



# IBM z/OS Connect Enterprise Edition

## Introduction and Overview

Mitch Johnson

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

Washington System Center



# Agenda

- z/OS Connect Introduction and overview
- Discuss enabling RESTful API to various z/OS resources, e.g.
  - CICS
  - Db2
  - IMS/TM
  - IMS/DB
  - MQ
  - MVS Batch
  - Outbound REST APIs
  - IBM DVM
  - 3270 screen based applications (HATS)
  - IBM File Manager
- z/OS Connect Security

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



© 2018, 2020 IBM Corporation



**z/OS Connect EE**

CICS

IMS/TM

IMS/DB

Db2

MQ

IBM File Manager

3270

IBM DVM

MVS<sup>+</sup>

WAS

Custom\*

\* Other Vendors or your own implementation

## **/but\_first, what\_is\_REST?**

What makes an API “RESTful”?

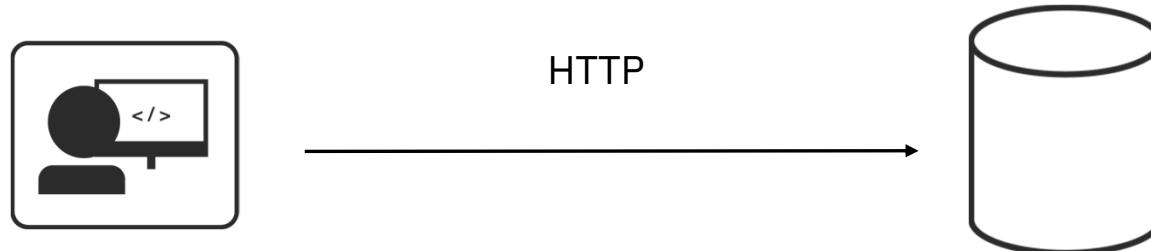
# REST is an Architectural Style

**REST** stands for **R**epresentational **S**tate **T**ransfer.

An architectural style for **accessing** and **updating** data.

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

GET  
POST  
PUT  
DELETE

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

Path and Query parameters are used for refinement of the request

URI path identifies a resource (or lists of resources)

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
- It should access all the resources from the server using only 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 JSON
- REST based services follow some of the above principles and not all, whereas RESTful means it follows all the above principles.
- Remember - Not all REST APIs are RESTful APIs
- The key is consistency, RESTful APIs are consistent, REST APIs are not

# RESTful Examples



z/OS Connect EE

## z/OS Connect Enterprise Edition:

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

**GET** /account?number=1234

**PUT** /account/1234 +  (*JSON 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 you are doing it wrong)

## 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 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" }

# Why is REST popular?

<b>Ubiquitous Foundation</b>	<p>It's based on HTTP, which operates on TCP/IP, which is a ubiquitous networking topology.</p>
<b>Relatively Lightweight</b>	<p>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.</p>
<b>Relatively Easy Development</b>	<p>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.</p>
<b>Increasingly Common</b>	<p>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.</p>
<b>Stateless</b>	<p>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.</p>

# **How do we describe a REST API?**



## **/swagger/open\_api**

The industry standard framework for describing RESTful APIs.

# Why use Swagger?

It is more than just an API framework



There are a number of tools available to aid consumption:

## Consume Swagger

**Swagger Codegen** create stub code to consume APIs from various languages



## Read Swagger

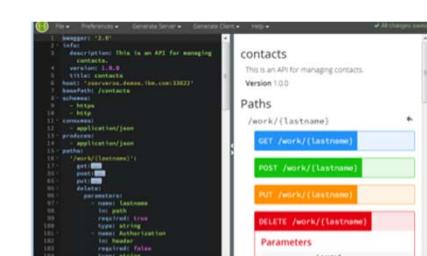
**Swagger UI** allows API consumers to easily browse and try APIs based on Swagger Doc.



The screenshot shows the Swagger UI interface for a 'contacts' API. It displays a list of operations under the 'default' path: GET /work/{lastname}, GET /work/{lastname}, POST /work/{lastname}, and PUT /work/{lastname}. The UI includes a sidebar with navigation links like 'Explore', 'ShowHide', and 'Expand Operations'. At the bottom, it shows the base URL as 'https://server001.demolic.com:33622/contacts/api-docs' and the API version as '1.0.0'.

## Write Swagger

**Swagger Editor** allows API developers to design their swagger documents.



The screenshot shows the Swagger Editor interface. On the left, there is a JSON editor pane displaying the Swagger specification for managing contacts. On the right, there is a preview pane showing the 'Paths' section for the '/work/{lastname}' endpoint, which includes four operations: GET, POST, PUT, and DELETE, each with its respective HTTP method and path parameters.

<https://blog.readme.io/what-is-swagger-and-why-it-matters/>



# Swagger Example

The image shows two side-by-side Swagger UI interfaces, each displaying a JSON API definition for a "Miniloan" service.

**Left Window (API Definition):**

- host:** `"localhost:8080"`
- basePath:** `"/miniloan"`
- parameters:** A parameter named `Authorization` is highlighted with a red oval.
- responses:** A response code `200` is shown.

**Right Window (Request Body Schema):**

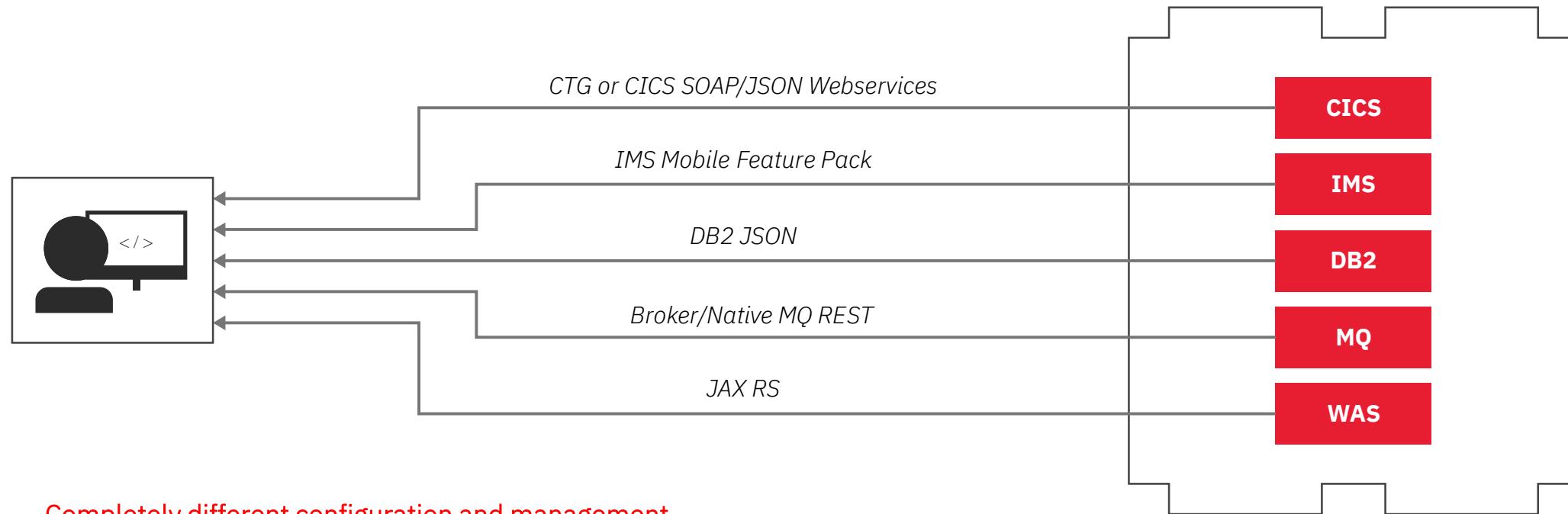
- definitions:**
  - postMiniloanService\_request:** A schema for the request body is defined, highlighted with a large red oval.
    - properties:**
      - MINILOAN\_COMMAREA:** An object type with properties:
        - name:** Type: string, maxLength: 20, creditscore: integer (range 0-10000000000000000)
        - yearlyIncome:** Type: integer (range 0-10000000000000000)
        - age:** Type: integer (range 0-999999999)
        - amount:** Type: integer (range 0-10000000000000000)
        - effectiveDate:** Type: string, maxLength: 8
        - yearlyRepayment:** Type: integer (range 0-10000000000000000)
  - responses:** A response code `200` is shown.



## **Why /zos\_connect\_ee?**

Truly RESTful APIs to and from your mainframe.

# Could we not do REST before z/OS Connect? Yes, but....



Completely different configuration and management.

Multiple endpoints for developers to call/maintain access to.

These are typically not RESTful!

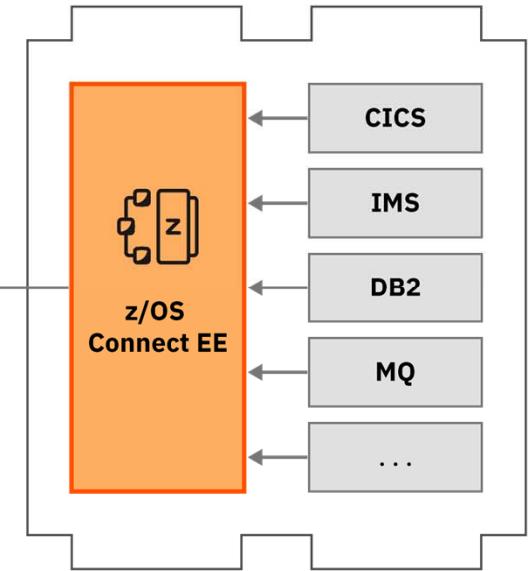
# A single entry point was needed

Expose 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.

