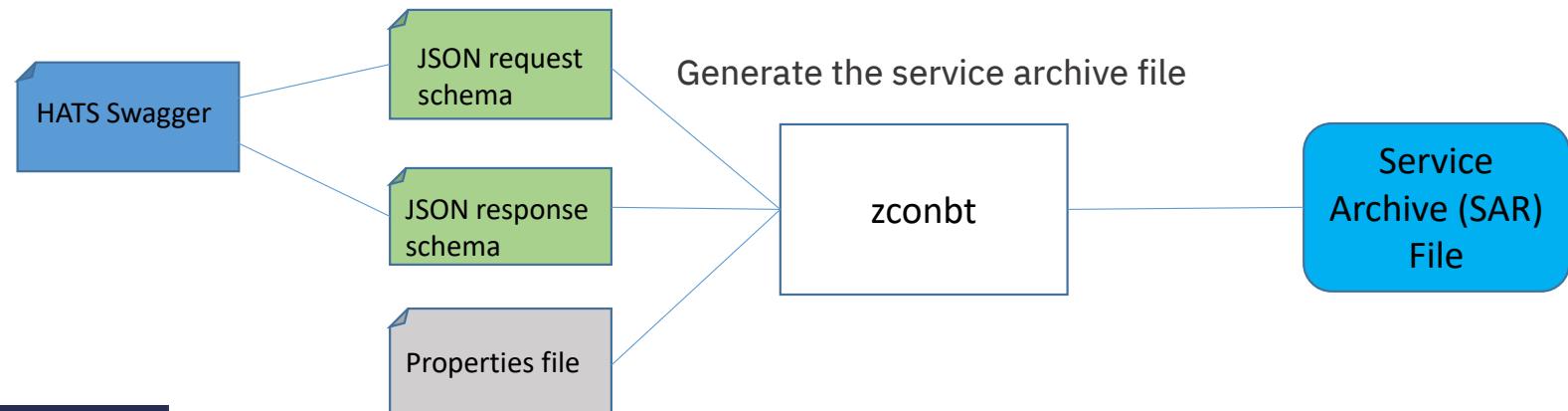
/api_toolkit/services

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



Command line(zconbt) – REST Services

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



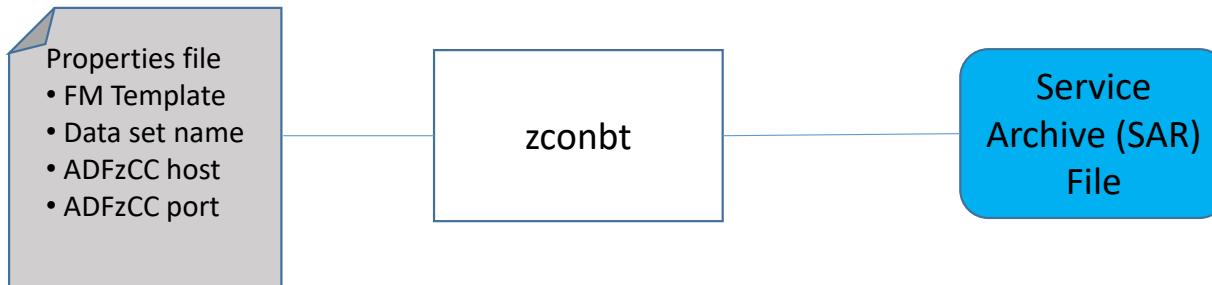
```
provider=rest  
name=getCompany  
version=1.0  
description=Obtain a list of companies  
requestSchemaFile=getCompanyRequest.json  
responseSchemaFile=getCompanyResponse.json  
verb=POST  
uri=/Trader/rest/GetCompany  
connectionRef=HatsConn
```