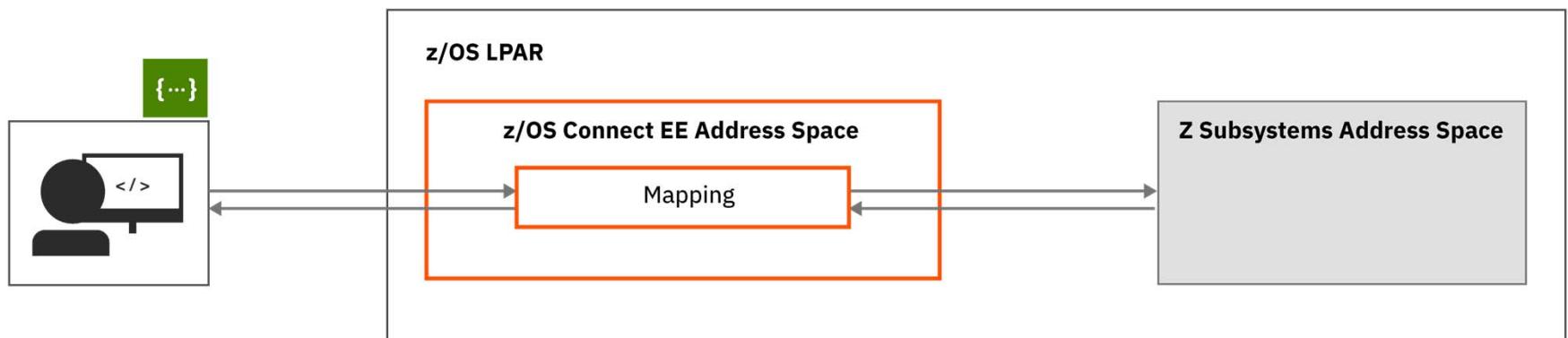




**Other than a RESTful interface,  
what does z/OS Connect provide?**

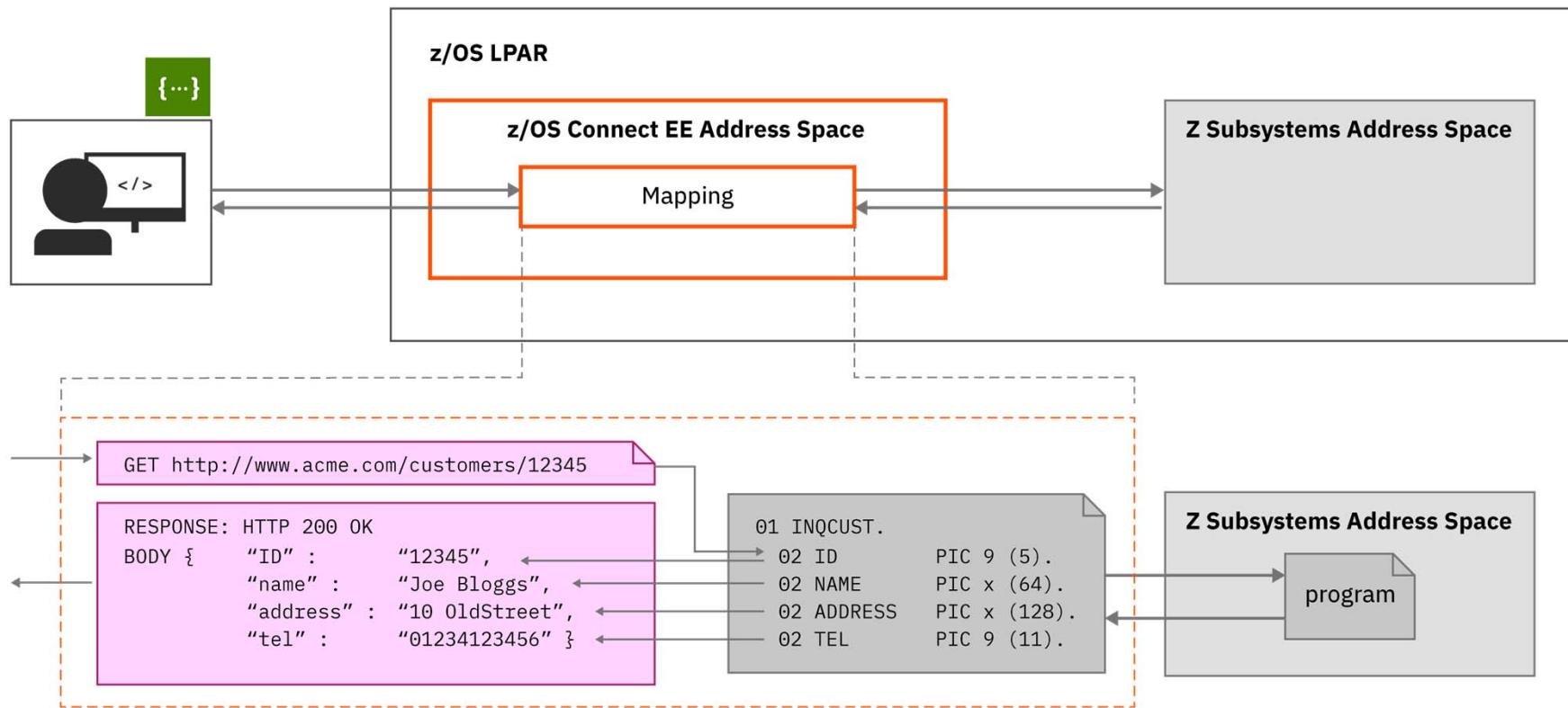
# Let's Start with Data mapping

Converting JSON to the target's subsystem's normal format



# Data mapping Example

A closer look



# COBOL versus JSON Example



```

01 MINILOAN-COMMAREA.
 10 name pic X(20).
 10 creditScore pic 9(16)v99.
 10 yearlyIncome pic 9(16)v99.
 10 age pic 9(10).
 10 amount pic 9999999v99.
 10 approved pic X.
    88 BoolValue value 'T'.
 10 effectDate pic X(8).
 10 yearlyInterestRate pic S9(5).
 10 yearlyRepayment pic 9(18).
 10 messages-Num pic 9(9).
 10 messages pic X(60) occurs 1 to 10 times
      depending on messages-Num.

```

```

"MINILOAN_COMMAREA" : {
  "type" : "object",
  "properties" : {
    "NAME" : {
      "maxLength" : 20,
      "type" : "string"
    },
    "CREDITSCORE" : {
      "multipleOf" : 0.01,
      "minimum" : 0,
      "maximum" : 99999999999999.99,
      "type" : "number",
      "format" : "decimal"
    },

```

```

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ns2:message xmlns:ns2="http://www.ibm.com/ims/Transaction" transactionCode="" messageName="miniloanRequest" direction="0" serviceType="0">
  <message id="1" name="miniloanRequest">
    <segment id="1" name="COMMAREA" originalName="COMMAREA" included="Y" path="MINILOAN_COMMAREA">
      <field name="NAME" originalName="NAME" included="Y" path="MINILOAN_COMMAREA.NAME">
        <startPos>1</startPos>
        <bytes>20</bytes>
        <maxBytes>20</maxBytes>
        <applicationDatatype datatype="CHAR"/>
      </field>
      <field name="CREDITSCORE" originalName="CREDITSCORE" included="Y" path="MINILOAN_COMMAREA.CREDITSCORE">
        <startPos>21</startPos>
        <bytes>18</bytes>
        <maxBytes>18</maxBytes>
        <marshaller isSigned="N" isSignLeading="N" isSignSeparate="N" isWCHAROnly="N">
          <typeConverter>ZONEDDECIMAL</typeConverter>
        </marshaller>
        <applicationDatatype datatype="DECIMAL" precision="18" scale="2"/>
      </field>
    </segment>
  </message>
</ns2:message>

```

“name”：“Mitch Johnson”，  
“creditScore”：“72000”

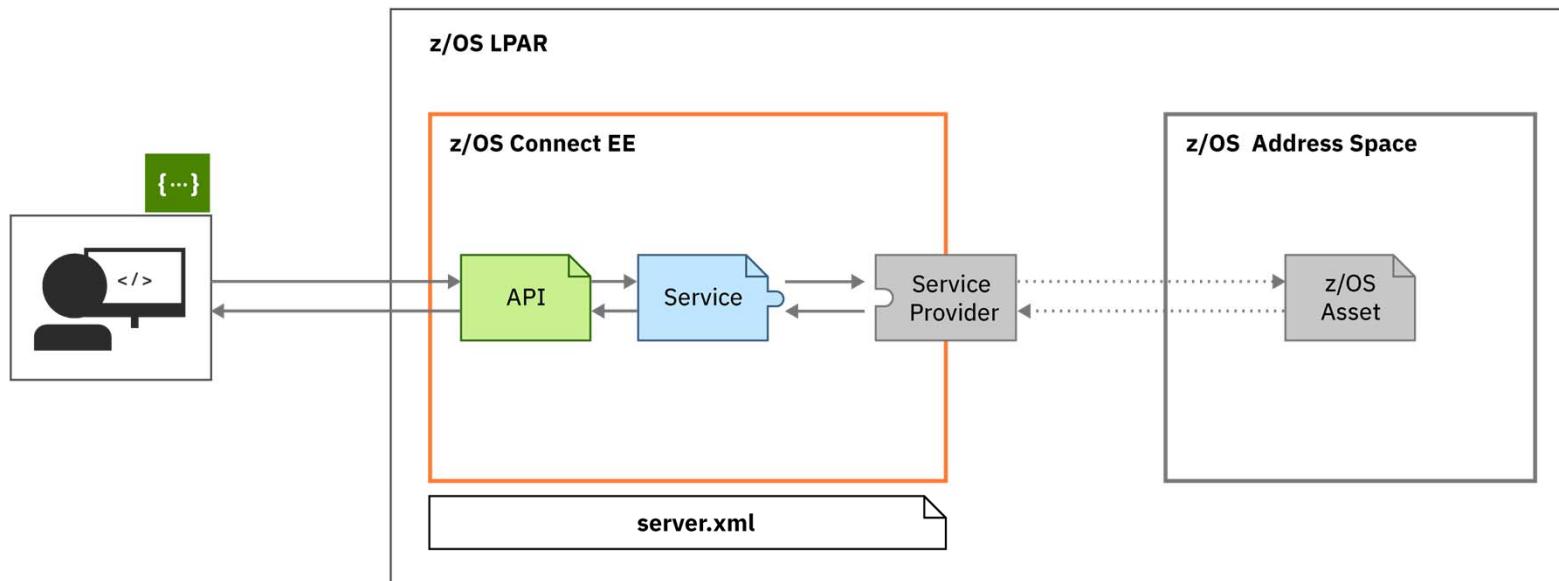
All data is sent as character strings,  
removing the big v. little endian and +/-  
issue

## Slide 21

---

**MJ1** Mitch Johnson, 6/16/2020

# Steps to expose a z/OS application



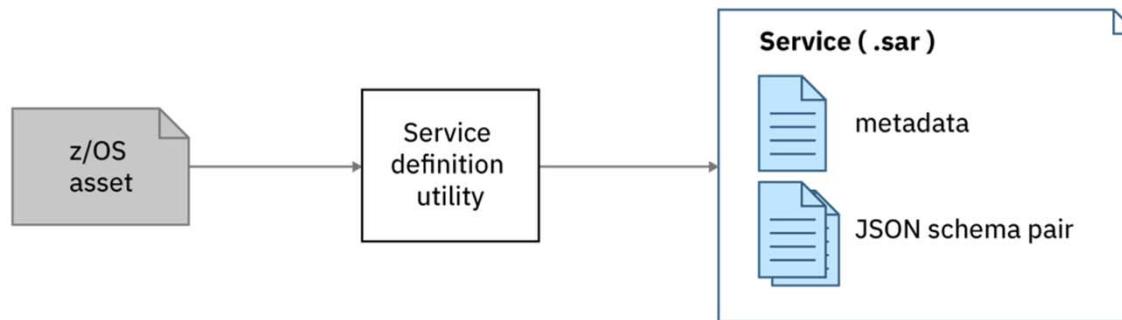
- The API provides the RESTful interface is ready to be consumed by a client and it requires no knowledge that a z/OS resource is being accessed
- The Service provides meta data specific to the z/OS Asset (e.g. CICS program, MQ queue manager, etc.)
- The Service Provider is tightly coupled to a specific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

# Steps to expose a z/OS application

## 1. Create a service

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



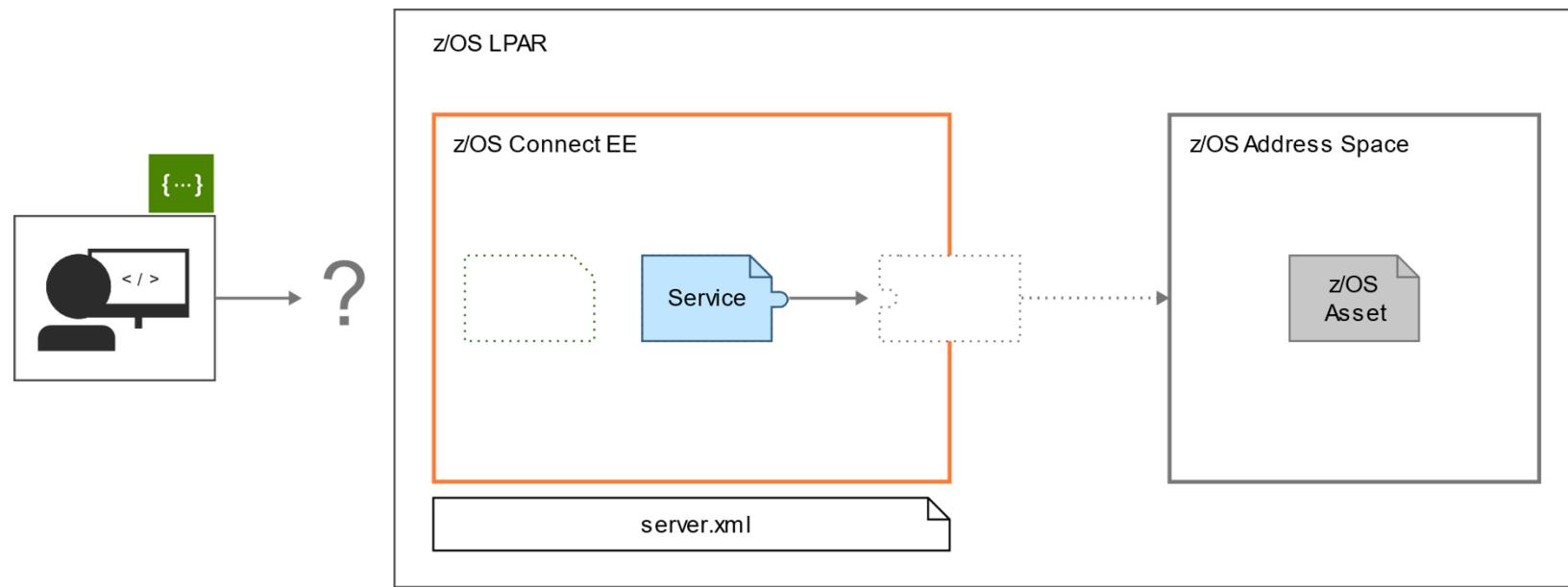
Use a system-appropriate utility to generate a `.sar` file for the z/OS application

- 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

 [ibm.biz/zosconnect-sar-creation](http://ibm.biz/zosconnect-sar-creation)

# Steps to expose a z/OS application

## 2. Deploy and export the service

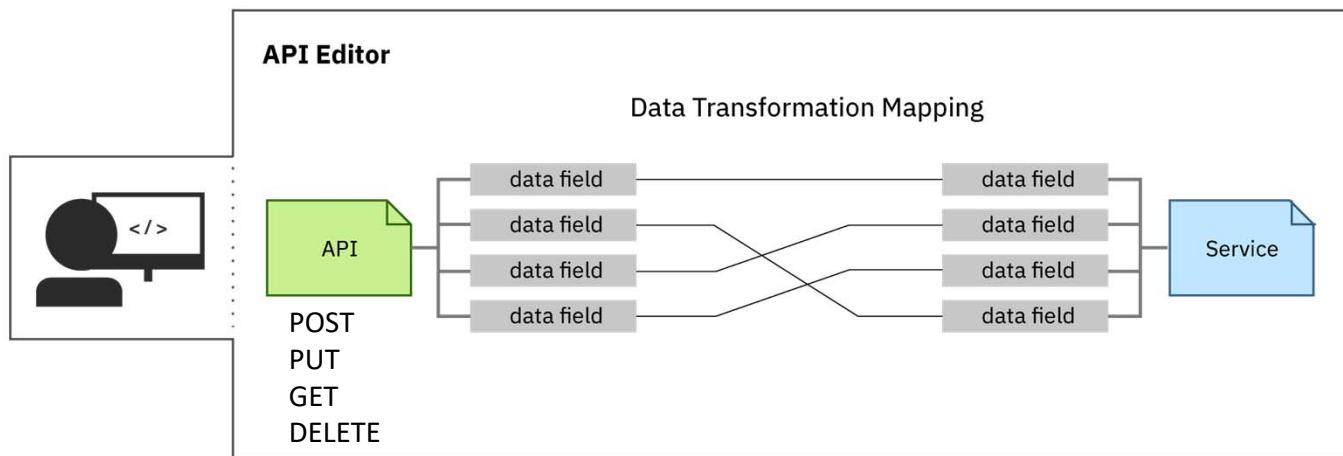


Deploy the service archive file generated in **Step 1** using the right-click deploy in **the API toolkit**.

 [ibm.biz/zosconnect-define-services](http://ibm.biz/zosconnect-define-services)

# Steps to expose a z/OS application

## 3. Create an API using exported services

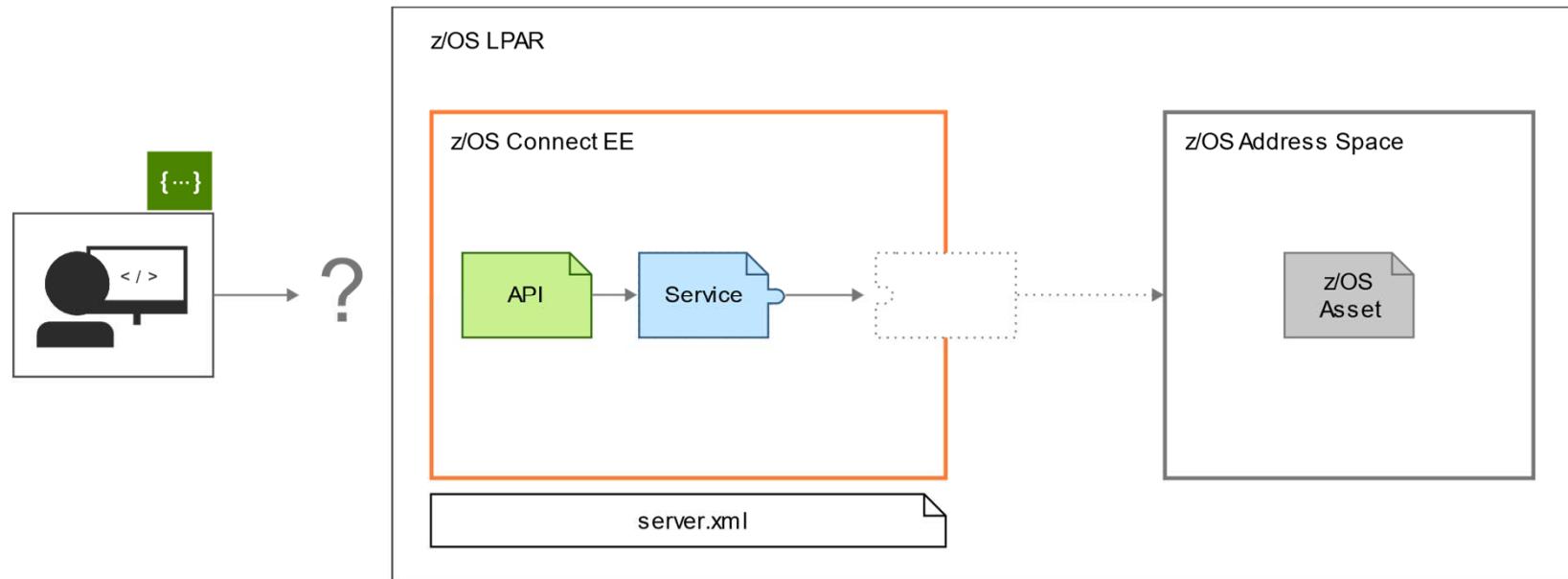


- Import the service archive file into the **API toolkit**, and start designing the RESTful API.
- Provides additional data mapping
- Use the editor to describe the API and how it maps to underlying services.

 [ibm.biz/zosconnect-create-api](http://ibm.biz/zosconnect-create-api)

# Steps to expose a z/OS application

## 4. Deploy the API

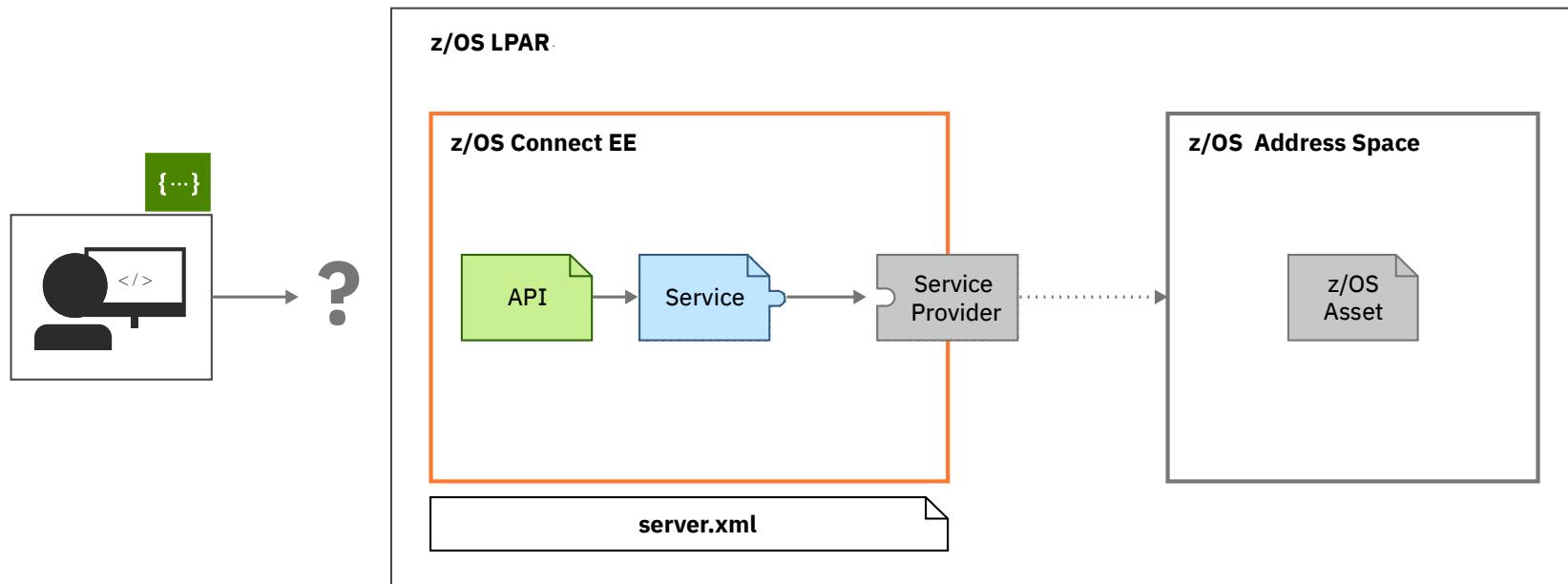


Deploy your API using the right-click deploy in **the API toolkit**

 [ibm.biz/zosconnect-deploy-api](http://ibm.biz/zosconnect-deploy-api)