Command line(zconbt) – File Manager

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

Generate the service archive file



```
WG31 - 3270
File Edit Settings View Communication Actions Window Help
Process Options Help
File Manager Copybook and Template Utility functions
Command ==> -
1 Workbench Create, edit or update single templates
2 Print View or print copybooks or templates
3 Import Convert existing template(s) into template(s)
4 Update Update template(s)
5 Import Import template(s)
6 Export Export template(s)
7 Repository Repository utilities menu

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

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



Deploying Service Archive files (required those not developed in Eclipse)

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

The screenshot shows the Postman application interface. At the top, there are tabs for File, Edit, View, Help, New, Import, Runner, and My Workspace. Below the tabs, there's a search bar and a sign-in button. The main area shows an 'Untitled Request' with a POST method selected. The URL is https://wg31.washington.ibm.com:9453/zosConnect/services. In the 'Body' tab, the content type is set to 'binary'. A file named 'selectEmployee.sar' is attached. The 'Pretty' tab displays the JSON representation of the SAR file content.

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

Command:

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

Results:

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

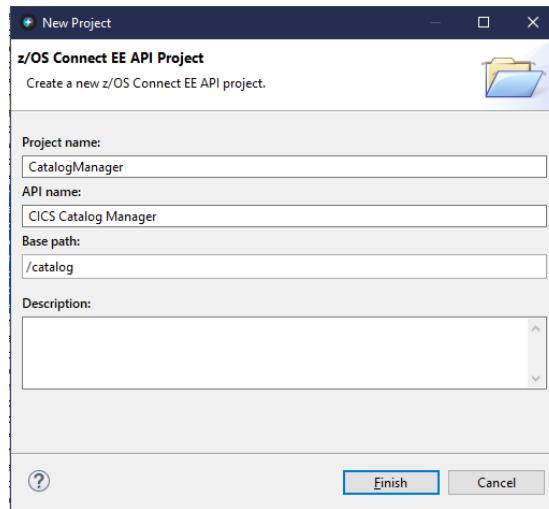


Once we have a Service Archive (SAR) What's next?

Quick and easy **API mapping**.

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

API toolkit – API Editor



Describe your API

Name: CICS Catalog Manager Description:

Base path: /catalog

Version: 1.0.0

Contact Information

Path

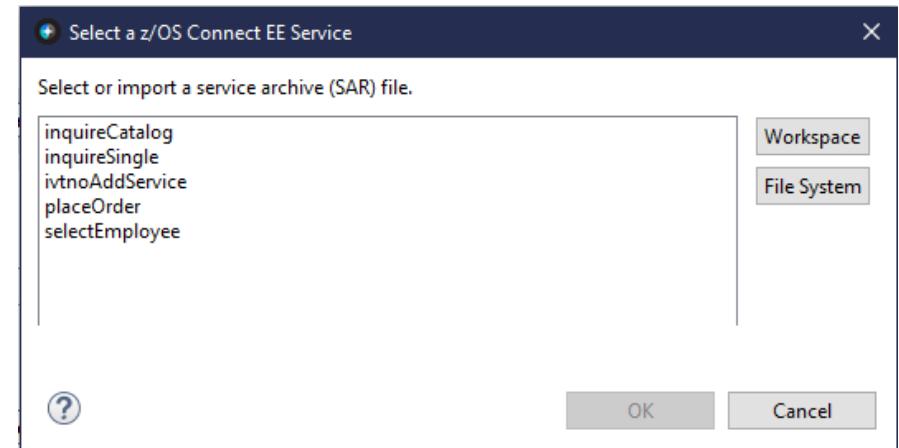
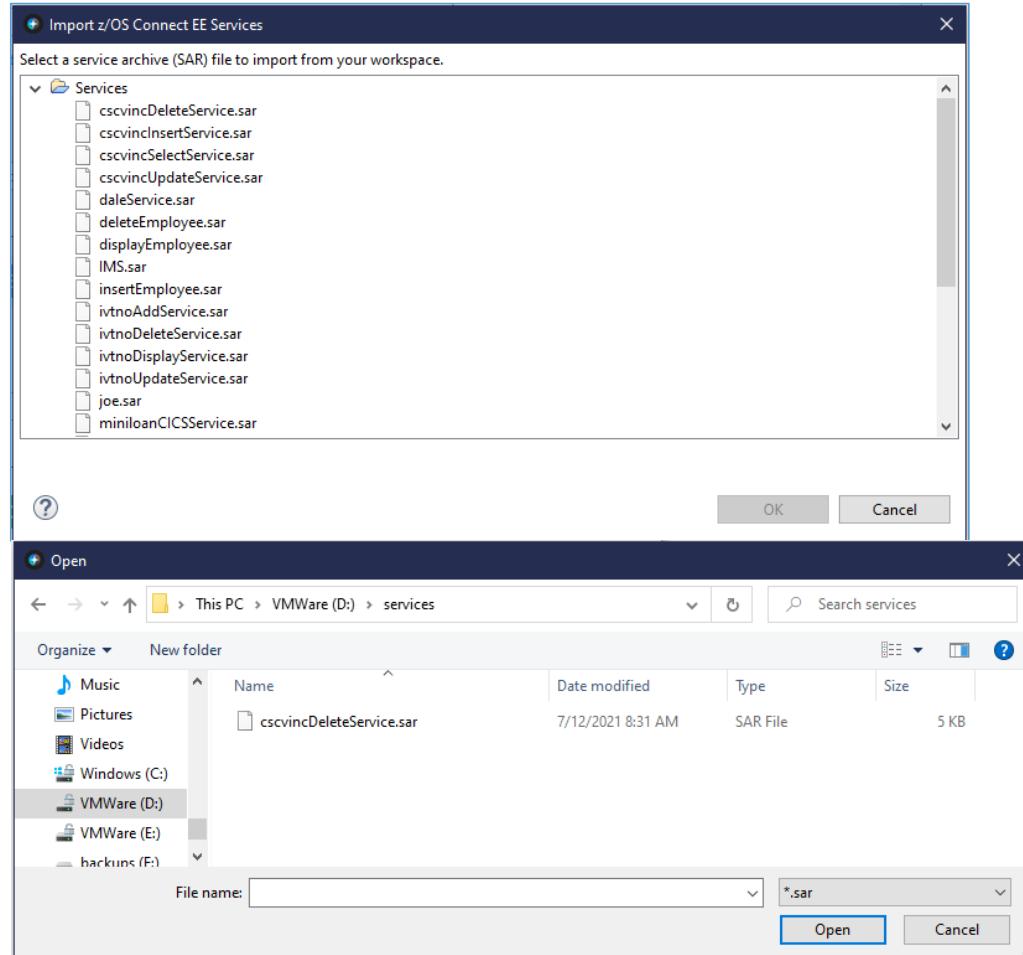
/newPath1

Methods (4)

Method	Action	Service...	Mapping...	Up	Down	X
POST						
GET						
PUT						
DELETE						



Importing the service archives files



mitchj@us.ibm.com



API toolkit – API Editor

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

- catalog**: Path `/catalog`, Version 1.0.0. Methods: GET `inquireCatalog`, PUT `ivtnoAddService`.
- order**: Path `/order`. Methods: POST `placeOrder`, PUT `selectEmployee`.
- item**: Path `/item/{itemID}`. Methods: GET `inquireSingle`.

A large red oval highlights the `item` endpoint.

mitchj@us.ibm.com

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

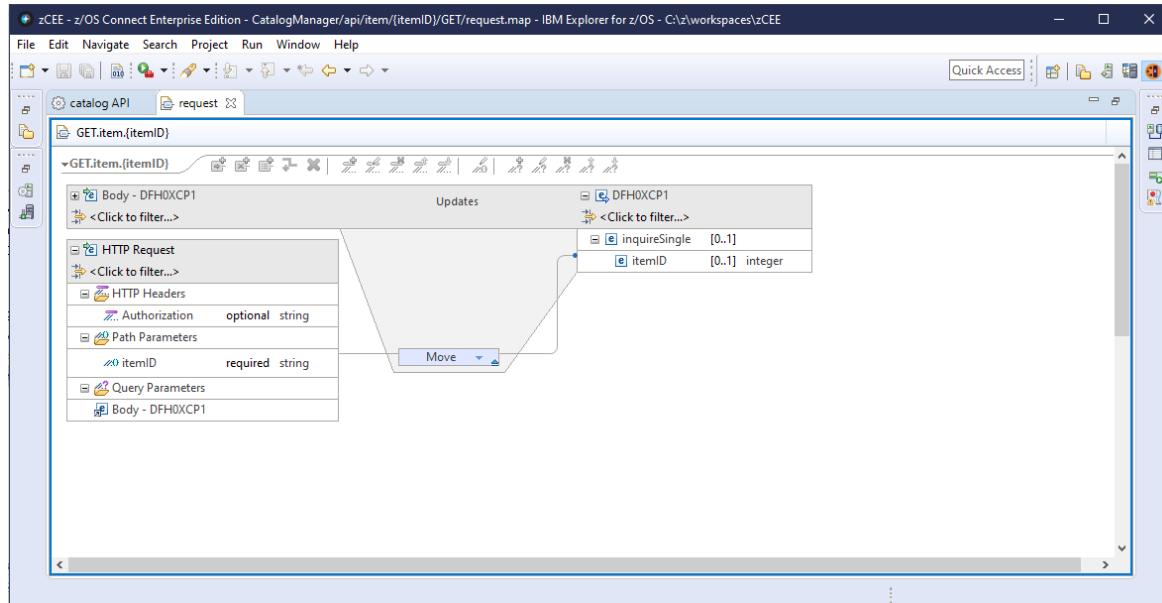
Once you define your API, you can map backend services to each request.

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



API toolkit – API Editor

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



Map both the request and response for each API.

Map path and query parameters to native data structures.

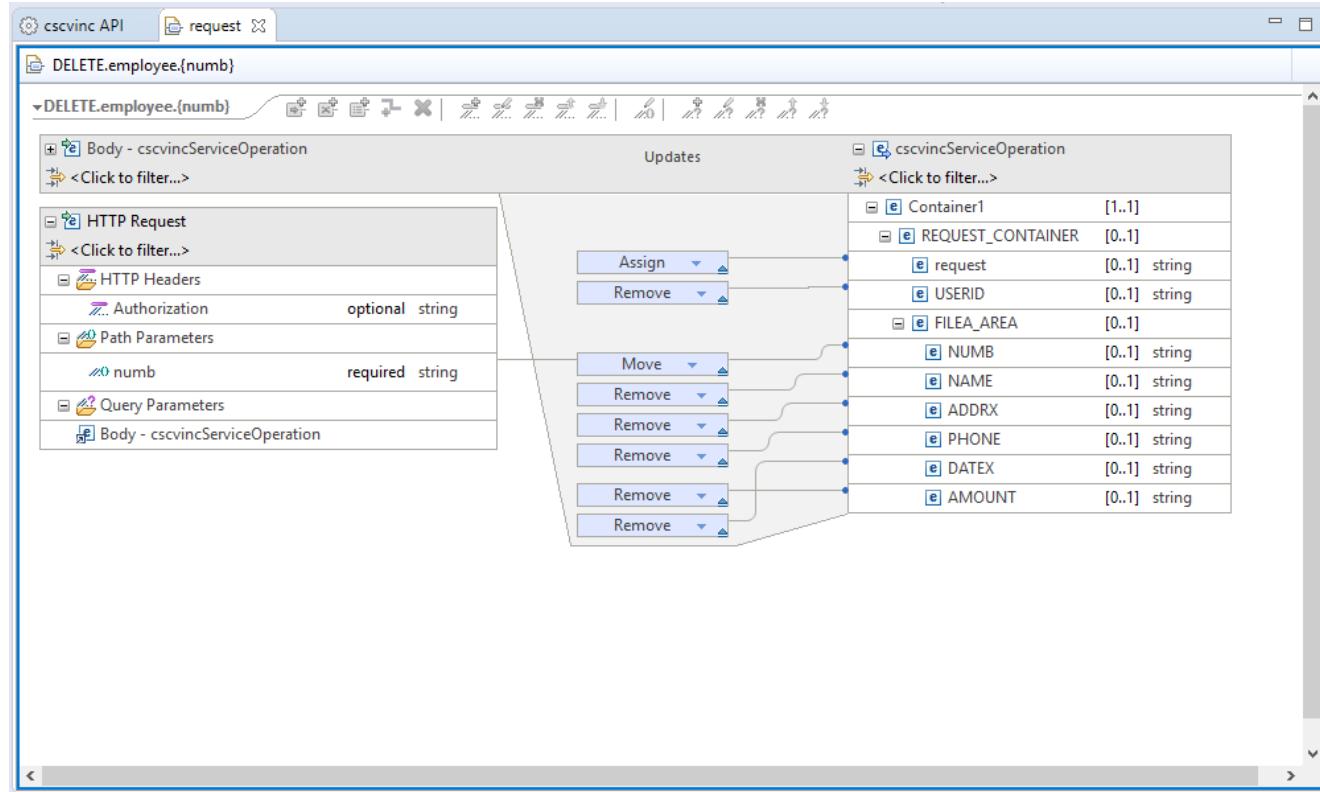
Assign static values to fields, useful for Op codes.

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



API toolkit – API Editor

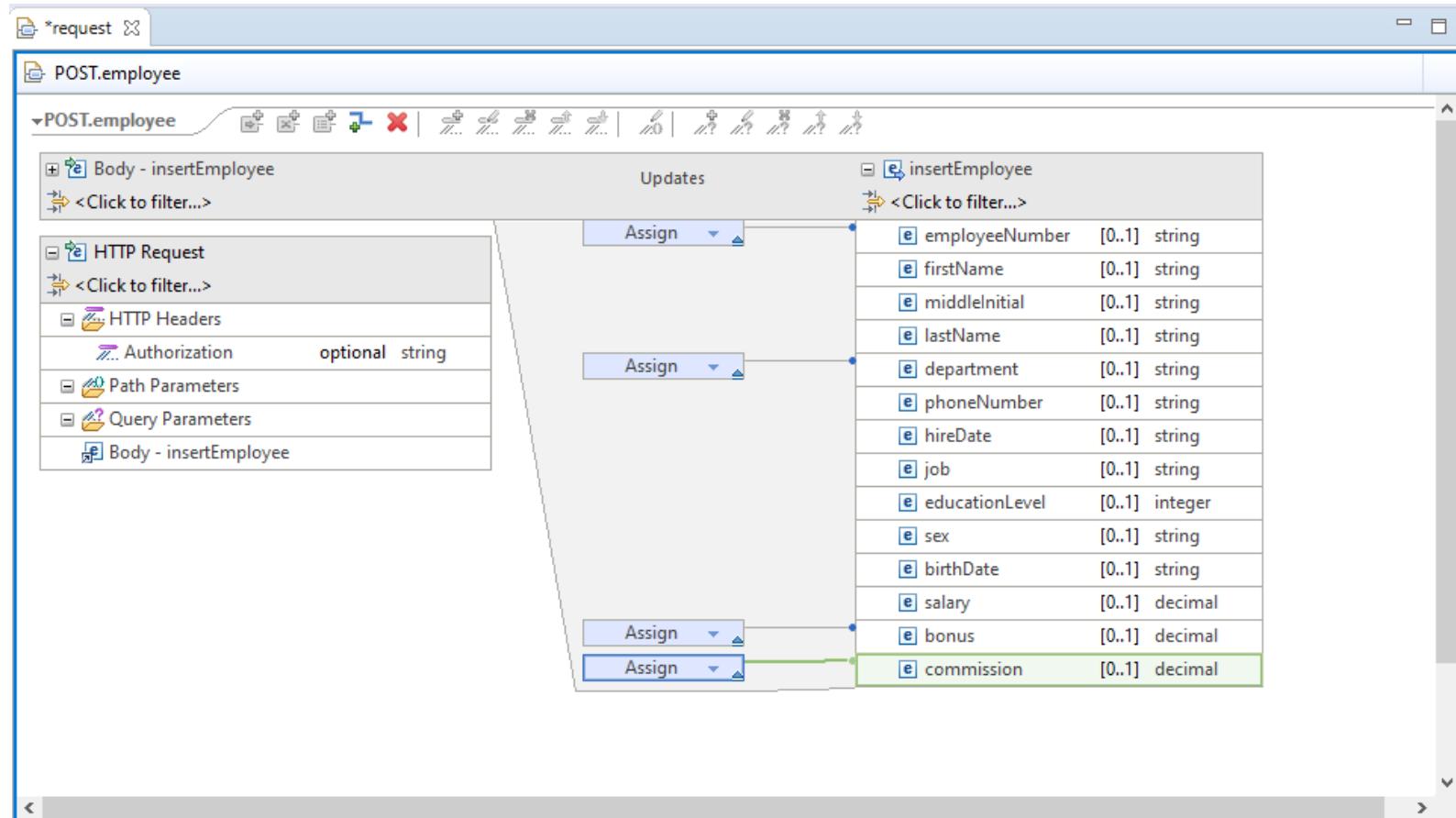
API mapping: Remove or assign values to the fields exposed by service developer





API toolkit – API Editor and Db2 REST service

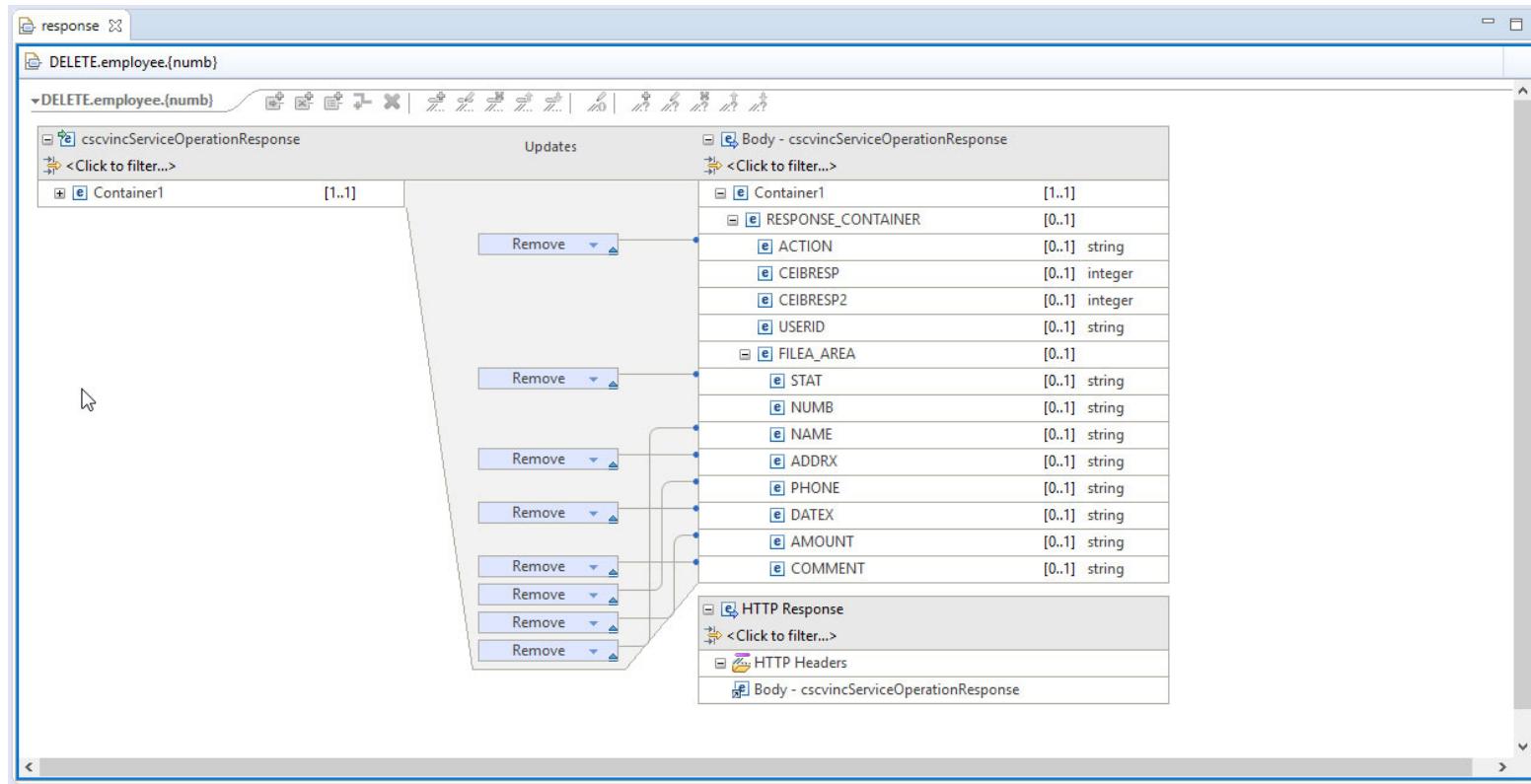
API mapping: Remove/Assign values columns exposed in Db2 REST service





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 a 'request' tab selected. The main window displays a 'POST.queue' request. On the left, the 'Body - MQPUTOperation' section is expanded, showing an 'HTTP Request' block with an 'HTTP Headers' sub-block. This sub-block contains two entries: 'Authorization' (optional string) and 'ibm-mq-md-correlID' (optional string). Both of these entries are circled in red. To the right, the 'MQPUTOperation' response structure is listed, containing fields: mqmessage, stat, numb, name, addrx, phone, datex, amount, and comment, each with a [1..1] multiplicity indicator.



API toolkit

API mapping: API definition with multiple response codes

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

In the center, a detailed view of a GET operation named "getCscvincSelectService" is shown. It includes fields for "Operation id:" and "Responses (2)".

A modal window titled "Edit Response 404" is open, showing the configuration for the 404 response code. The "Response code:" field is set to "404 - Not Found" and the "Description:" field is set to "Not Found". Below these fields, instructions say: "Define rules that indicate whether to use this response code and apply its response mapping, if defined." Two rules are defined:

- Rule 1: `se/Container1/RESPONSE_CONTAINER/CEIBRESP` = 13
- Rule 2: `/Container1/RESPONSE_CONTAINER/CEIBRESP2` = 80

At the bottom of the modal, there is a "Summary" section containing the text "Rule 1 AND Rule 2".

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

Separate mappings can be defined for each response code.

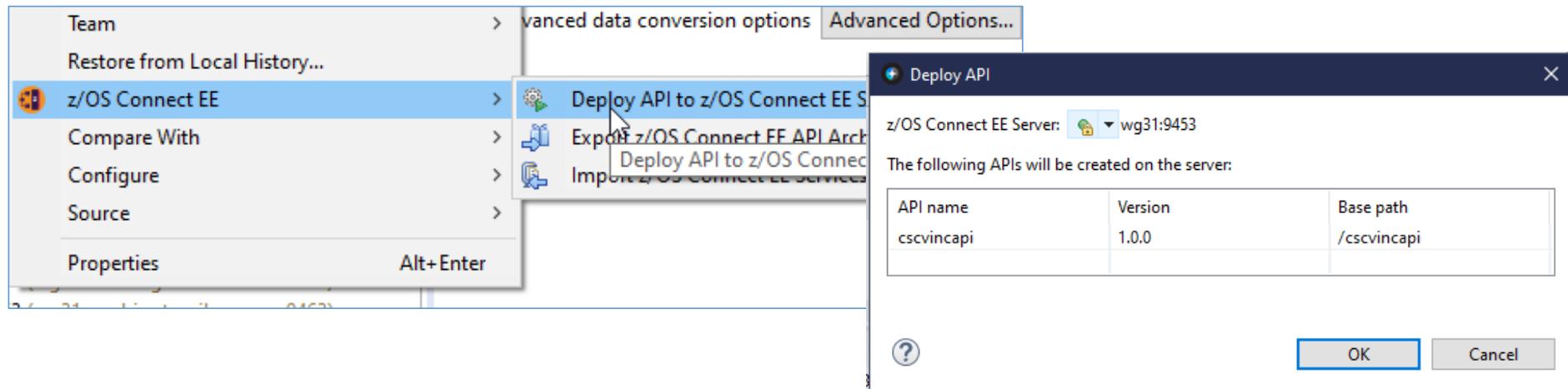
You can define rules based on fields in the service's return interface to tell z/OS Connect EE which response code to return



API toolkit – API Editor

Server connection and API deployment

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



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

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



API toolkit – API Editor

Testing with Swagger UI

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

The screenshot shows the z/OS Connect EE Servers interface with the API Editor open. On the left, the navigation pane shows 'wg31:9443 (wg31.washington.ibm.com:9443)' with 'APIs (1)' expanded, showing 'cscvinc (S)' with options: 'Open In Swagger UI', 'Open In API Explorer', 'Start API', 'Stop API', 'Remove API', and 'Show Properties View'. Below it, 'Services (4)' are listed. The main pane displays the Swagger UI for the 'cscvinc' service. It shows four API endpoints: POST /employee, DELETE /employee/{employee}, GET /employee/{employee}, and PUT /employee/{employee}. The GET endpoint is selected. The right pane shows the detailed API definition for the GET /employee/{employee} endpoint. It includes the Response Class (Status 200), which is OK, and the Model (Example Value) which is a JSON object:

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

The Response Content Type is set to application/json. Parameters for Authorization (header, string) and employee (path, string) are defined. The Response Messages section shows a 404 Not Found message with a Model Example Value.



/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)

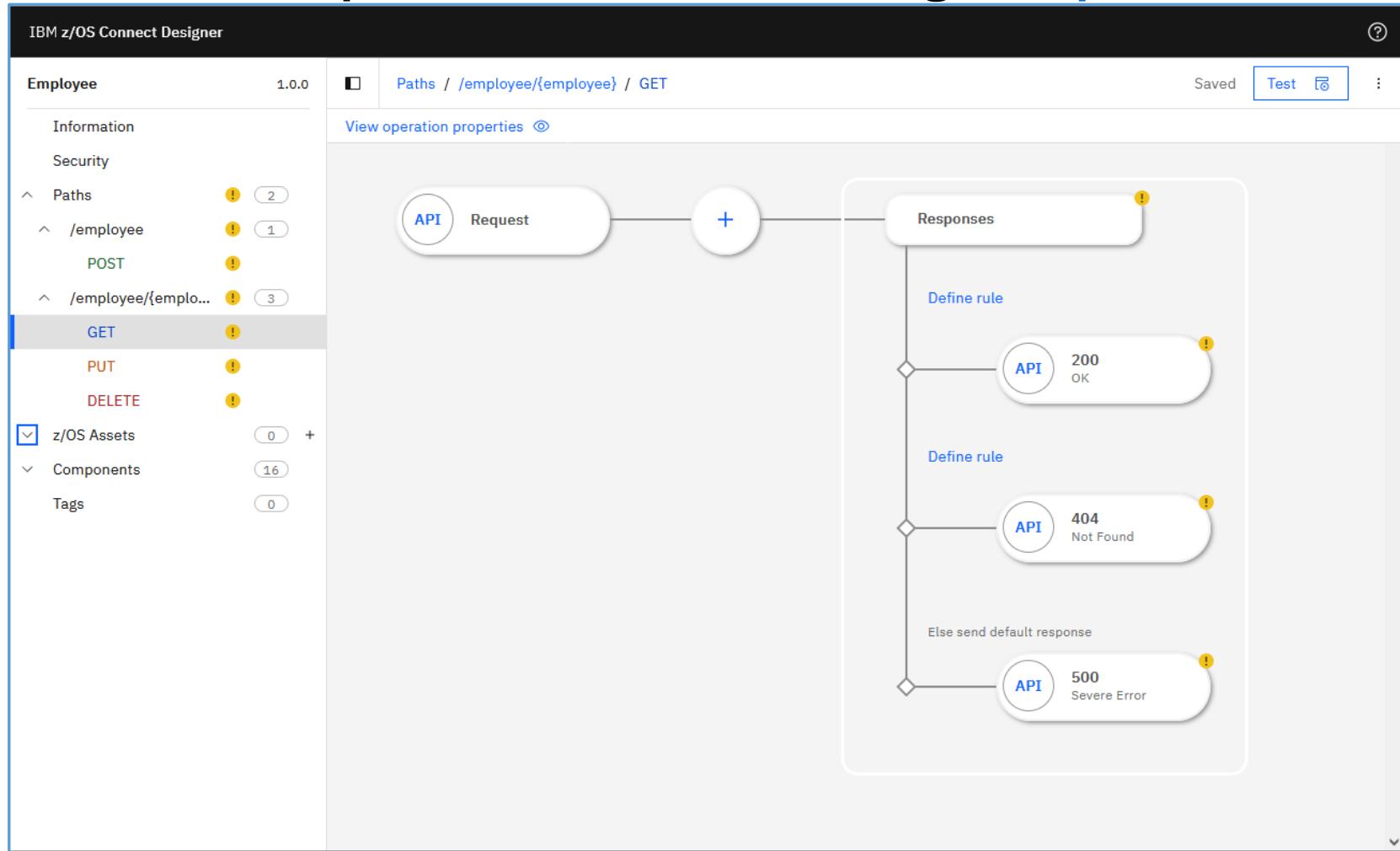


```
cscvinc.yaml - Notepad
File Edit Format View Help
openapi: 3.0.0
info:
  description: "CICS Filea Sample VSAM Application"
  version: 1.0.0
  title: cscvinc
x-ibm-zcon-roles-allowed:
- Manager
paths:
  /employee:
    post:
      tags:
        - cscvinc
      operationId: postCscvincInsertService
      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