# Steps to expose a z/OS application

## 5. Configure the service provider

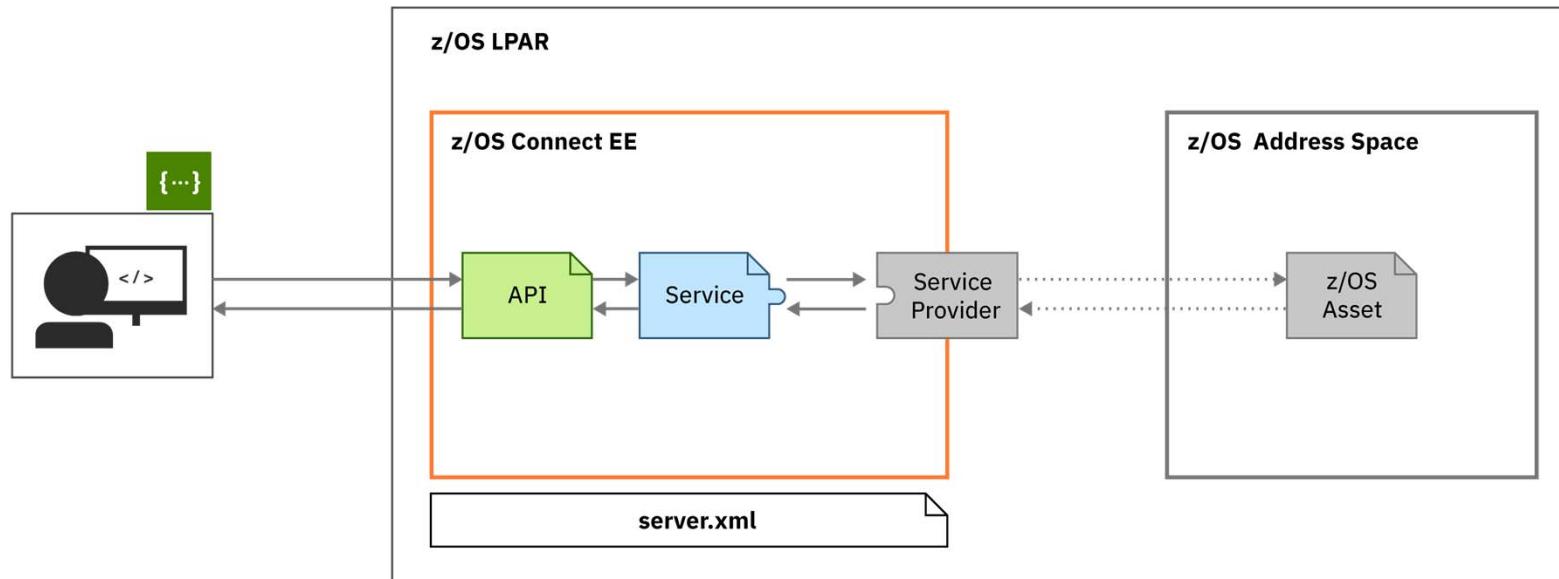


Configure the system-appropriate service provider to connect to your backend system in your `server.xml`.

 [ibm.biz/zosconnect-configuring](http://ibm.biz/zosconnect-configuring)

# Steps to expose a z/OS application

## 6. Done



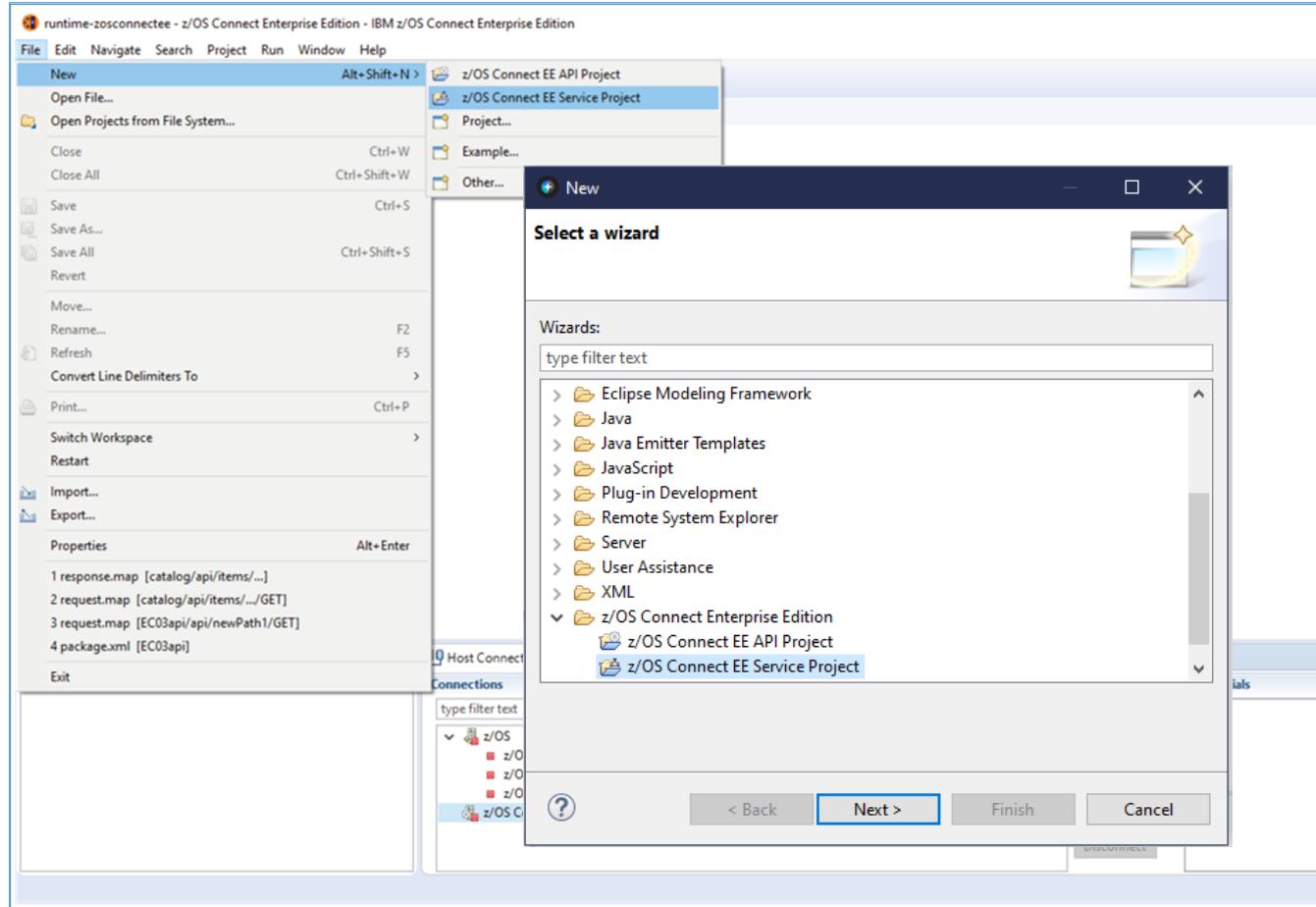
- The API is ready to be consumed and requires no knowledge that a z/OS resource is being accessed
- The Service provides meta data specific to the z/OS Asset (e.g. CICS program, MQ queue manager, etc.)
- The Service Provider is tightly coupled to a specific instance of a resource (e.g. host and port)



## /api\_toolkit/services

Simple **service creation.**

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

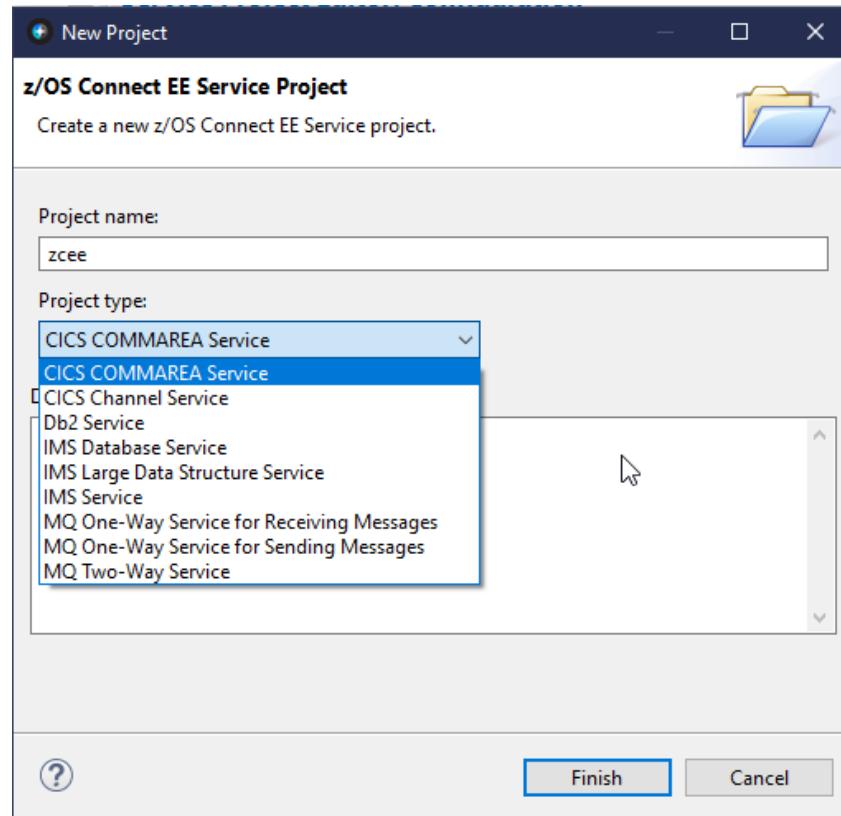


Use the **API toolkit** to create services through Eclipse-based tooling.