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

Step 2 of 5
Add z/OS Asset

Select a z/OS Asset type
CICS channel program

CICS program name
CSCVINC

Program language
COBOL

CCSID
037

Select a CICS connection
cicsConn

Optional configuration
Transaction ID (optional)
Input Transaction ID

Transaction ID usage (optional)
Select usage

Previous Next



Accessing a z/OS CICS program (OpenAPI 3)

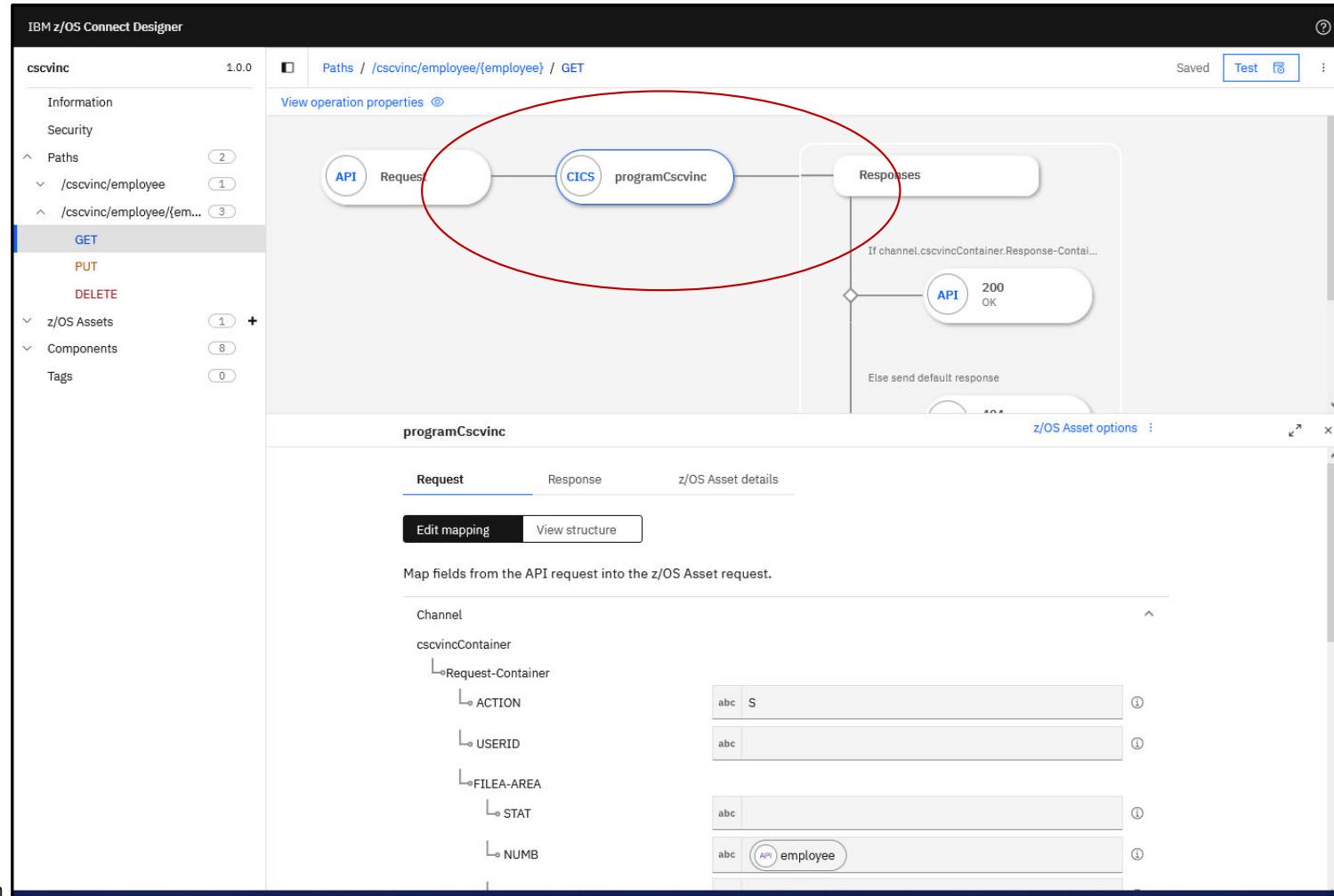
The screenshot shows the IBM z/OS Connect Designer interface. On the left, there is a navigation sidebar with the following structure:

- cscvinc** (selected) 1.0.0
- Information
- Security
- Paths 2
- z/OS Assets** 1 +
- programCscvinc** (selected)
- Components 8
- Tags 0

The main panel displays the configuration for the selected asset:

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

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





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)





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

IBM z/OS Connect Designer

EmployeesApi 1.1

Information

Security

Paths

/employees/{id}

GET

PUT

DELETE

/employees

GET

POST

z/OS Assets

addEmployee

deleteEmployee

getEmployee

getEmployees

selectByRole

updateEmployee

Components

Tags

Step 3 of 4

Add z/OS Asset

Select a Db2 connection

db2Conn

Import from Db2 service manager

Db2 native REST service collection ID

e.g. SYSIBMSERVICE

Db2 native REST service name

e.g. myService

Db2 native REST service version (optional)

e.g. V1

Import Db2 native REST service request schema

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

Specify a URL

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

Import file

Import Db2 native REST service response schema

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

Previous

Next



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

Add z/OS Asset / Import Db2 native REST service

Import Db2 native REST service

Select a Db2 connection

db2Conn

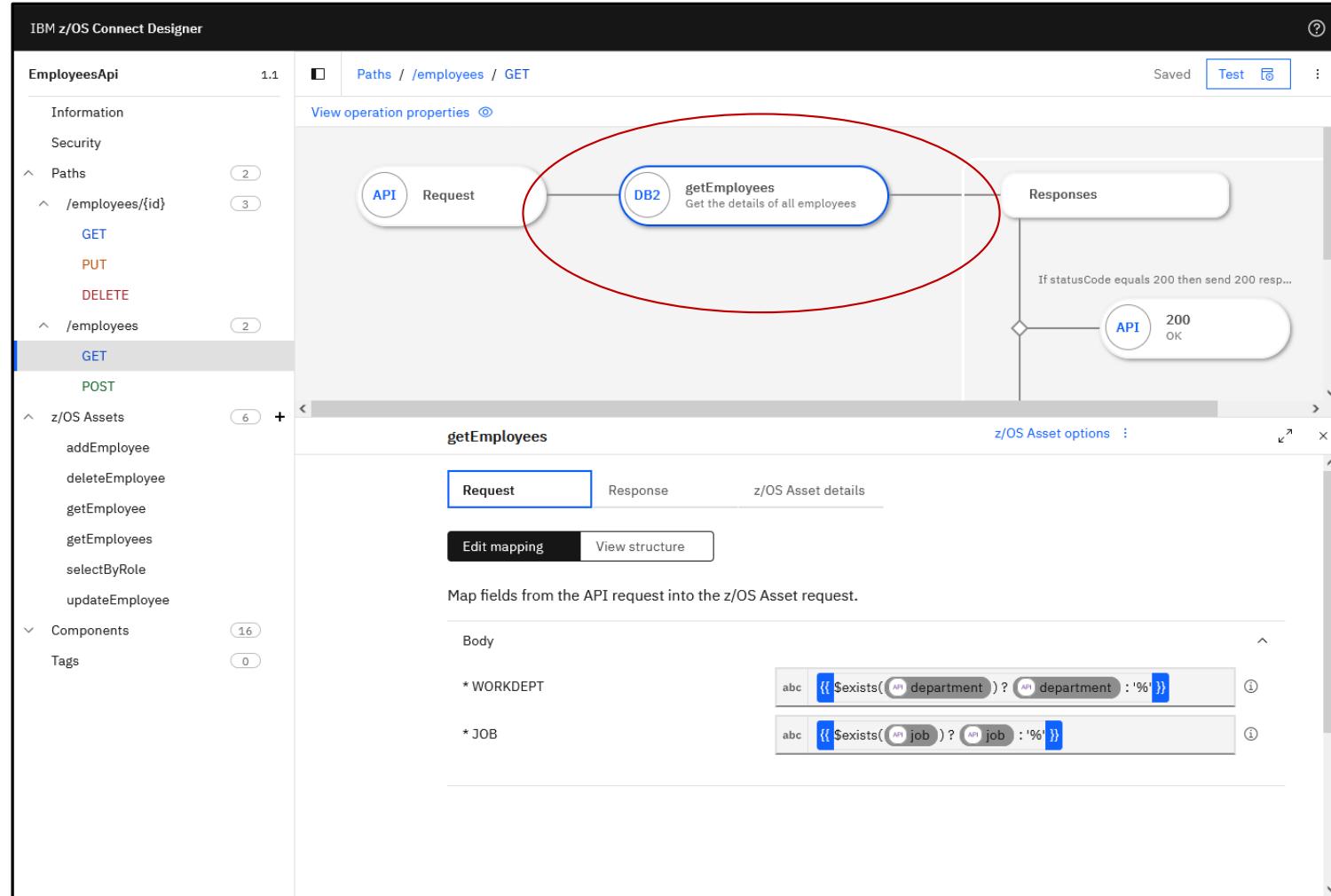
Service name	Version	Collection ID	Path	Description	Status
addEmployee	V1	SYSIBMSERVICE	/services/SYSIBMSERVI...	Add the details of an ind...	Available
deleteEmployee	V1	SYSIBMSERVICE	/services/SYSIBMSERVI...	Remove the details of a...	Available
getEmployee	V1	SYSIBMSERVICE	/services/SYSIBMSERVI...	Get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

Items per page: 10 ▾ 1–10 of 12 items

Previous Import

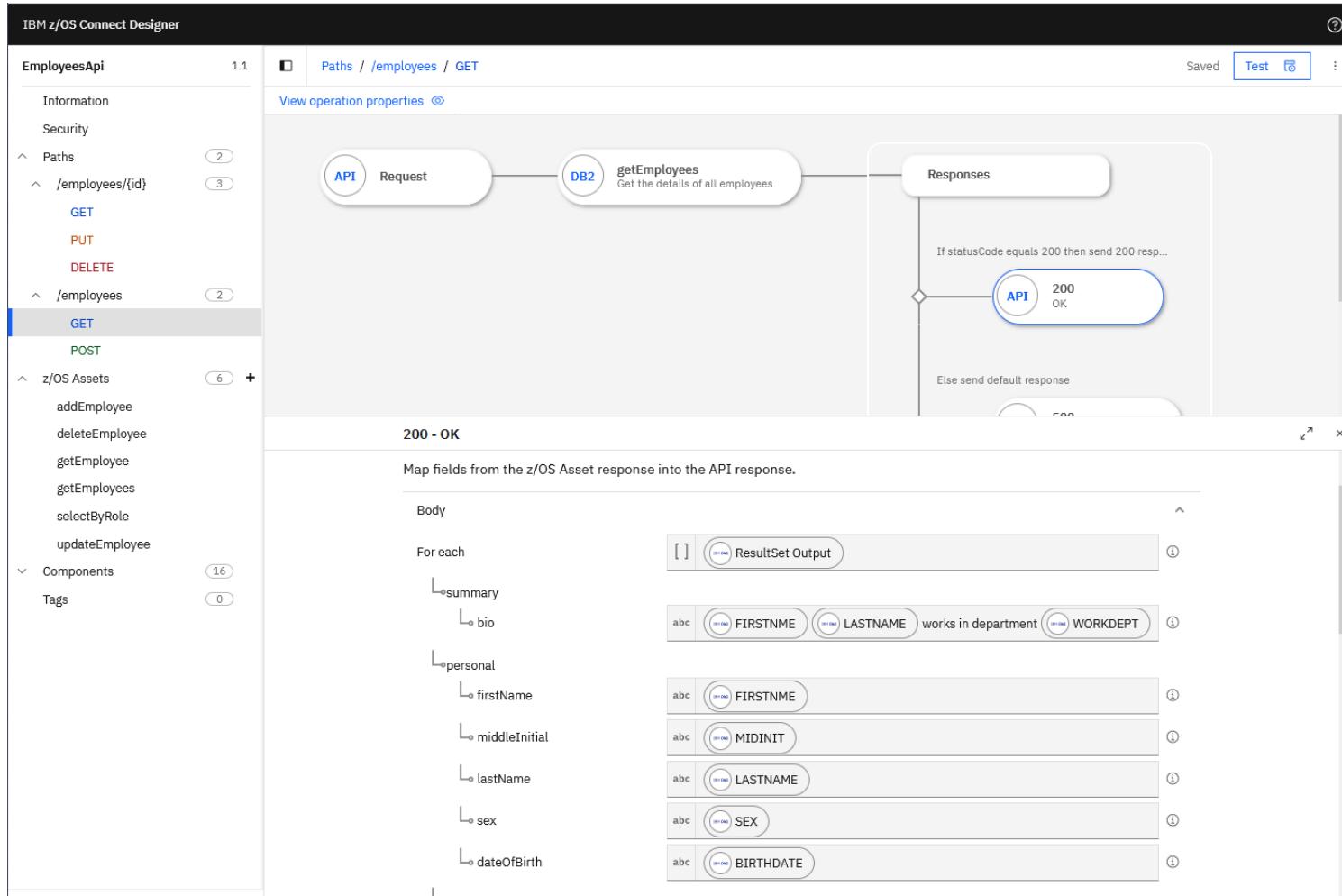
1 ▾ of 2 pages

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)

The screenshot shows the IBM z/OS Connect Designer interface for creating an API. On the left, the navigation pane lists the 'EmployeesApi' version 1.1, containing paths for '/employees/{id}' (with GET, PUT, DELETE methods) and '/employees' (with GET, POST methods). It also lists 'z/OS Assets' like 'displayEmployee', 'getEmployee', and 'getEmployees', and 'Components' (16 available). The main workspace displays the flow for the '/employees' GET operation. A 'Request' node connects to a 'DB2' node labeled 'getEmployees' with the description 'Get the details of all employees'. An arrow points from the DB2 node to a 'Responses' block. Inside the Responses block, a condition 'If statusCode equals 200 then send 200 resp...' leads to a '200 OK' response node. Below the flow, the '200 - OK' response body is defined as a 'ResultSet Output' containing fields: 'FIRSTNME', 'LASTNAME', 'WORKDEPT', 'FIRSTNME', and 'MIDINIT'. The 'Tags' section is currently empty.



z/OS Connect Designer for OpenAPI 3 (404)

IBM z/OS Connect Designer

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

Information Security Paths /employees/{id} 2 3

GET PUT DELETE

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

View operation properties

200 Updated

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

404 Not Found

Else send default response

500 Internal Server Error

404 - Not Found

Edit mapping View structure

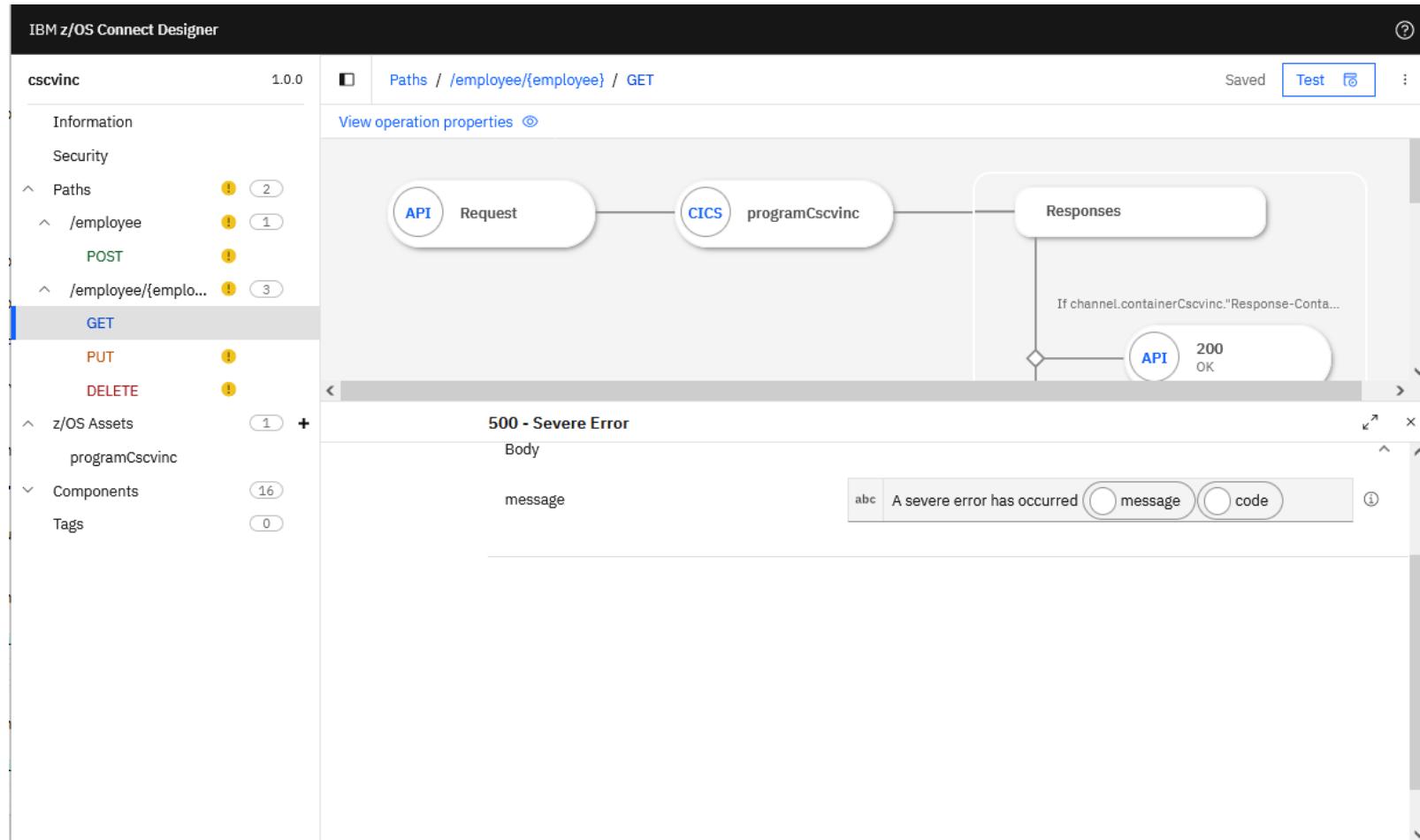
Map fields from the z/OS Asset response into the API response.