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

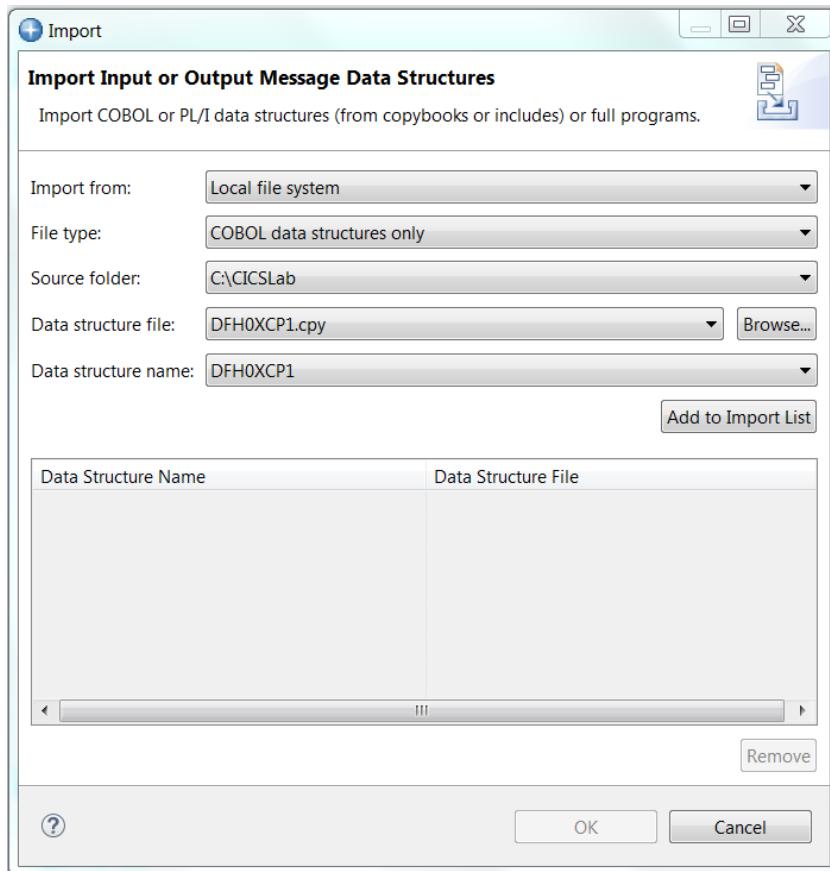
## Service creation – a common interface



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

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

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



Start by importing data structures into the service interface from the local file system or the workspace.

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 service interface definition

\*inquireSingle Service \*inquireSingleRequest

## Service Interface Definition

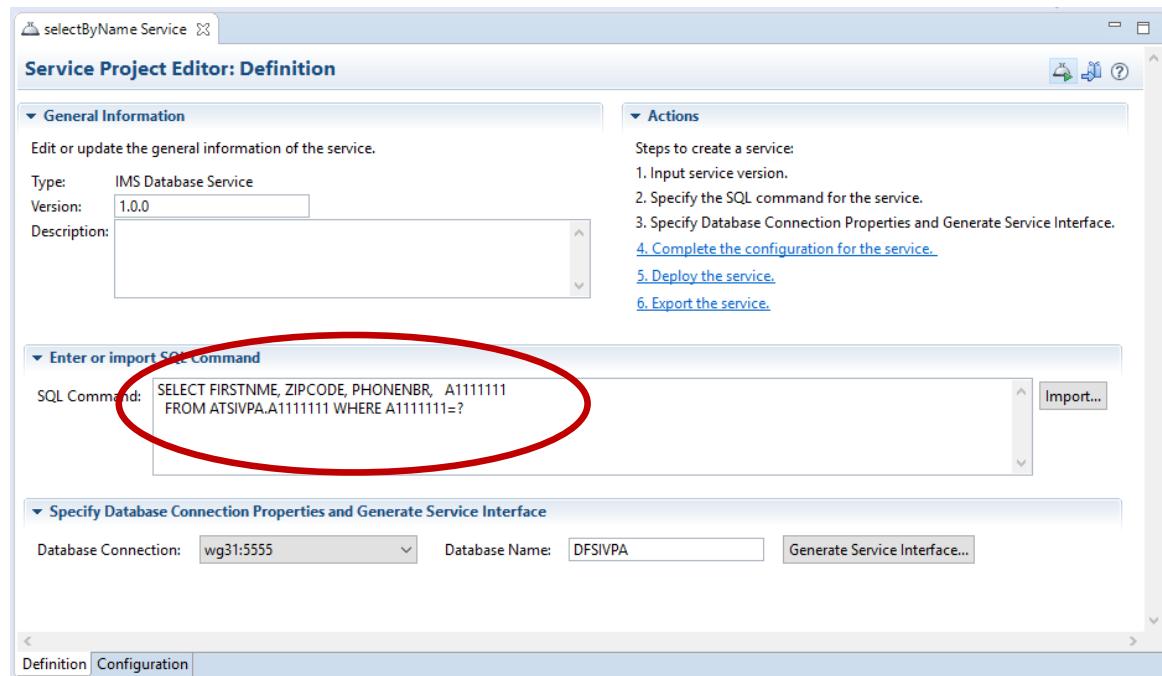
Define and customize your request and response service interfaces. Right-click a row and select the appropriate action from the context menu, or select a row and click the appropriate button.

Fields	Include	Interface rename	Default Field Value	Data Type	Field Length	Start Byte	
COMMAREA							
DFH0XCP1							
CA_REQUEST_ID	<input type="checkbox"/>	CA_REQUEST_ID	01INQS	CHAR	6	1	
CA_RETURN_CODE	<input type="checkbox"/>	CA_RETURN_CODE		DECIMAL	2	7	
CA_RESPONSE_MESSAGE	<input type="checkbox"/>	CA_RESPONSE_MESSAGE		CHAR	79	9	
CA_REQUEST_SPECIFIC (Redefine)	<input type="checkbox"/>	CA_REQUEST_SPECIFIC		CHAR	911	88	
CA_INQUIRE_REQUEST redefines C	<input type="checkbox"/>	CA_INQUIRE REQUEST		STRUCT	911	88	
CA_INQUIRE_SINGLE redefines C	<input checked="" type="checkbox"/>	inquireSingle		STRUCT	911	88	
CA_ITEM_REF_REQ	<input checked="" type="checkbox"/>	itemID		DECIMAL	4	88	
FILL_0	<input type="checkbox"/>	FILL_0		DECIMAL	4	92	
FILL_1	<input type="checkbox"/>	FILL_1		DECIMAL	3	96	
CA_SINGLE_ITEM	<input type="checkbox"/>	CA_SINGLE_ITEM		STRUCT	60	99	
FILL_2	<input type="checkbox"/>	FILL_2		CHAR	840	159	
CA_ORDER_REQUEST redefines C	<input type="checkbox"/>	CA_ORDER_REQUEST		STRUCT	911	88	

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.

# API toolkit – Creating Services for IMS DB

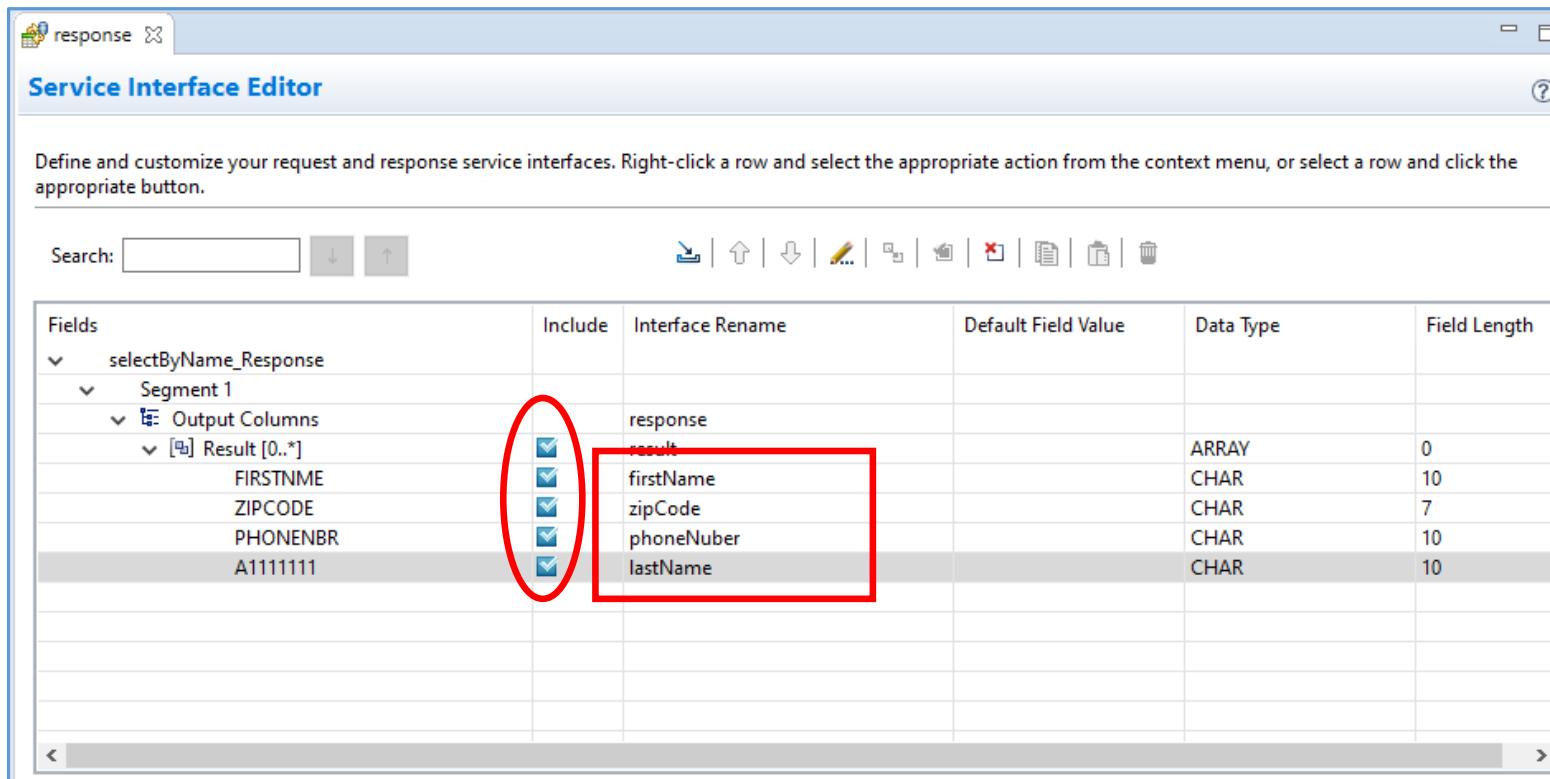
## Creating a service project from the IMS Catalog



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

## API toolkit – Creating Services for IMS DB

Again allows editing a service interface definition\*



The screenshot shows the 'Service Interface Editor' window. The left pane displays a tree view of fields under 'selectByName\_Response' and 'Segment 1'. The right pane is a table with columns: Fields, Include, Interface Rename, Default Field Value, Data Type, and Field Length. The table rows represent the fields from the tree view. A red circle highlights the checkbox for 'result' in the 'Include' column. A red rectangle 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			
FIRSTMNAME	<input checked="" type="checkbox"/>	result		ARRAY	0
ZIPCODE	<input checked="" type="checkbox"/>	firstName		CHAR	10
PHONENBR	<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

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

## Editing a response message

\*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.