Body

message abc Employee not found ⓘ

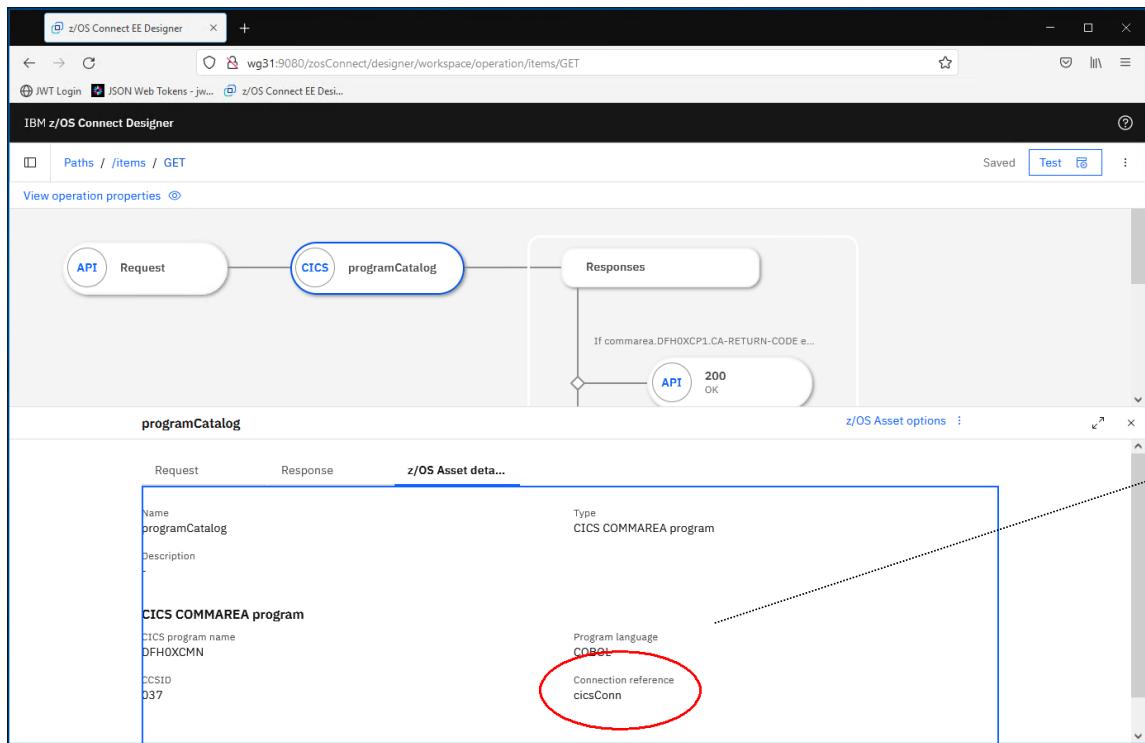


z/OS Connect Designer for OpenAPI 3 (500)





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



The screenshot shows the 'Server Config' interface with the 'cics.xml' file open. The 'Source' tab is selected, displaying the XML configuration code. A callout box points to the 'zosconnect_cicsIpicConnection' element, which is highlighted with a red oval.

```
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
```

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 IBM z/OS Connect Designer interface. At the top, there's a navigation bar with tabs like 'Paths / /items / GET'. Below it is a main canvas with a flowchart. A 'Request' node leads to a 'DB2' node labeled 'getEmployee-V1'. This leads to a 'Responses' node containing a condition: 'If commarea.DFH0XCP1.CA-RETURN-CODE e...'. From this condition, an 'API' node leads to a '200 OK' response. The 'getEmployee-V1' node has a detailed description: 'Get the details of all employees'. It also lists 'Db2 native REST service' details: 'Db2 native REST service name: getEmployee', 'Collection ID: SYSIBMSERVICE', and 'Db2 native REST service description: Get the details of all employees'. A red circle highlights the 'Connection reference: db2Conn' under the 'Db2 native REST service' section.

The screenshot shows the 'Server Config' interface with the 'db2.xml' configuration file open. The 'Source' tab is selected, displaying XML code for a Db2 connection. A red arrow points from the 'Connection reference: db2Conn' in the Designer interface to the 'zosconnect_db2Connection' element in the XML code. A callout box on the right says: 'Define connections to Db2 using variables defined in bootstrap.properties file'.

```

<?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>

```

```

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" 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 EJBRULE BBGZDFLT.CatalogManager.Manager
REDFINE EJBRULE 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*.



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. Below the navigation is a toolbar with icons for cloud, collection, settings, and notifications, along with an Upgrade button. The main workspace shows a list of collections and a selected item: "https://wg31.washington.ibm.com:9453/cscvinc/employee/948478". A detailed request card is displayed, showing a GET method, URL "https://mpz3.washington.ibm.com:9443/cscvinc/employee/111111", and various tabs like Params, Authorization, Headers, Body, Pre-request Script, Tests, and Settings. The Params tab is active, showing a table for Query Params. The Body tab shows the raw JSON response. The status bar at the bottom indicates a 200 OK status, 205 ms time, and 899 B size, with a Save Response button.

mitchj@us.ibm.com

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



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

IBM

all Filter

Liberty REST APIs

Discover REST APIs available within Liberty

cscvinc

POST /cscvinc/employee

DELETE /cscvinc/employee/{employee}

GET /cscvinc/employee/{employee}

PUT /cscvinc/employee/{employee}

db2employee

filemgr

imsPhoneBook

jwltlpvDemoApi

miniloancics

mqapi

phonebook

Show/Hide | List Operations | Expand

File Edit View History Bookmarks Tools Help REST API Documentation + https://mpz3.washington.ibm.com:9443/api/explorer/#/cscvinc/employee/111111

Curl Try it out! Hide Response curl -X GET --header 'Accept: application/json' --header 'Authorization: Basic RnJlZDpmcmVk' 'https://mpz3.washington.ibm.com:9443/cscvinc/employee/111111'

Request URL https://mpz3.washington.ibm.com:9443/cscvinc/employee/111111

Response Body

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

Response Code 200

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



API Testing with cURL

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

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

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

MVS JCL Invoking curl using Rocket Software's tooling



```
//*****  
/* SET SYMBOLS  
//*****  
//EXPORT EXPORT SYMLIST=(*)  
// SET CURL= '/usr/1pp/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=sto+  
pped > 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
```



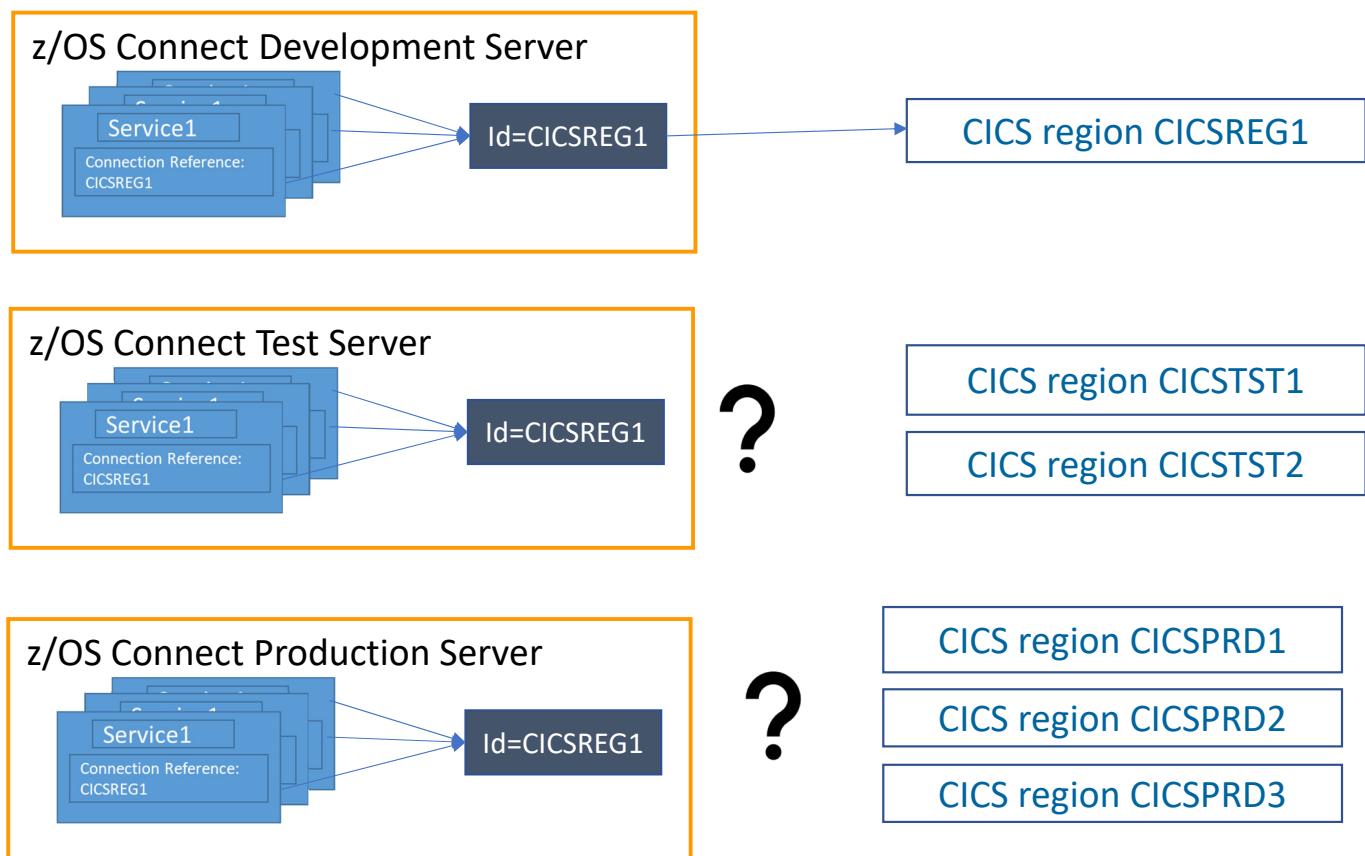
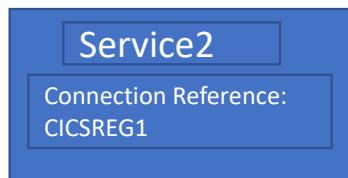
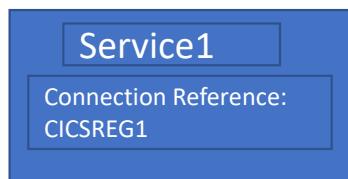
Connection References

Carefully consider the names used for connections



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

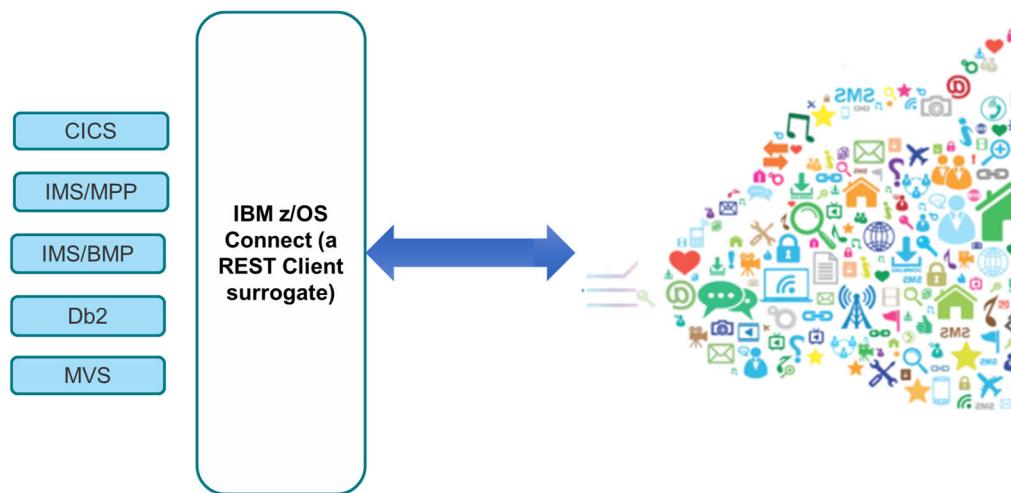
Don't couple service and API requester connection names to specific systems or endpoints





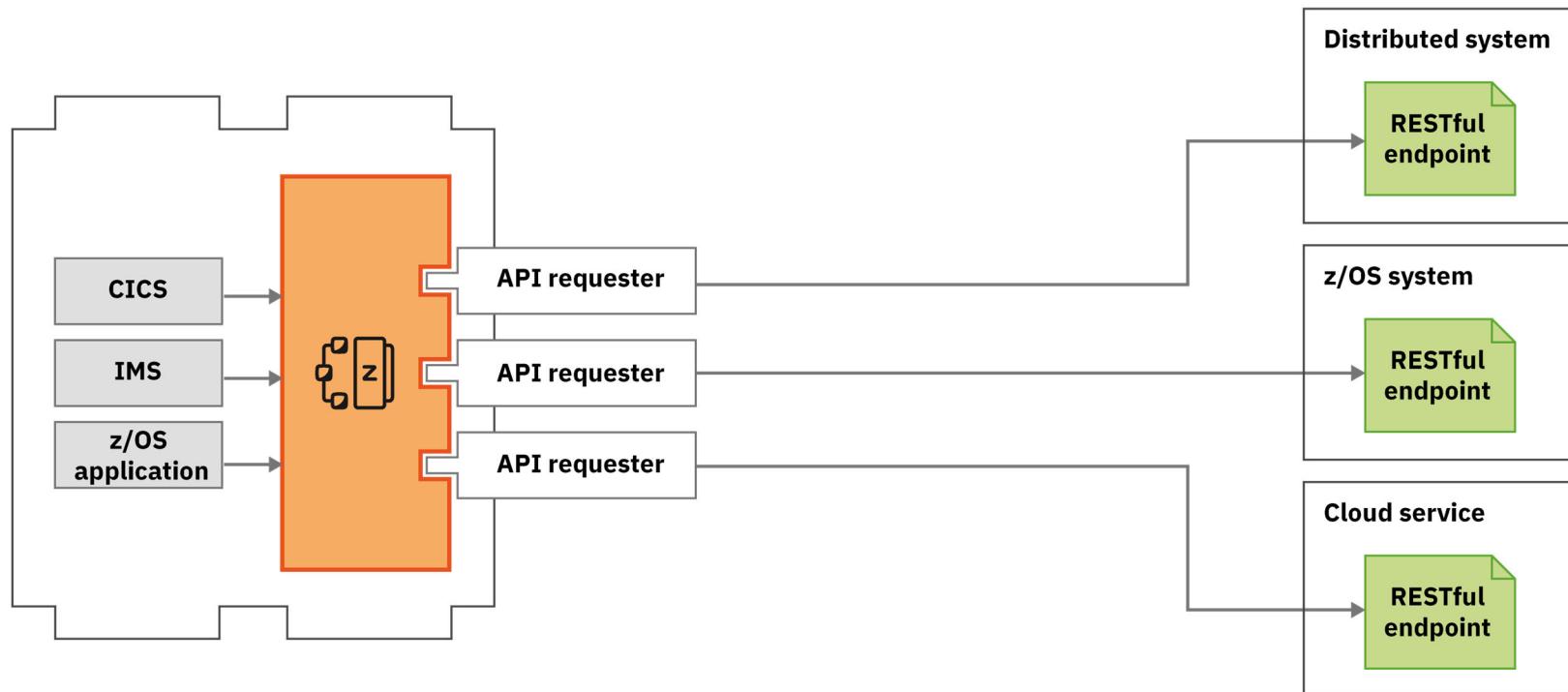
/api_toolkit/apiRequesters

Quick and easy **API mapping**.





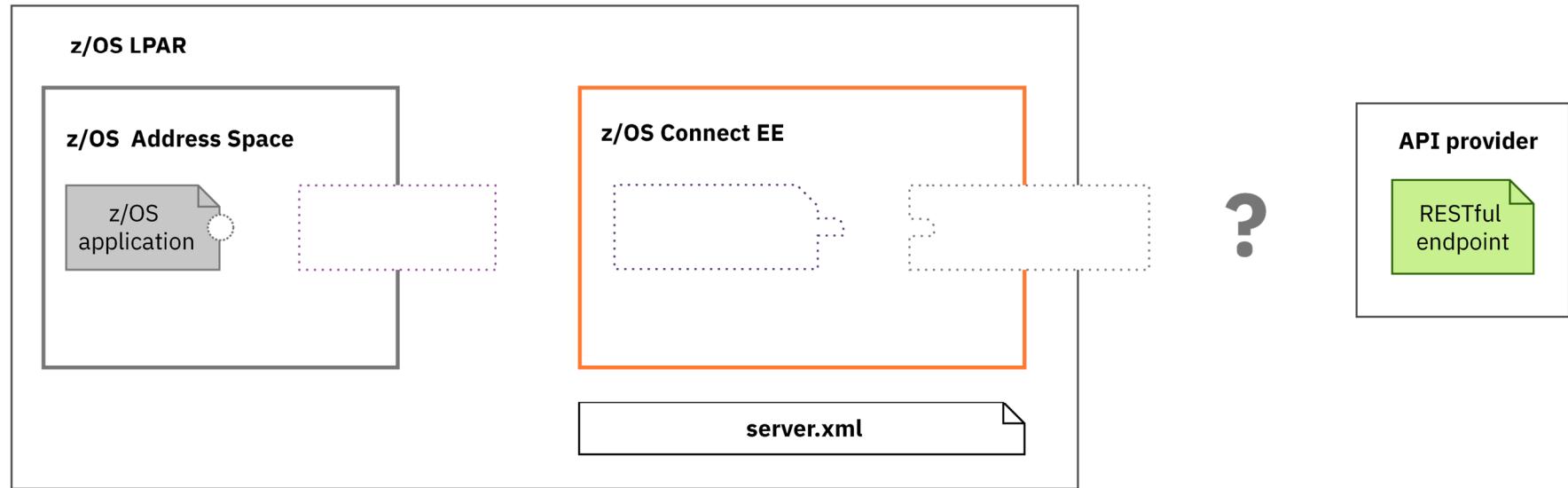
Use API requester to call external APIs from z/OS assets





Steps to calling an external API

Starting point



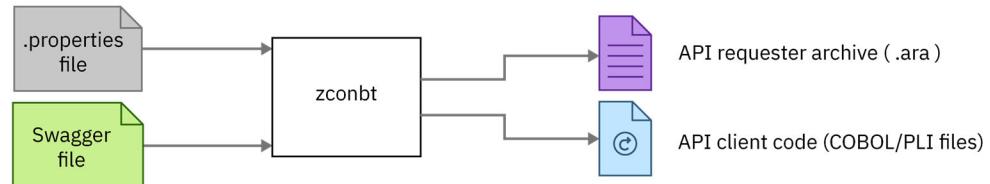


Steps to calling an external API

Generate API requester archive and API client code from Swagger

The screenshot shows a JSON editor window displaying a Swagger JSON file. The file defines an API endpoint for retrieving employee information. Key fields include:

- swagger: "2.0"
- info: { description: "", version: "1.0.0", title: "cscvincapi", basePath: "/cscvincapi" }
- schemes: []
- consumes: ["application/json"]
- produces: ["application/json"]
- paths:
 - /employee/{employee}:
 - get:
 - tags: []
 - operationId: "getCscvincSelectService"
 - parameters:
 - 0:
 - name: "employee"
 - in: "path"
 - required: true
 - type: "string"
 - maxLength: 6
 - responses:
 - 200:
 - description: "OK"
 - schema: {}
 - 404:
 - schema: {}



```
.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>



Stop and consider COBOL Storage

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  
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"
```

mitchj@us.ibm.com

```
ATS01P01 - Notepad  
File Edit Format View Help  
* ++++++  
06 RespBody.  
09 memberContactsResponse-num PIC S9(9) COMP-5 SYNC.  
09 memberContactResponse OCCURS 255.  
12 umi-num PIC S9(9) COMP-5 SYNC.  
12 umi.  
 15 umi2-length  
 15 umi2 PIC X(255). SYNC.  
12 pin-num PIC S9(9) COMP-5 SYNC.  
12 pin.  
 15 pin2-length  
 15 pin2 PIC X(255). SYNC.  
12 firstName-num PIC S9(9) COMP-5 SYNC.  
12 firstName.  
 15 firstName2-length  
 15 firstName2 PIC X(255). SYNC.  
12 middleName-num PIC S9(9) COMP-5 SYNC.  
12 middleName.  
 15 middleName2-length  
 15 middleName2 PIC X(255). SYNC.  
12 lastName-num PIC S9(9) COMP-5 SYNC.  
12 lastName.  
 15 lastName2-length  
 15 lastName2 PIC X(255). SYNC.
```



Add constraints to the properties in the specification file

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

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

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

Three specific attributes are highlighted with red circles: `maxItems: 10`, `maxLength: 12`, and `maxLength: 30`.

The screenshot shows a Notepad window with the following content:

```
* 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 S9999 COMP-5
15 umi2                      PIC X(12).           SYNC.

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

12 firstName-num              PIC S9(9) COMP-5      SYNC.
12 firstName,
15 firstName2-length          PIC S9999 COMP-5
15 firstName2                 PIC X(30).           SYNC.

12 middleName-num             PIC S9(9) COMP-5      SYNC.
12 middleName,
15 middleName2-length         PIC S9999 COMP-5
15 middleName2                PIC X(30).           SYNC.
```

Several fields are circled in red: `memberContactsResponse` (OCCURS 10), `umi`, `umi2`, `pin`, `pin2`, `firstName`, `firstName2`, and `middleName`.



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 (firstName2)
      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 lastName delimited by size
      X'00' delimited by size into _lastName.
```



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.

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

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



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 'request.cpy' window displays a COBOL copybook with the following fields:

```
*      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-HZN-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 PIC S9999 COMP-5
SYNC.
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     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            PIC S9999 COMP-5
SYNC.
12 X-HZN-UserRole2-length   PIC S9999 COMP-5
SYNC.
12 X-HZN-UserRole2          PIC X(255).
09 X-HZN-UserAssociationI-num PIC S9(9) COMP-5 SYNC.
09 X-HZN-UserAssociationI    PIC S9999 COMP-5
SYNC.
```

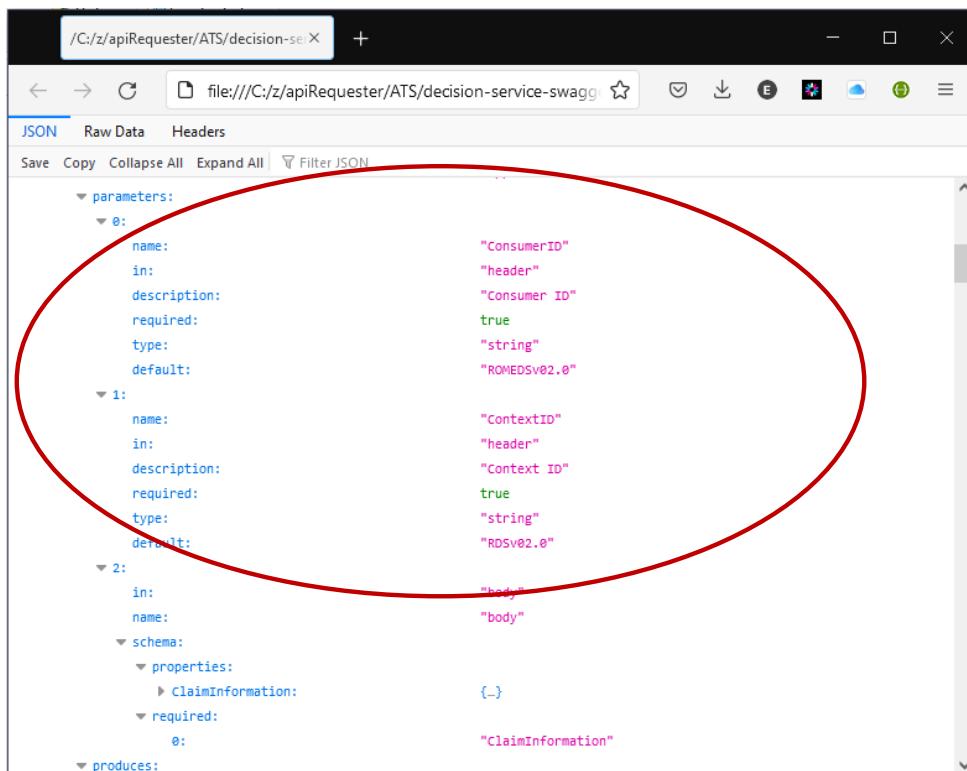
The 'mpz3' window displays an AS/400 edit session of a source program (USER1.ZCEE.SOURCE(GETAPIEN)) with the following code:

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

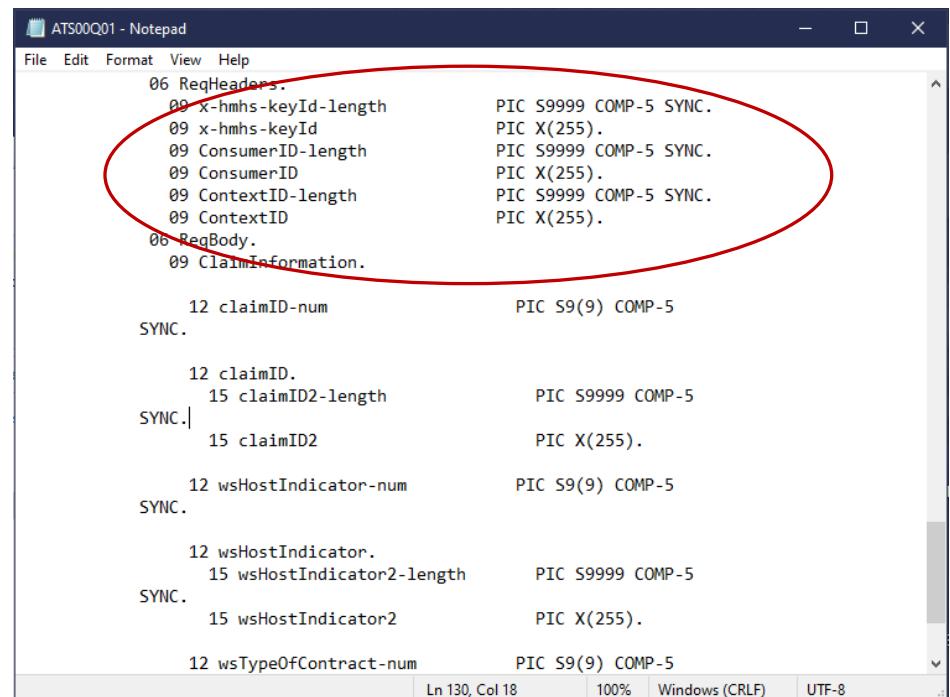


Additional Swagger header properties

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



```
/C:/apiRequester/ATS/decision-se X +  
file:///C:/apiRequester/ATS/decision-service-swagg  
JSON Raw Data Headers  
Save Copy Collapse All Expand All Filter JSON  
parameters:  
  0:  
    name: "ConsumerID"  
    in: "header"  
    description: "Consumer ID"  
    required: true  
    type: "string"  
    default: "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:
```



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



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

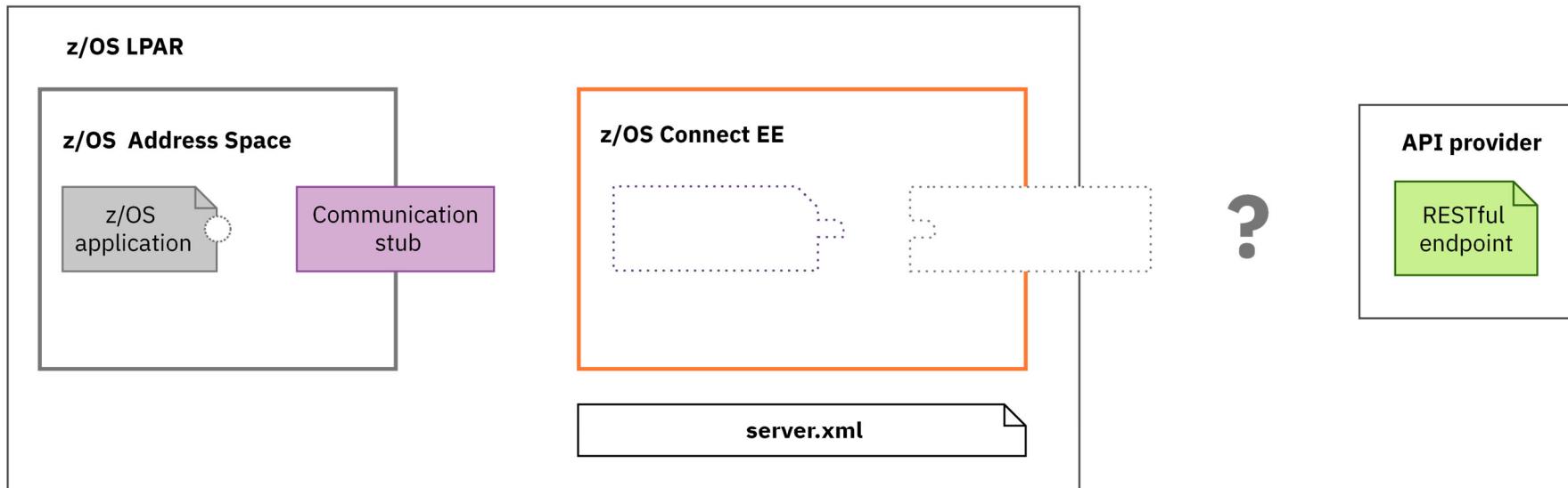
operationId: deleteCscvincDeleteService, basePath: /cscvincapi, relativePath: /employee/{employee}, method: DELETE
- request data structure : CSC03Q01
- response data structure : CSC03P01
- api info file : CSC03I01

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



Steps to calling an external API

Update the application by adding the copy books 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

i ibm.biz/zosconnect-configure-comms-stub



Steps to calling an external API

Include the generated copy books in a COBOL program

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

  * Copy API Requester required copybook
  COPY BAQRINFO.

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

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

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

API-REQUEST

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

API-RESPONSE

```
CSC00I01  CSC00Q01  CSC00P01 X
  * ++++++
  06 RespBody.
    09 cscvincap0-num          PIC S9(9) COMP-5 SYNC.
    09 cscvincap0-operatio.
      12 Container1.
    15 RESPONSE-CONTAINER2-num PIC S9(9) COMP-5
      SYNC.
```

API-INFO-OPER1

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



Steps to calling an external API

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

The diagram illustrates the steps to calling an external API. It shows the GETAPI program (left) interacting with the CSC00101 and CSC00P01 copy books (right).

GETAPI Program:

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

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

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

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

CSC00101 Copy Book:

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 'S2fcsvincapi%2Femployee%7D'.	
03 BAQ-APIPATH-LEN	PIC S9(9) COMP-5 SYNC
VALUE 41.	
03 BAQ-APIMETHOD	PIC X(255)
VALUE 'GET'.	
03 BAQ-APIMETHOD-LEN	PIC S9(9) COMP-5 SYNC
VALUE 3.	

CSC00P01 Copy Book:

06 ReqPathParameters.	
09 employee-length	PIC S9999 COMP-5 SYNC.
09 employee	PIC X(6).
*	
*	
*	
06 RespBody.	
09 cscvincSelectServiceOp-num	PIC S9(9) COMP-5 SYNC.
09 cscvincSelectServiceOperatio-	
12 Container1.	
15 RESPONSE-CONTAINER2-num	PIC S9(9) COMP-5
SYNC.	