Fields	Include	Interface rename	Default Field Value	Data Type	Field Length	Start Byte
COMMAREA						
DFH0XCP1						
CA_REQUEST_ID	<input type="checkbox"/>	CA.REQUEST_ID		CHAR	6	1
CA_RETURN_CODE	<input checked="" type="checkbox"/>	returnCode		DECIMAL	2	7
CA_RESPONSE_MESSAGE	<input checked="" type="checkbox"/>	responseMessage		CHAR	79	9
CA_REQUEST_SPECIFIC (Redefine)	<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 checked="" type="checkbox"/>	inStock		DECIMAL	4	152
ON_SNGL_ORDER	<input checked="" type="checkbox"/>	onOrder		DECIMAL	3	156
FILL_2	<input type="checkbox"/>	FILL_2		CHAR	840	159
CA_ORDER_REQUEST redefines CA_INQUIRE_ORDER	<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

© 2018, 2020 IBM Corporation

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

# API toolkit – Creating Services for CICS



## Creating multiple services to the same resource

\*cscvincSelectService Service \*cscvincSelectRequest

**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
Channel		CSCCINCCContainer			
@ Container1		REQUEST_CONTAINER			
ACTION	<input type="checkbox"/>	ACTION	S	CHAR	1
USERID	<input type="checkbox"/>	USERID		CHAR	8
FILEA_AREA	<input checked="" type="checkbox"/>	FILEA_AREA		STRUCT	80
STAT	<input type="checkbox"/>	STAT		CHAR	1
NUMB	<input checked="" type="checkbox"/>	NUMB		CHAR	6
NAME	<input type="checkbox"/>	NAME		CHAR	20
ADDRX	<input type="checkbox"/>	ADDRX		CHAR	20
PHONE	<input type="checkbox"/>	PHONE		CHAR	8
DATEX	<input type="checkbox"/>	DATEX		CHAR	8
AMOUNT	<input type="checkbox"/>	AMOUNT		CHAR	8
COMMENT	<input type="checkbox"/>	COMMENT		CHAR	9

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

\*cscvincSelectService Service \*cscvincInsertRequest

**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
Channel		cscvincInsertContainer			
@ Container1		REQUEST_CONTAINER			
ACTION	<input type="checkbox"/>	ACTION	I	CHAR	1
USERID	<input type="checkbox"/>	USERID		CHAR	8
FILEA_AREA	<input checked="" type="checkbox"/>	FILEA_AREA		STRUCT	80
STAT	<input checked="" type="checkbox"/>	status		CHAR	1
NUMB	<input checked="" type="checkbox"/>	employeeNumber		CHAR	6
NAME	<input checked="" type="checkbox"/>	employeeName		CHAR	20
ADDRX	<input checked="" type="checkbox"/>	address		CHAR	20
PHONE	<input checked="" type="checkbox"/>	phoneNumber		CHAR	8
DATEX	<input checked="" type="checkbox"/>	startDate		CHAR	8
AMOUNT	<input checked="" type="checkbox"/>	amount		CHAR	8
COMMENT	<input checked="" type="checkbox"/>	comment		CHAR	9

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.

**CICS Meta Data**

**Service Project Editor: Definition**

General Information

Type: CICS Channel Service  
Version: 1.0.0  
Description:

Actions

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

Program: CSCVINC (circled)

**Define Request and Response Service Interface**

Create new request and response service interface  
Create Service Interface... Import

Request service interface: cscvinc  
Response service interface: cscvinc

Set advanced data conversion options Advanced

**Service Project Editor: Configuration**

Required Configuration

Enter the required configuration for this service.  
Coded character set identifier (CCSID): 37  
Connection reference: cscvinc

Optional Configuration

Enter the optional configuration for this service.  
Transaction ID: CSMI (circled)  
Transaction ID usage: EIB\_ONLY (circled)  
Use context containers:   
Context containers HTTP headers: Add another

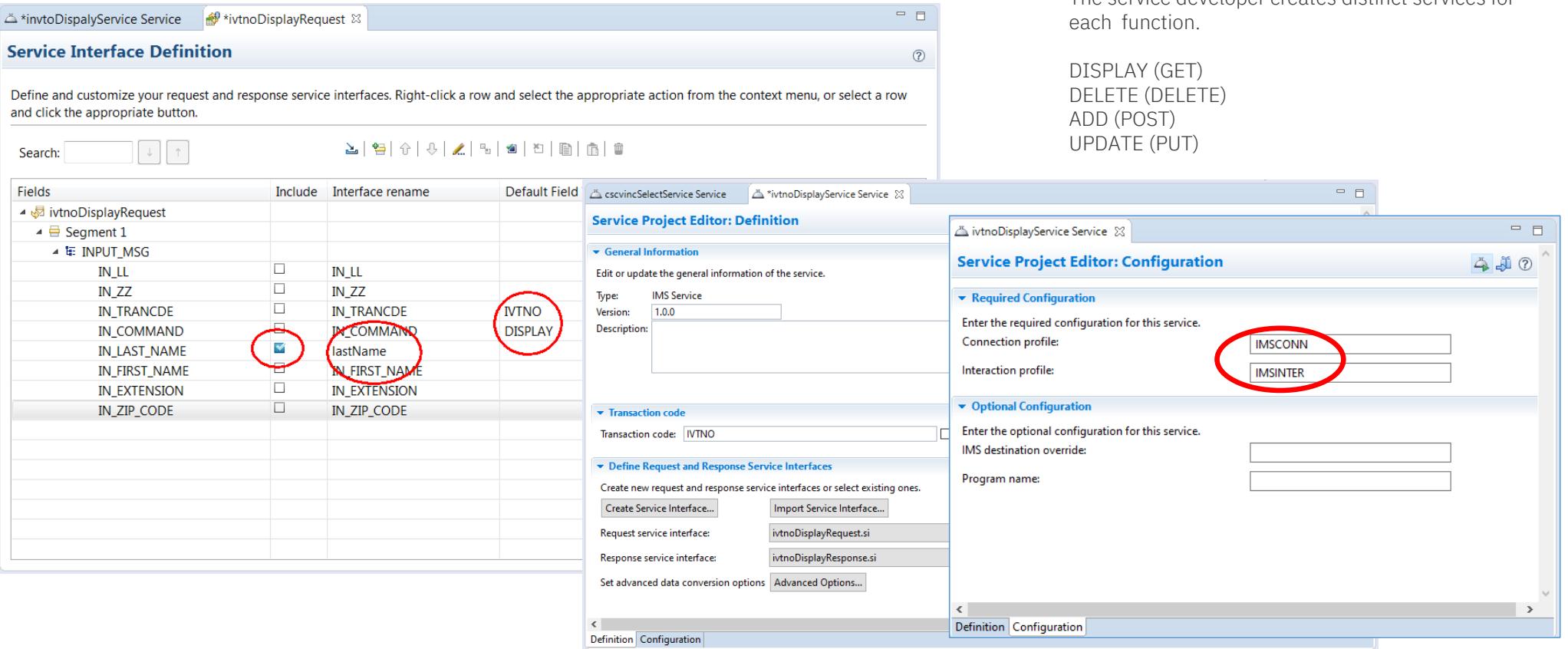
The service developer creates distinct services for each function by setting the ACTION field to S for select, I for insert, U for update or D for delete

# API toolkit – Creating Services for IMS

## Creating a “GET” service interface request definition

The service developer creates distinct services for each function.

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



**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.

**Fields**

Fields	Include	Interface rename	Default Field
ivtnoDisplayRequest			
Segment 1			
INPUT_MSG			
IN_LL	<input type="checkbox"/>	IN_LL	
IN_ZZ	<input type="checkbox"/>	IN_ZZ	
IN_TRANCDE	<input type="checkbox"/>	IN_TRANCDE	
IN_COMMAND	<input type="checkbox"/>	IN_COMMAND	IVTNO DISPLAY
IN_LAST_NAME	<input checked="" type="checkbox"/>	lastName	
IN_FIRST_NAME	<input type="checkbox"/>	IN_FIRST_NAME	
IN_EXTENSION	<input type="checkbox"/>	IN_EXTENSION	
IN_ZIP_CODE	<input type="checkbox"/>	IN_ZIP_CODE	

**Service Project Editor: Definition**

**General Information**

Type: IMS Service  
 Version: 1.0.0  
 Description:

**Transaction code**

Transaction code: IVTNO

**Define Request and Response Service Interfaces**

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

**Service Project Editor: Configuration**

**Required Configuration**

Enter the required configuration for this service.

Connection profile: IMSCONN  
 Interaction profile: IMSINTER

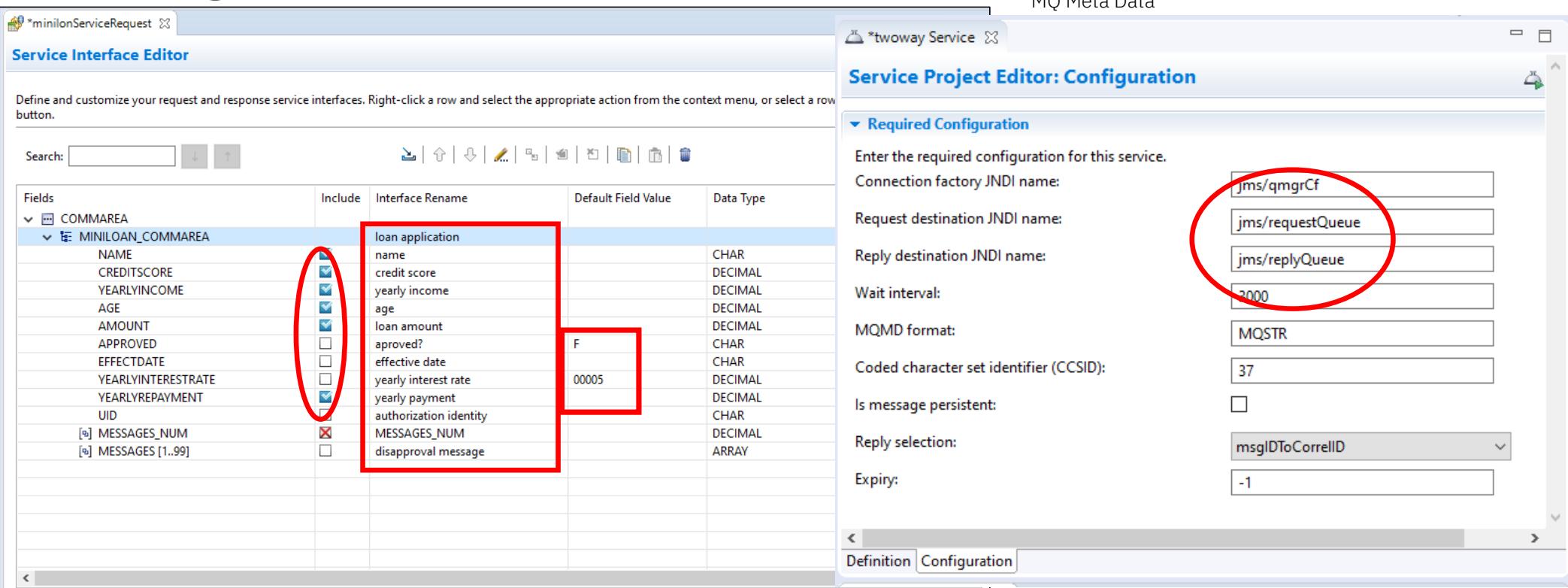
**Optional Configuration**

Enter the optional configuration for this service.