Steps to calling an external API

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

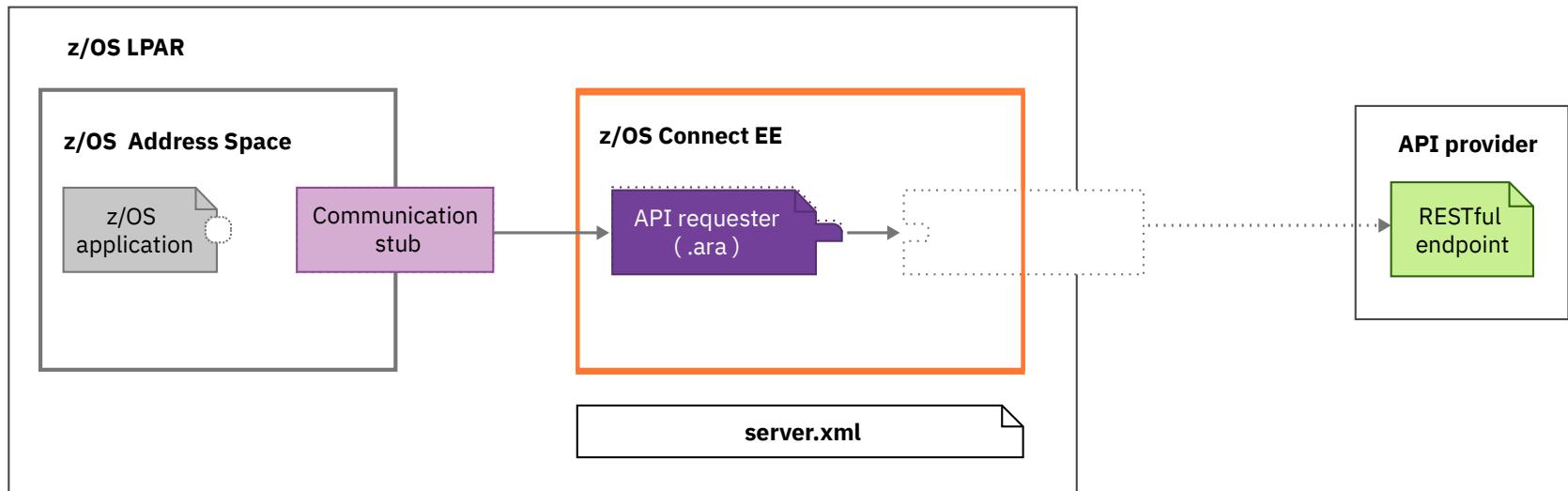
* Otherwise, some error happened in API, z/OS Connect EE server
* or communication stub. 'BAQ-STATUS-CODE' and
* 'BAQ-STATUS-MESSAGE' contain the detailed information
* of this error.
ELSE
    DISPLAY "Error code: " BAQ-STATUS-CODE
    DISPLAY "Error msg: " BAQ-STATUS-MESSAGE
    MOVE BAQ-STATUS-CODE 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.
    LINES BAQ-ERROR-IN-API
```

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



Steps to calling an external API

Deploy API requester (.ara) archive



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



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 made to the URL <https://wg31.washington.ibm.com:9483/zosConnect/apiRequesters>. The 'Body' tab is selected, and the file 'filea.ara' is chosen as the data type. The response status is 201 Created, and the JSON response body is displayed:

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

Command:

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

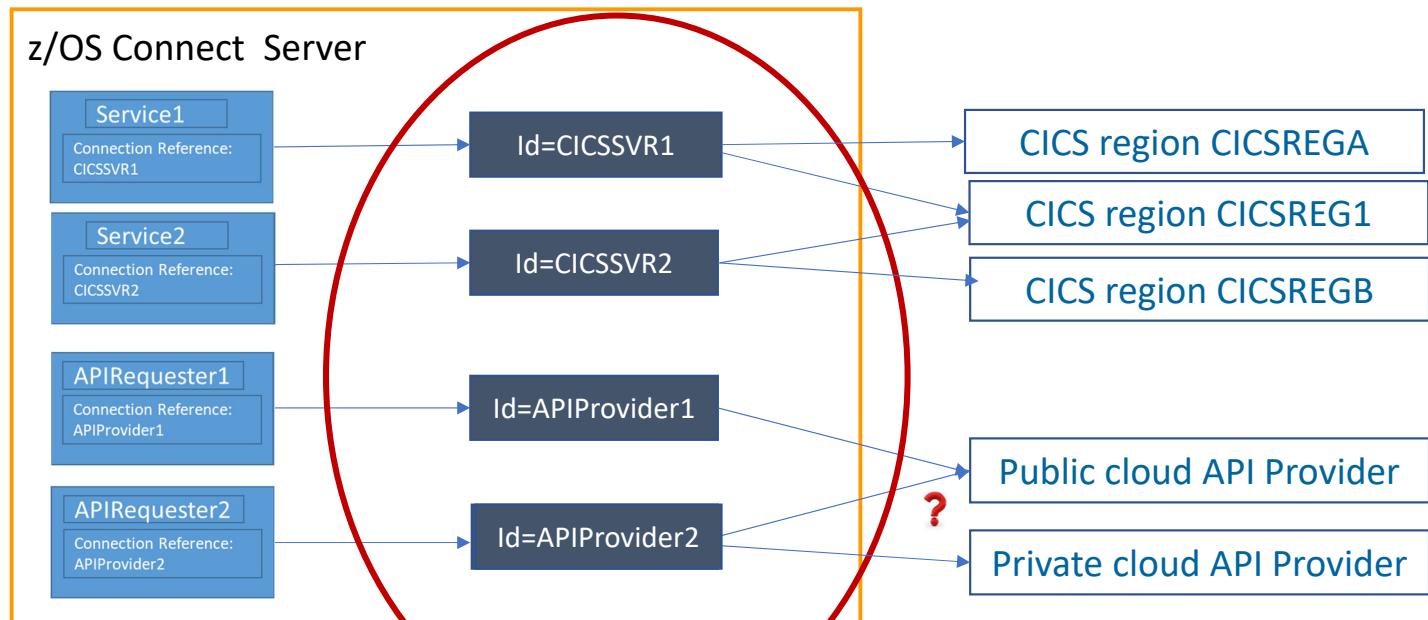
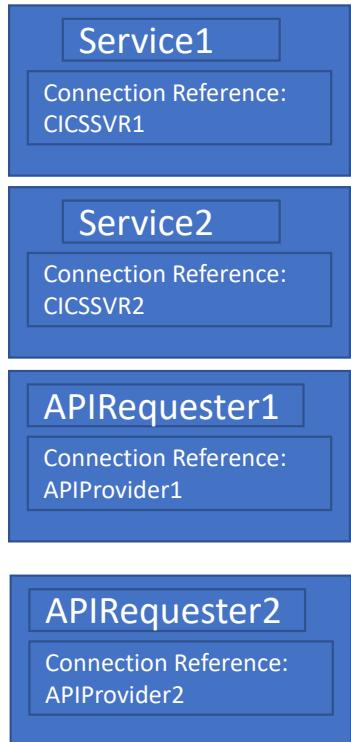
Results:

```
{"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"}
```



Use naming conventions for connection references

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

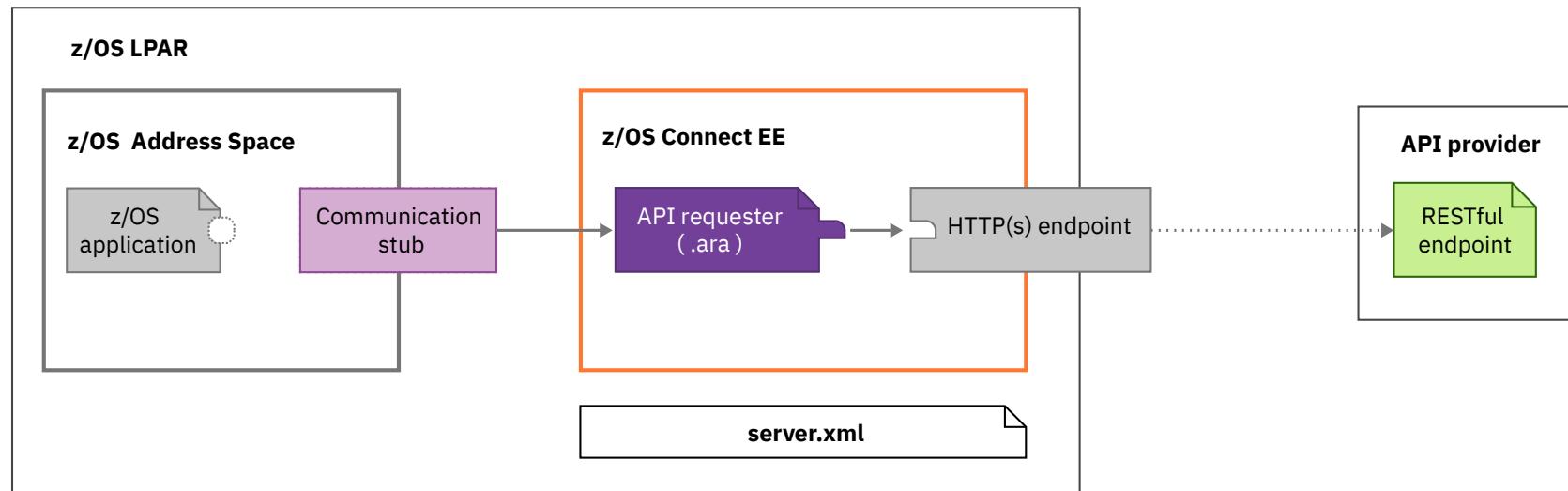


```
<zosconnect_apiRequesters>
  requireAuth="true|false"
  <apiRequester name="cscvincapi_1.0.0"
    connectionRef="APIProvider2"
  </zosconnect_apiRequesters>
```



Steps to calling an external API

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 window. The URL is /C:/apiRequester/cscvinc/swagger.json. The JSON content includes:

```
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:
```

A red arrow points from the highlighted 'basePath' field in the JSON to the 'BAQ-APIPATH' field in the IBM i editor below.

The IBM i editor window titled CSC02I01 contains the following code:

```
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 for 'apiRequesterHTTPS.xml'. 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" />
```

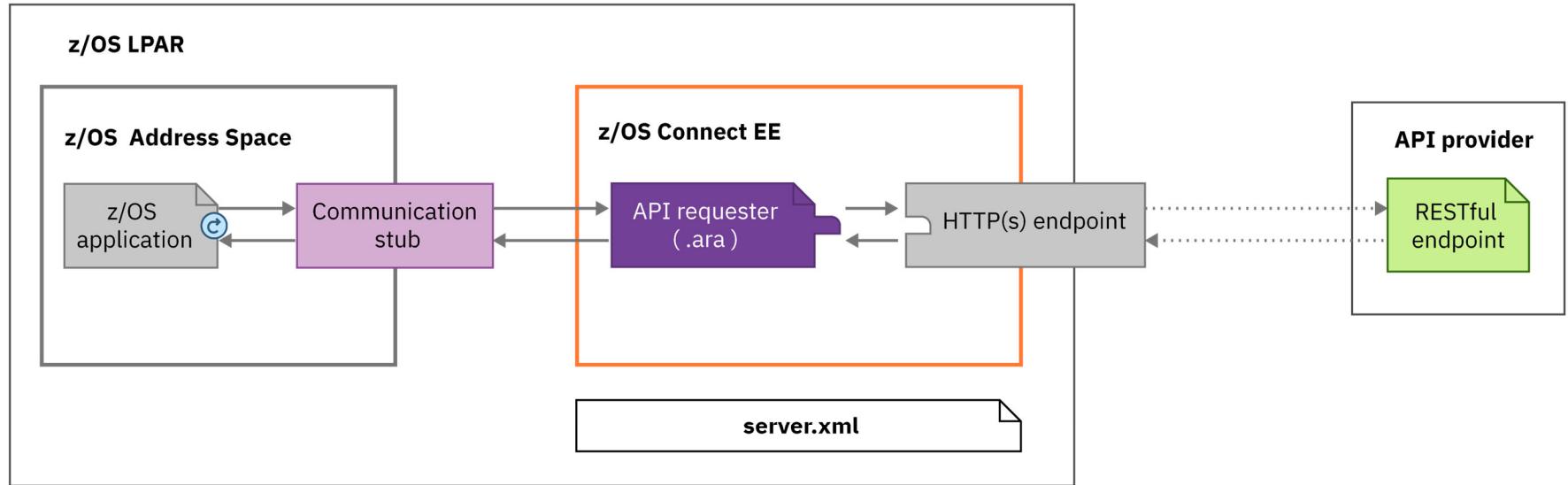
A red arrow points from the 'BAQ-APIPATH' field in the IBM i editor above to the 'host' attribute in the XML code.

<http://dvipa.washington.ibm.com:9443/cscvincapi/employee/{numb}>



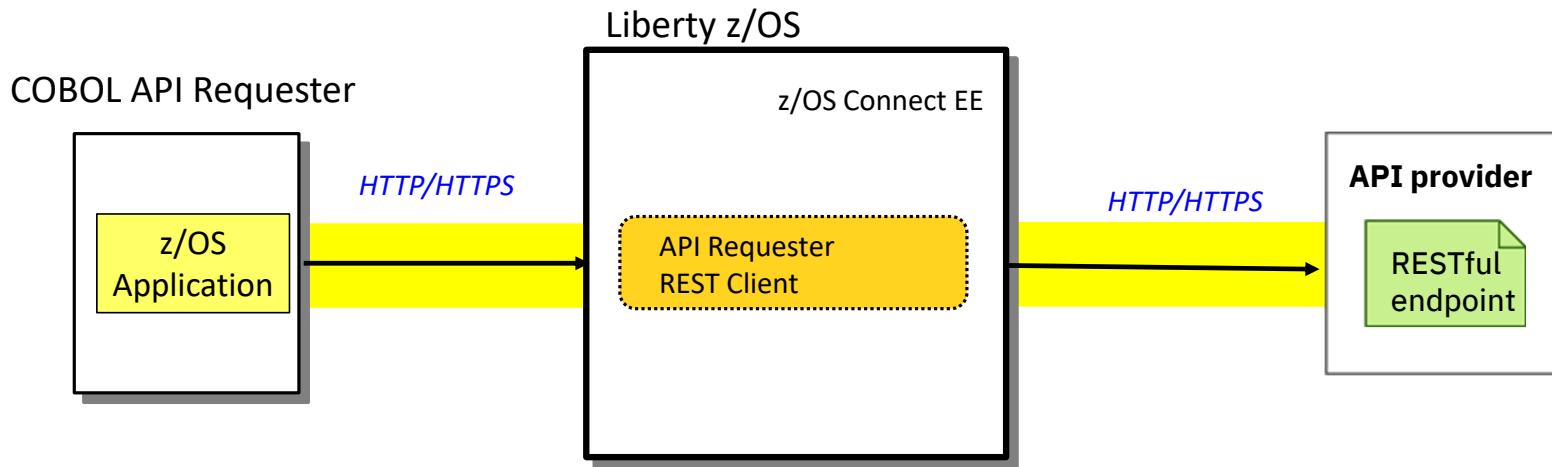
Steps to calling an external API

Done





API requester to API Provider connection overview



MVS Batch and IMS HTTP connection details provided by:

- Environment Variables (BAQURI, BAQPORT)
 - 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 )
Tcpinservice()
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
M A 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

mitchj@us.ibm.com

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")

```

```

mp3
File Edit Settings View Communication Actions Window Help
Menu Utilities Compilers Help
BROWSE ZCEE30.SBAQC0B(BAQRINFO) Line 000000010 Col 001 080
Command ==> Scroll ==> PAGE
* (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
07 BAQ-OAUTH-PASSWORD VALUE 0.
07 BAQ-OAUTH-PASSWORD-LEN PIC X(256),
PIC S9(9) COMP-5 SYNC
04/015
Connected to remote server/host mp3 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="BBGZDFLT" />
```



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 'ATSPKTTC'.
*-----*
***** 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 (ATSPKTTC)

```
*-----*
* 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
```



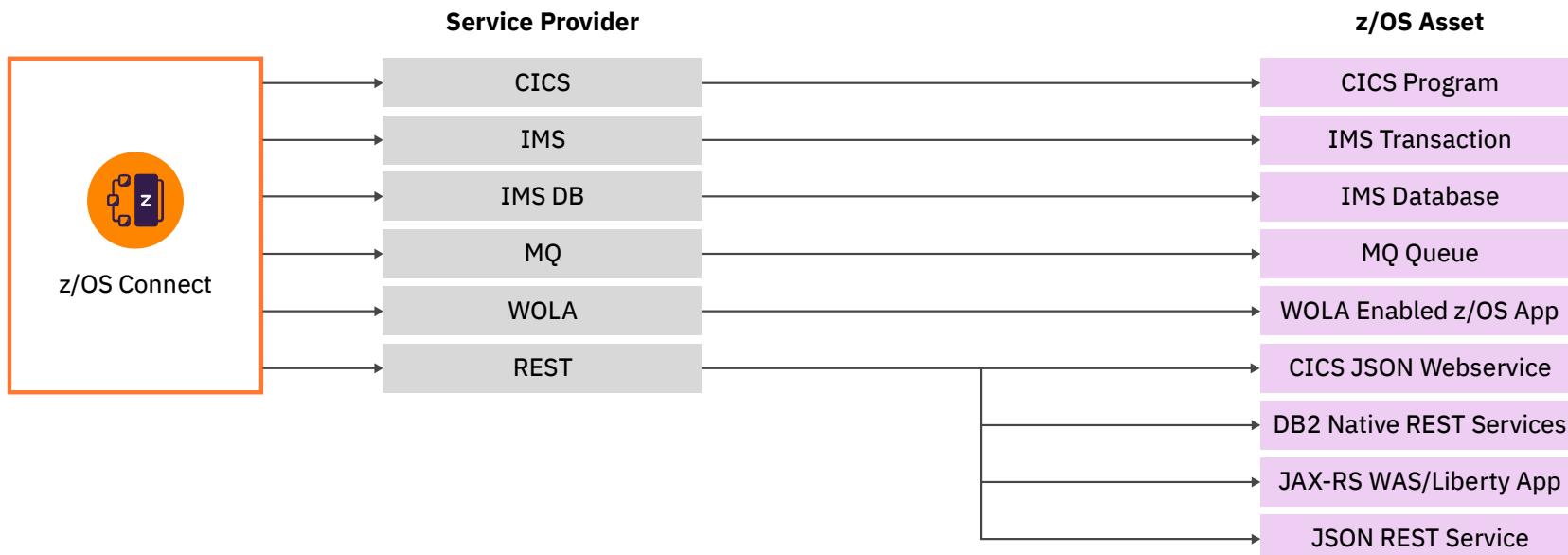
/miscellaneousTopics

performance, high availability, Liberty



What assets can z/OS Connect EE map to?

And which service provider could I use?

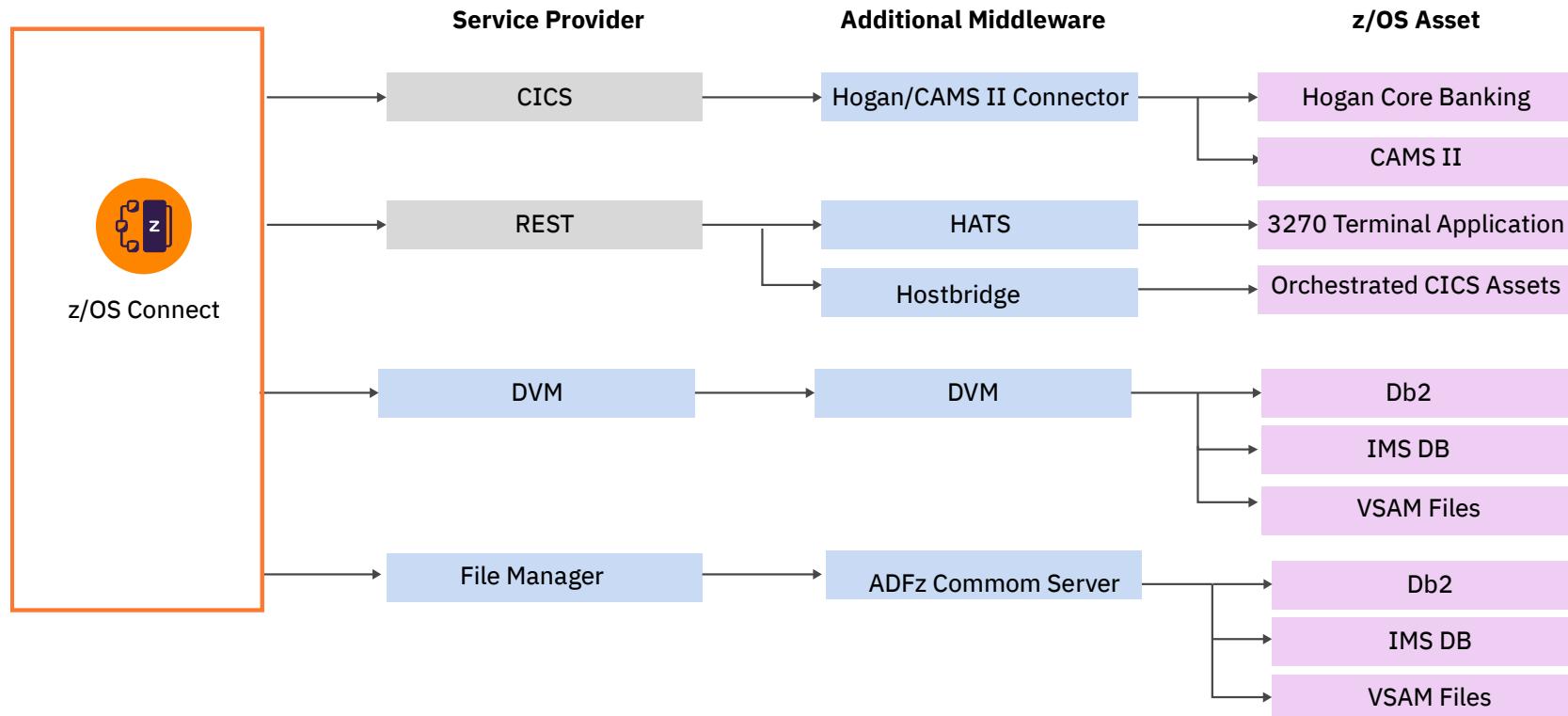


The core **service providers** included with z/OS Connect EE provide API access to a wide range of z/OS assets.



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

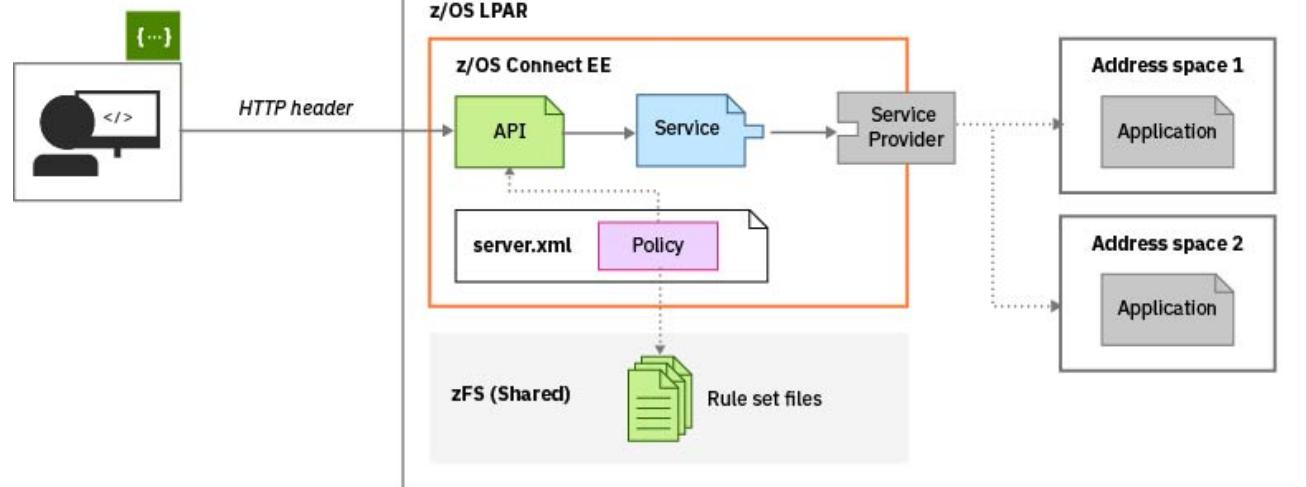
- imsConnectionRef
- imsInteractionRef
- imsInteractionTimeout
- imsLtermOverrideName
- imsTranCode
- imsTranExpiration

Db2 attributes

- db2ConnectionRef
- db2CollectionID

MQ attributes

- mqConnectionFactory
- mqDestination
- mqReplyDestination





A sample API Policies for CICS

```
<ruleset name="CICS rules">
  <rule name="csmi-rule">
    <conditions>
      <header name="cicsMirror" 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>
```

GET.employee.{numb}

GET.employee.{numb}

HTTP Request

HTTP Headers

cicsMirror optional string

cicsConnection optional string

Path Parameters

numb Required string

Query Parameters

Body - cscvincServiceOperation

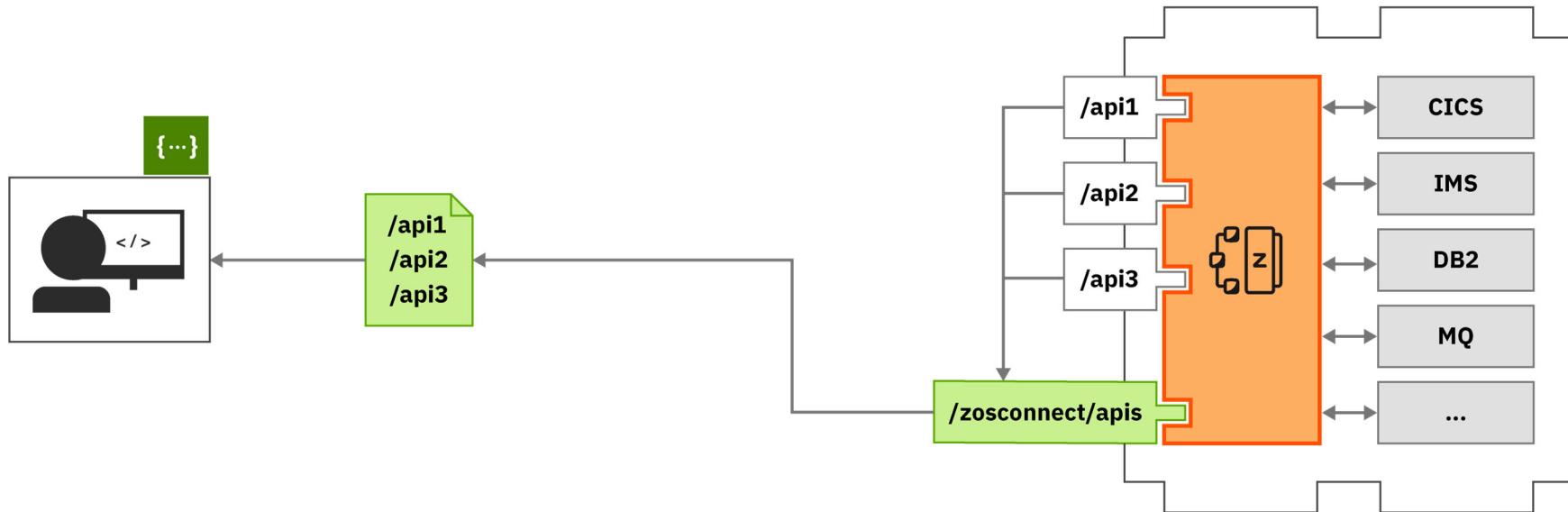
Curl

```
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



RESTful Administrative Interface for Services

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

Most administration tasks are supported by the RESTful administration interface

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

POST /zosConnect/services inquireSingle.sar

PUT /zosConnect/services/{serviceName}?status=started|stopped

PUT /zosConnect/services inquireSingle.sar

GET /zosConnect/services

GET /zosConnect/services/{serviceName}

DELETE /zosConnect/services/{serviceName}

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



RESTful Administrative Interface for APIs

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

Most administration tasks are supported by the RESTful administration interface

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

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



RESTful Administrative Interface for API Requesters

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

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

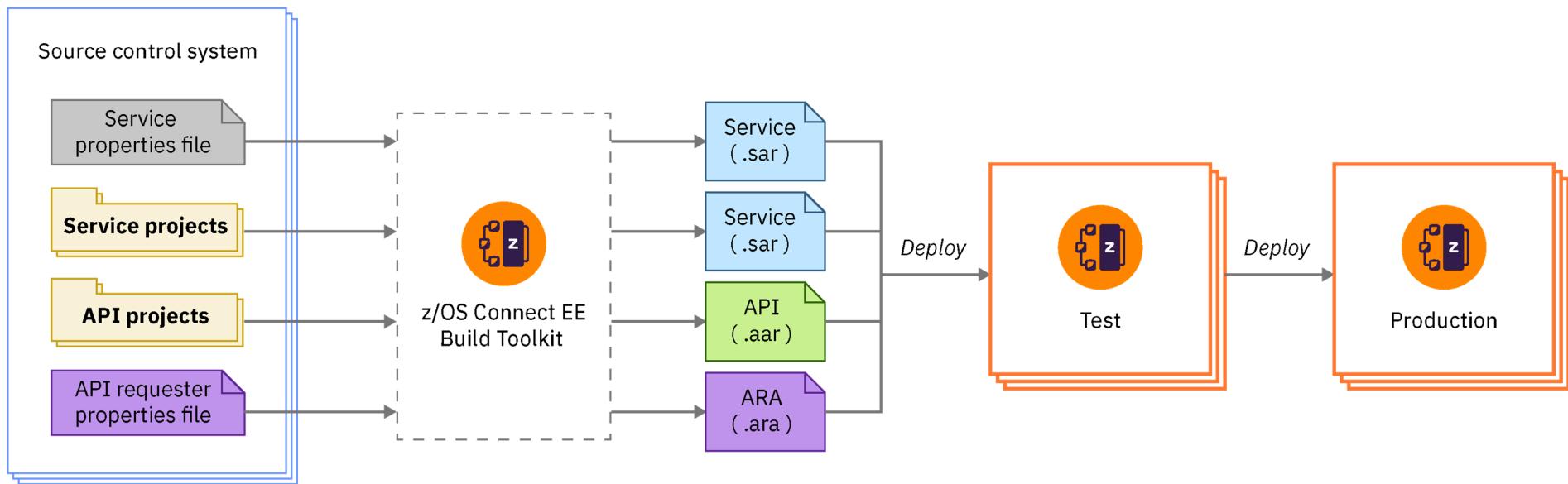
```
GET  /zosConnect/apiRequesters cscvinc.aar
PUT  /zosConnect/apiRequesters/{apiRequesterName}?status=started|stopped
PUT  /zosConnect/apiRequesters cscvinc.aar
GET  /zosConnect/apiRequesters
GET  /zosConnect/apiRequesters/{apRequesterName}
DELETE /zosConnect/apiRequesters
```



DevOps using z/OS Connect EE

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

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



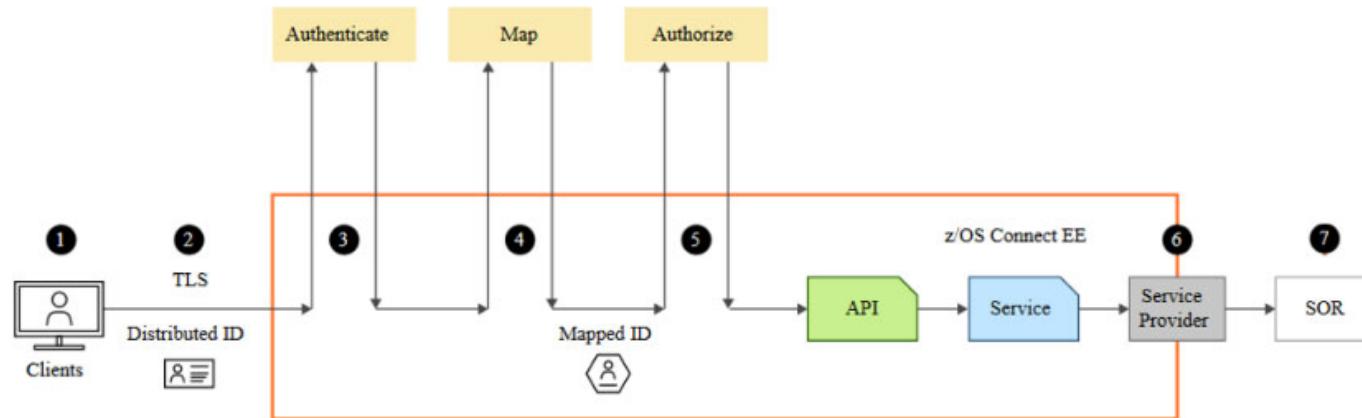


/security

How is security implement?



Typical z/OS Connect EE API Provider security flow



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



OPENAPI 3 roles

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

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

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

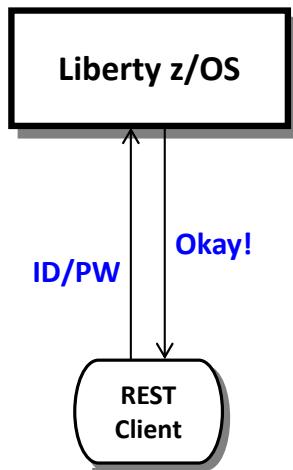
<authorization-roles id="Manager">
  <security-role name="Manager">
    <group name="managerGroup"/>
  </security-role>
</authorization-roles>
<authorization-roles id="Staff">
  <security-role name="Staff">
    <group name="staffGroup"/>
  </security-role>
</authorization-roles>
```



API Provider Authentication

Several different ways this can be accomplished:

Basic Authentication



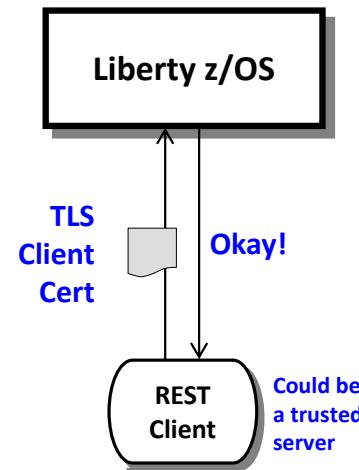
Server prompts for ID/PW

Client supplies ID/PW or
ID/Passticket

Server checks registry:

- Basic (server.xml)
- LDAP
- SAF

Client Certificate



Server prompts for cert.