IMS destination override:  
 Program name:

# API toolkit – Creating Services for MQ

## Creating a service interface definition



The screenshot shows two windows side-by-side:

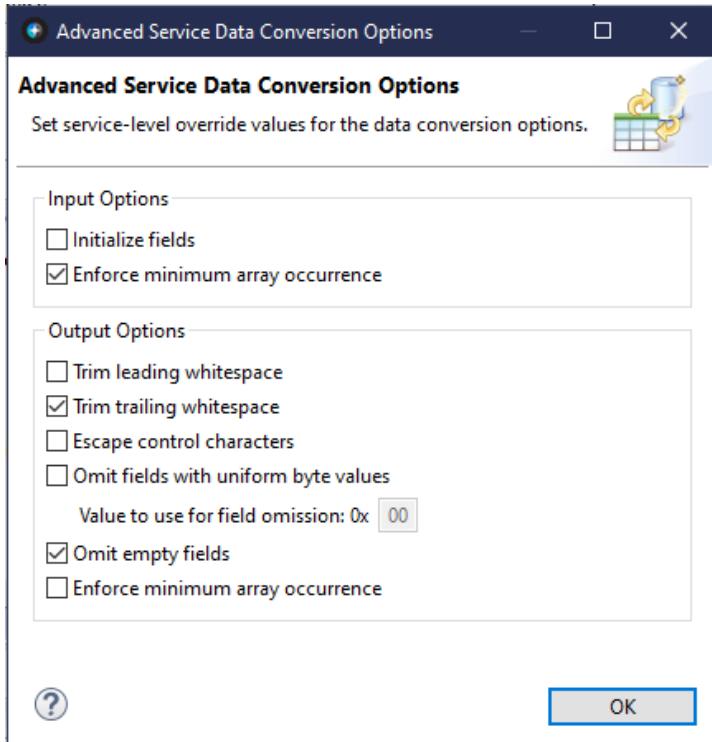
- Service Interface Editor:** This window displays a table of fields for a service interface named "minilnServiceRequest". The table columns are: Fields, Include, Interface Rename, Default Field Value, and Data Type. A red box highlights the "Interface Rename" column for the "loan application" row, which contains the value "name". Another red box highlights the "Default Field Value" column for the same row, which contains the value "F".
- Service Project Editor: Configuration:** This window shows configuration settings for a service named "twoWay Service". A red circle highlights several JNDI name fields:
  - Connection factory JNDI name: jms/qmgrCf
  - Request destination JNDI name: jms/requestQueue
  - Reply destination JNDI name: jms/replyQueue
  - Wait interval: 2000
  - MQMD format: MQSTR
  - Coded character set identifier (CCSID): 37
  - Is message persistent: (checkbox)
  - Reply selection: msgIDToCorrelID
  - 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.



z/OS Connect EE

# API toolkit – Advanced Data Conversion Options



## Request Messages:

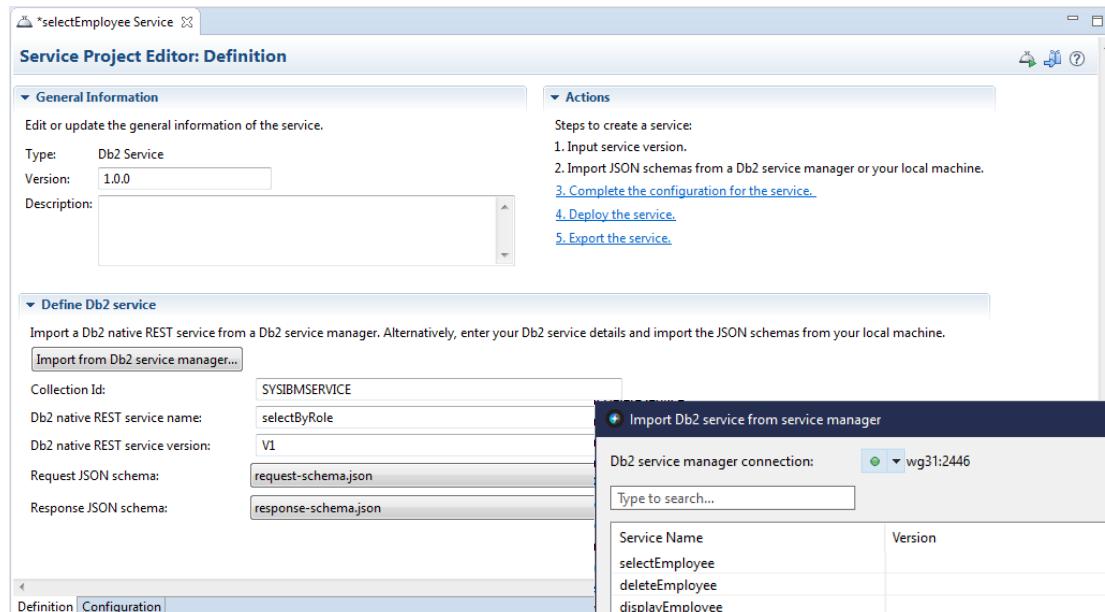
- Initialize fields
- Enforce minimum array occurrence

## 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

 \*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**

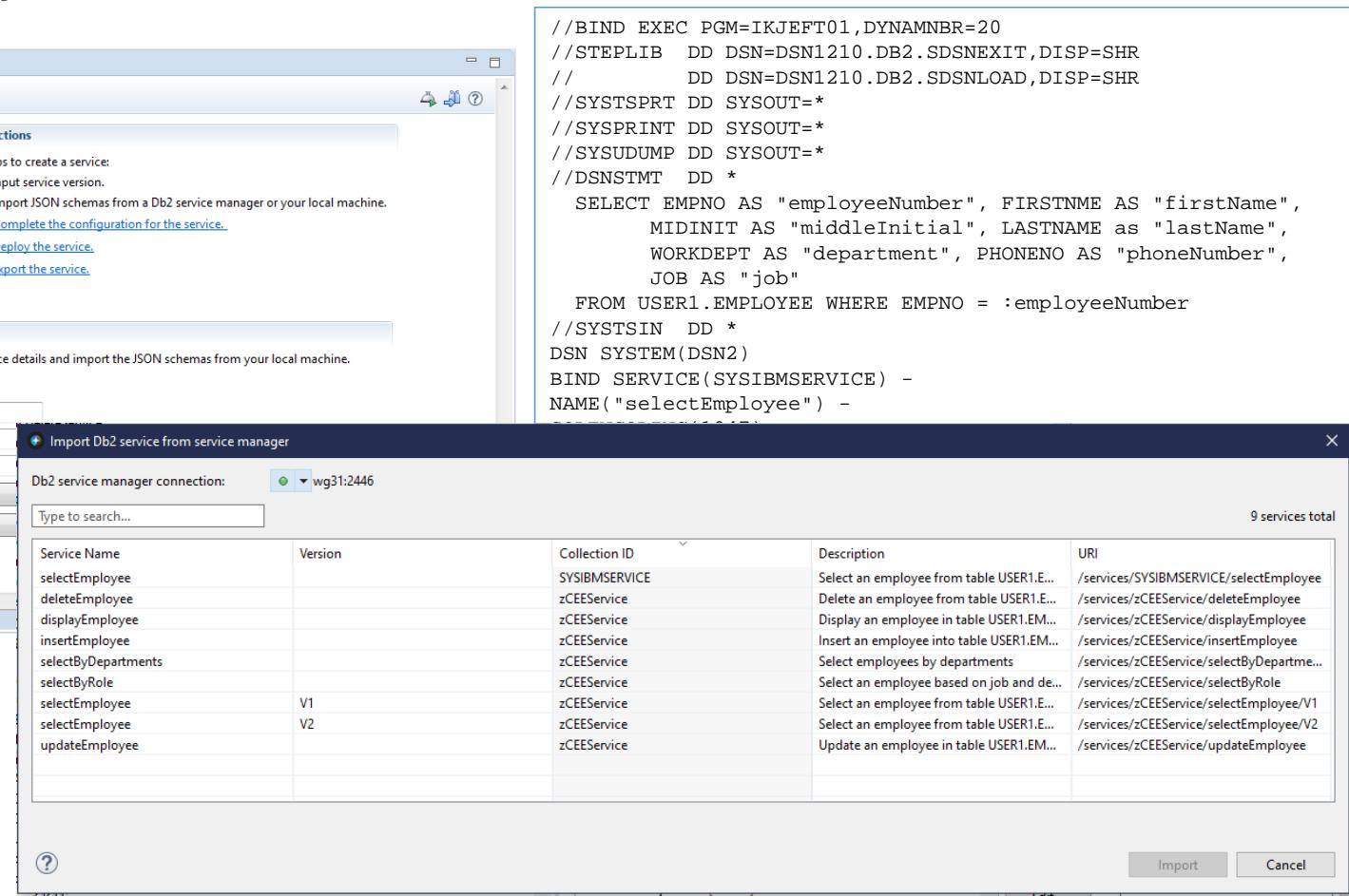
Import a Db2 native REST service from a Db2 service manager. Alternatively, enter your Db2 service details and import the JSON schemas from your local machine.

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

**Definition | Configuration**

The service developer retrieves the Db2 REST services



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

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

Import Cancel

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



z/OS Connect EE

The screenshot shows the IBM z/OS Connect Enterprise Edition interface. On the left is the Project Explorer with a tree view of project components like 'InquireSingle', 'bin', 'serv', and 'src'. The main window displays the 'InquireSingle Service' configuration. Under 'General Information', the service is defined as a 'CICS COMMAREA Service' with version '1.0.0' and a description 'Catalog example Inquire Single Service'. The 'Program' field is set to 'DFH0XLMN'. Under 'Actions', steps for creating a service are listed: 1. Input service version, 2. Specify program or transaction code for the service, 3. Create or import a service interface for the request and response in your service, 4. Complete the configuration for the service, and 5. Export the service. A 'Program' section shows 'DFH0XLMN' selected. Below it, under 'Define Request and Response Service Interfaces', there are fields for 'Request service interface' (set to 'DFH0XCP4.si') and 'Response service interface' (set to 'DFH0XCPA.si'). A context menu is open over the 'Request service interface' field, with options 'Create Service Interface...', 'Import Service Interface...', 'Edit', and 'Delete'. The 'z/OS Connect EE' menu is also open, showing options like 'Compare With', 'Configure', and 'Source'. On the right, a 'Deploy Service' dialog box is open, prompting to 'Deploy Service to z/OS Connect EE Server'. It lists the service 'inquireSingle' with version '13.00' and type 'CICS COMMAREA Se...'. The dialog has 'OK' and 'Cancel' buttons.