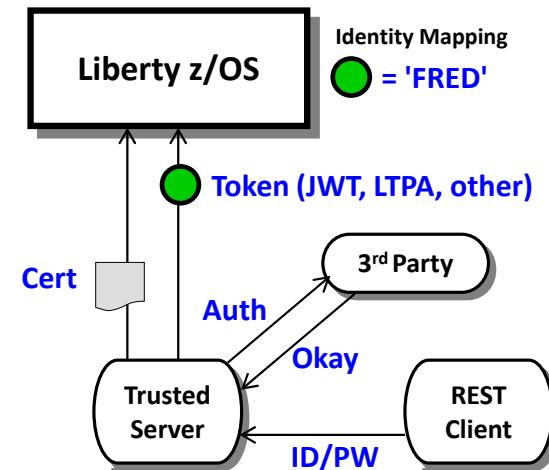
Client supplies certificate

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

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

Left Screenshot: UPS Sign Up

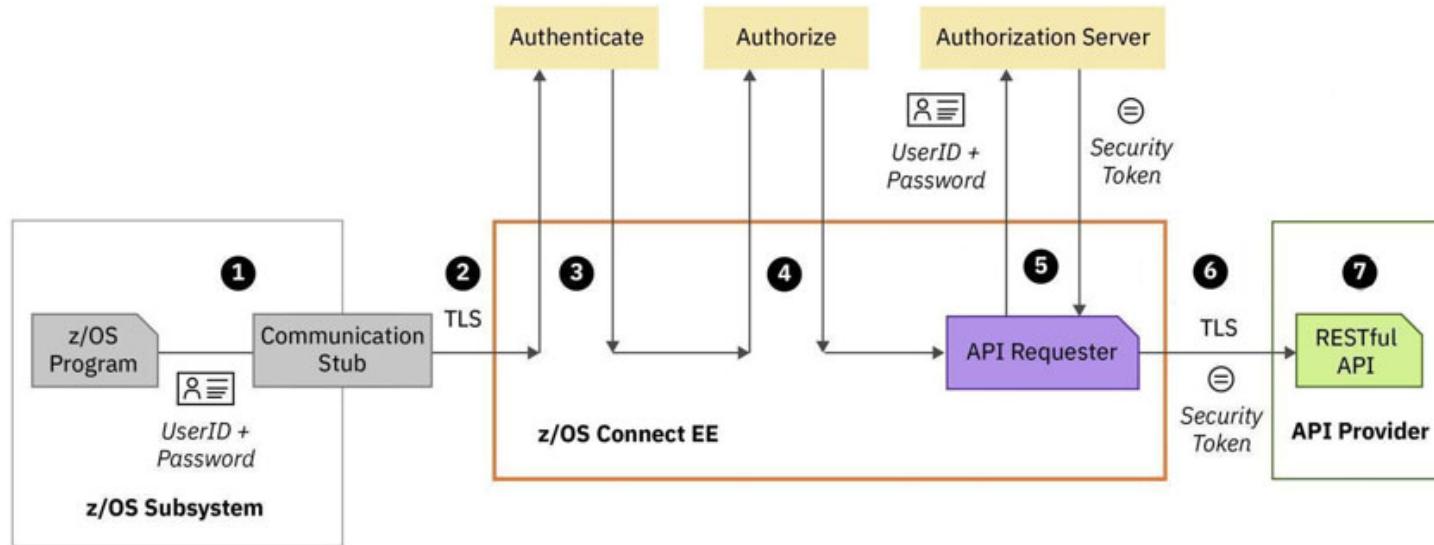
This screenshot shows the UPS Sign Up page. At the top, there's a banner stating "UPS is open for business: Service impacts related to Coronavirus ...More". Below the banner, the UPS logo is displayed. A "Sign Up / Log in" link and a "Search or Track" input field are visible. The main section is titled "Sign Up" and includes a link for users who already have an ID. It provides several social media integration options: Google, Facebook, Amazon, Apple, and Twitter. Below these, there are fields for entering personal information: Name*, Email*, User ID*, Password*, and Phone. The password field has a "Show" link next to it. A "Feedback" button is located on the right side of the form.

Right Screenshot: myNCDMV Log In

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



Typical z/OS Connect EE API Requester security flow



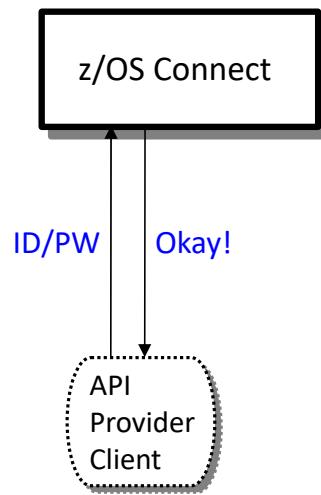
1. A user ID and password can be used for basic authentication by the z/OS Connect EE server
2. Connection between the CICS, IMS, or z/OS application and the z/OS Connect EE server can use TLS
3. Authenticate the CICS, IMS, or z/OS application.
4. Authorize the authenticated user ID to connect to z/OS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



z/OS Application to z/OS Connect API Requester

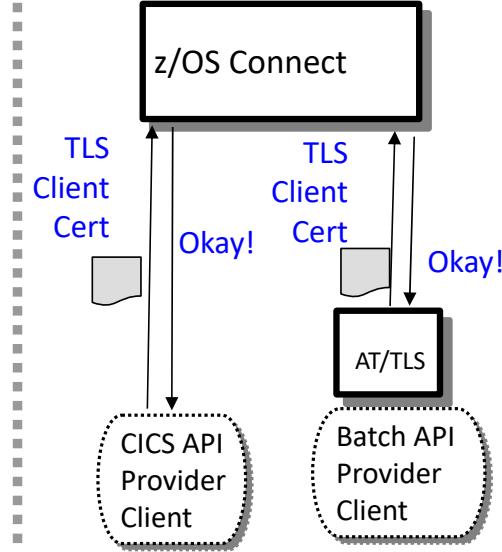
Two options for providing credentials for authentication

Basic Authentication



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

Client Certificate



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

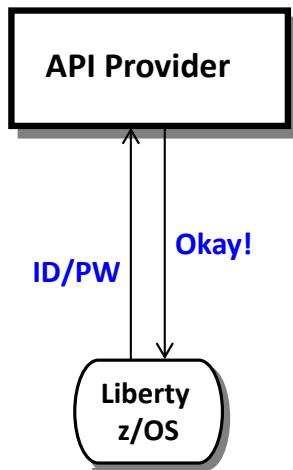
**CICS or AT/TLS supplies a
client certificate**



API Requester - API Provider Authentication

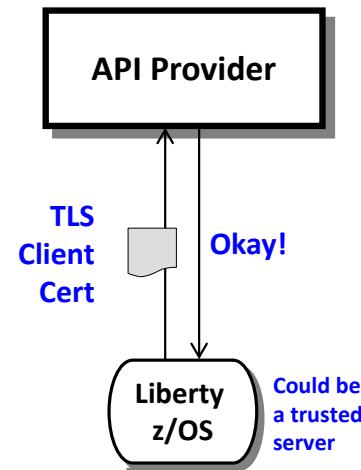
Several different ways this can be accomplished:

Basic Authentication



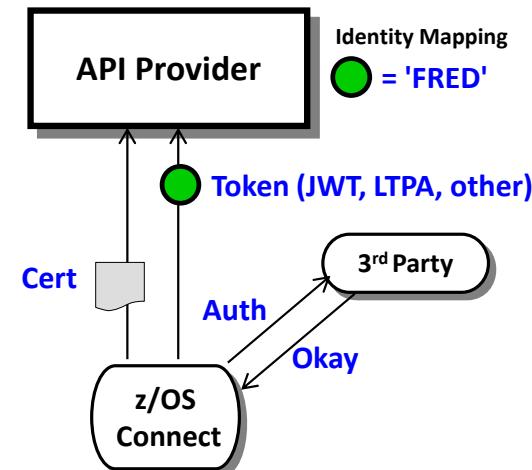
zCEE supplies ID/PW or
ID/Passticket

Client Certificate



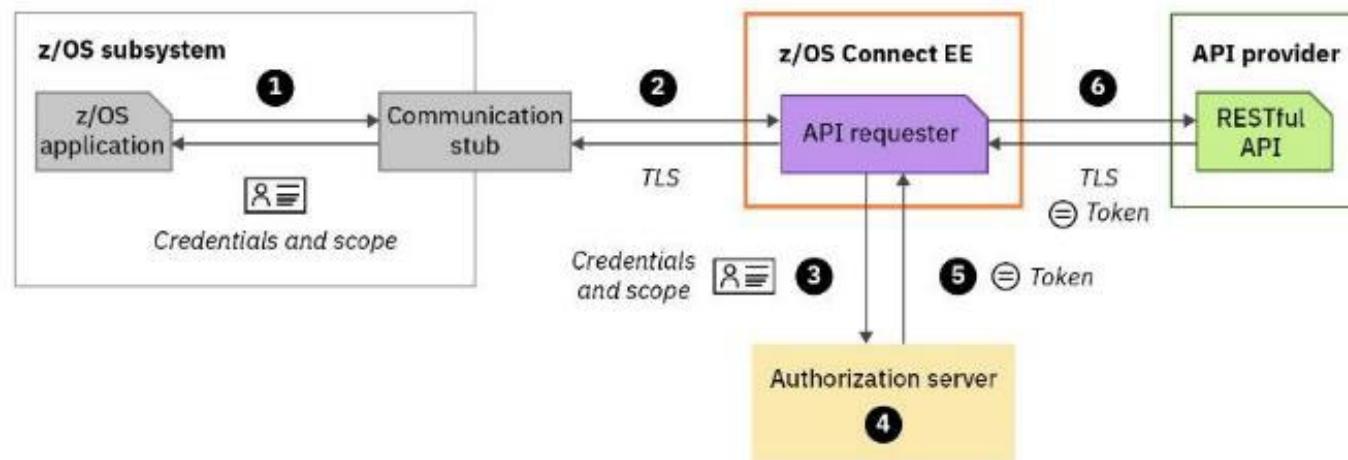
Server prompts for certificate
zCEE supplies certificate

Third Party Authentication

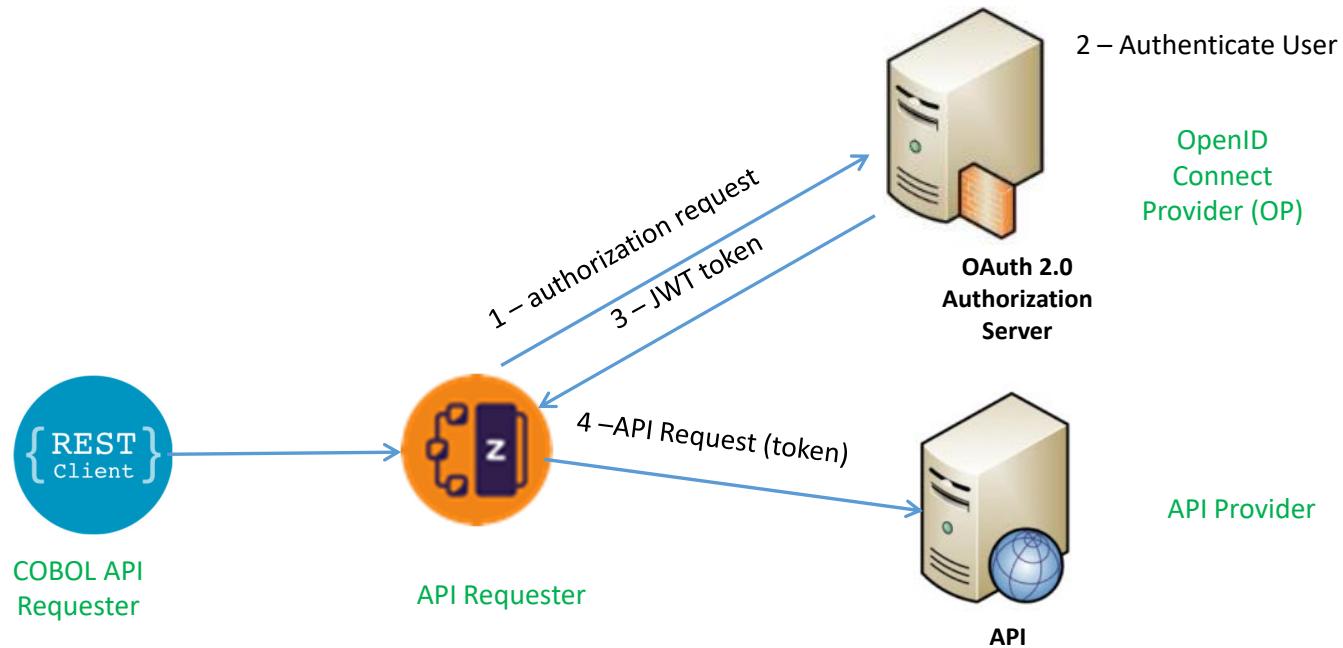


zCEE authenticates to 3rd party sever
zCEE receives a trusted 3rd party token
Token flows to API Provider

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/BdPRGD>



The screenshot displays two GitHub browser tabs. The left tab shows the main repository page for 'ibm-wsc/zCONNEE-Wildfire-Workshop'. It lists several branches: master, OpenAPI2, OpenAPI3, cobol, xml, Developing RESTful APIs f..., README.md, ZADMIN - zOS Connect..., ZCESEC - zOS Connect ..., ZCINTRO - Introduction t..., ZREQUEST - Introductio..., zOS Connect EE V3 Adva..., and zOS Connect EE V3 Getti... . The 'OpenAPI2' and 'OpenAPI3' branches are circled in red. The right tab shows the 'exercises' directory under the 'master' branch. It contains a large number of PDF files, all uploaded by user 'emitchj', including 'Developing CICS API Requester Applications.pdf', 'Developing IMS API Requester Applications.pdf', 'Developing MVS Batch API Requester Applications.pdf', 'Developing RESTful APIs for DVM VSAM Services.pdf', 'Developing RESTful APIs for DVM VSAMCICS Services.pdf', 'Developing RESTful APIs for Db2 REST Services.pdf', 'Developing RESTful APIs for HATS REST Services.pdf', 'Developing RESTful APIs for IMS Database REST Services....', 'Developing RESTful APIs for IMS Transactions.pdf', 'Developing RESTful APIs for MQ.pdf', 'Developing RESTful APIs for MVS Batch.pdf', 'Developing RESTful APIs for a CICS COMMAREA progra...', and 'Developing RESTful APIs for a CICS Container program.pdf'. The files were uploaded 20 days ago.

mitchj@us.ibm.com

- Contact your IBM representative to schedule access to these exercises

© 2018, 2022 IBM Corporation
Slide 163



Thank you for listening and your questions

And thank you for completing the Medallia survey