Finally, you can deploy the service project as a **Service Archive file (.sar)**

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

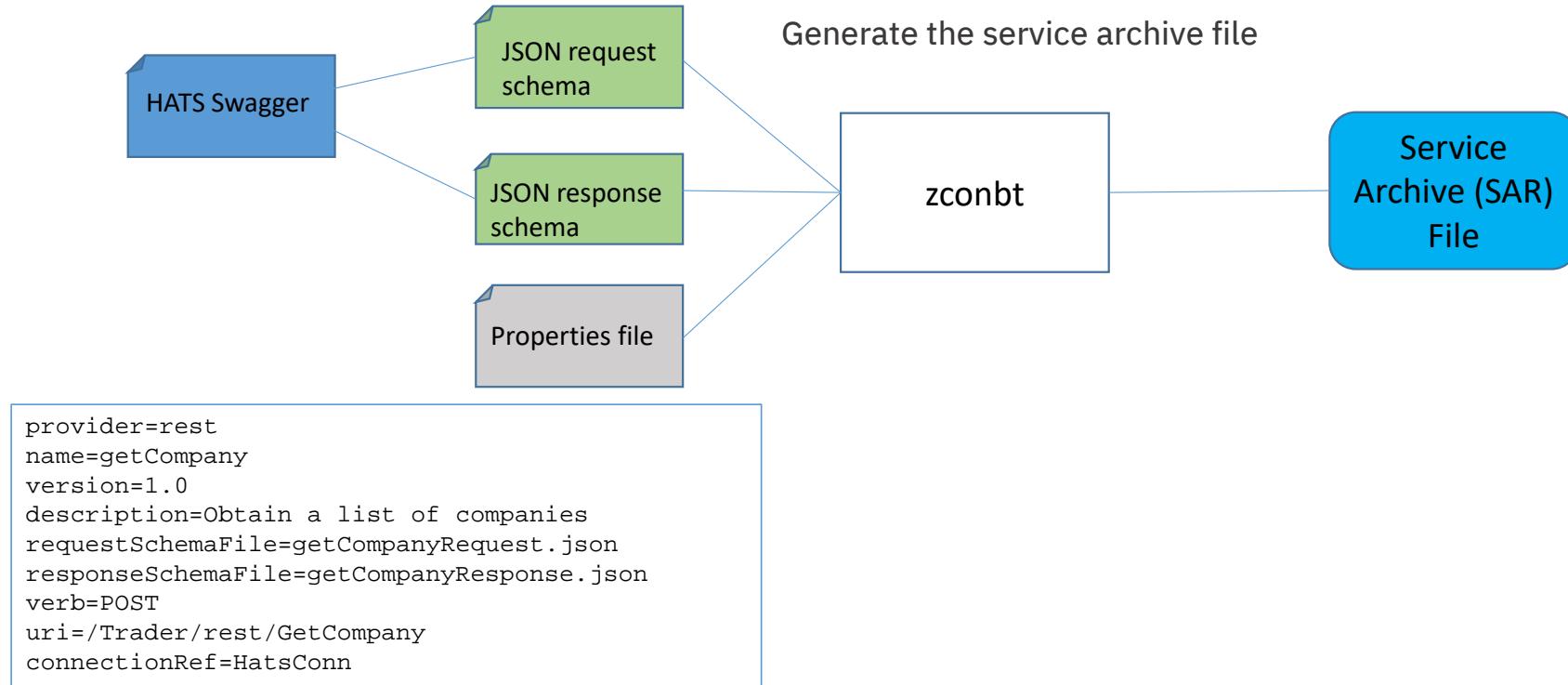


The screenshot shows the IBM z/OS Connect Enterprise Edition interface. On the left, the Project Explorer displays a service named 'InquireSingle'. The main window shows the 'Overview' tab of the 'InquireSingle Service' configuration. Under 'General Information', the 'Type' is set to 'CICS COMMAREA Service' and 'Version' to '1.0.0'. The 'Program' field contains 'DFH0XLMN'. In the 'Actions' section, steps for creating a service are listed: 1. Input service version, 2. Specify program or transaction code for the service, 3. Create or import a service interface for the request and response in your service, 4. Complete the configuration for the service, and 5. Export the service. Below this, under 'Define Request and Response Service Interfaces', there are fields for 'Request service interface' (set to 'DFH0XCP4.si') and 'Response service interface' (set to 'DFH0XCPA.si'). A context menu is open over the 'Export...' option in the 'Actions' section, showing options like 'Deploy Service to z/OS Connect EE Server' and 'Export z/OS Connect EE Service Archive'. A separate 'Export Service Package' dialog box is overlaid on the interface, prompting the user to select where to export the service package. The dialog has fields for 'Export service package to:' (radio buttons for 'Workspace' and 'Local file system', with 'Workspace' selected), 'Folder:' (set to '/services'), 'File name:' (set to 'inquireSingle'), and 'Overwrite service package file' (checkbox). Buttons for '?', 'OK', and 'Cancel' are at the bottom.

Finally, you can export the service project as a **Service Archive file (.sar)**.

# Creating Services without the Toolkit – REST

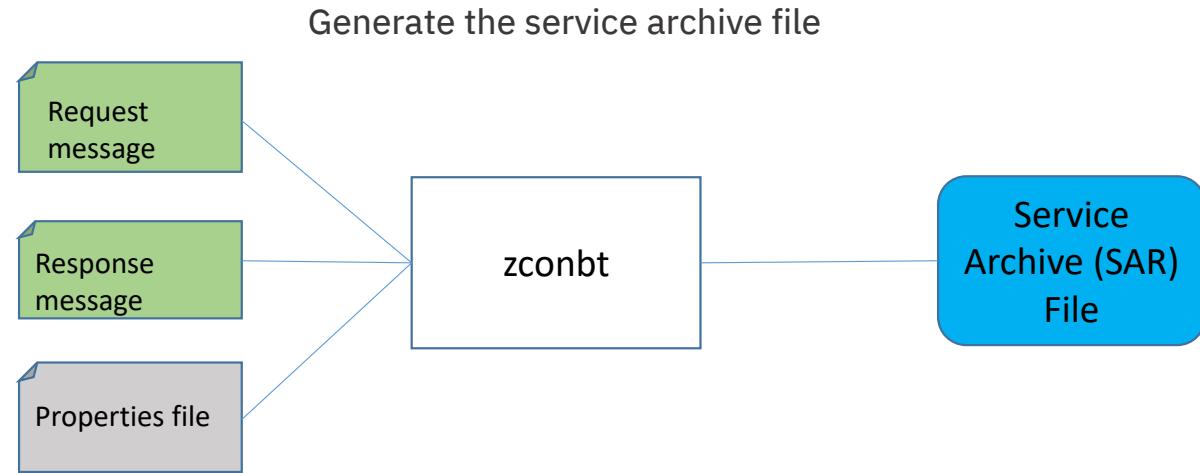
For HATS REST Services use the z/OS Connect Build toolkit (zconbt)



# Creating Services without the Toolkit – MVS Batch

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

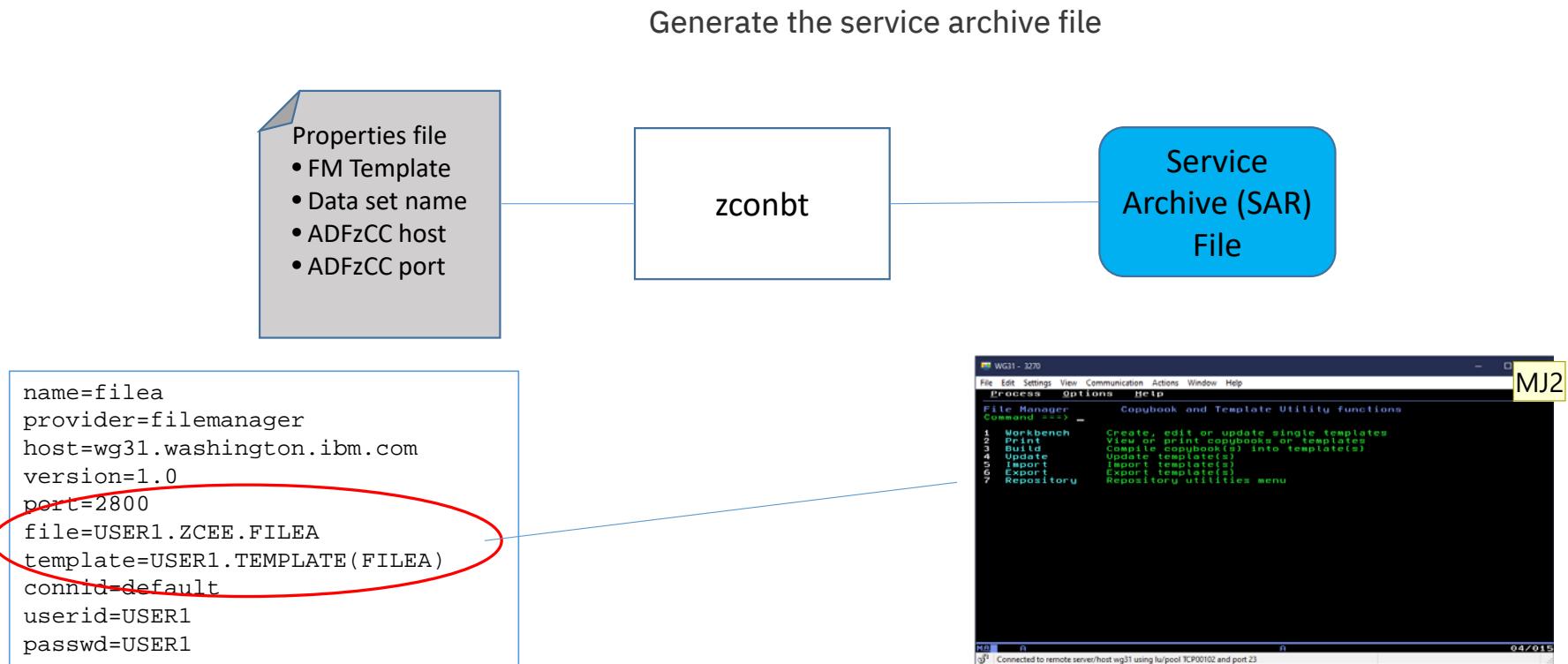
```
name=Filea
version=1.0
provider=wola
description=COBOL batch program
language=COBOL
program=ATSFIL
register=FILEAZCON
connectionRef=wolaCF
requestStructure=fileareq.cpy
responseStructure=filearsp.cpy
```



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

# Creating Services without the Toolkit – FM

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



## Slide 46

---

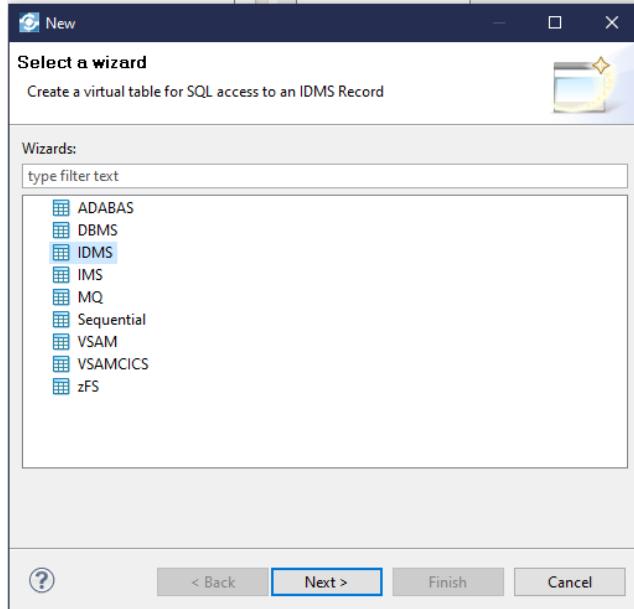
**MJ2** Mitch Johnson, 9/2/2020

# Creating Services without the Toolkit



z/OS Connect EE

For DVM use the DVM Studio



DV Data - Data Virtualization Manager/SQL Samples/Generated.sql - IBM Data Virtualization Manager for z/OS

File Edit Navigate Search Project SQL Run Window Help

Host: wg31.washington.ibm.com (Port 12)  
Server: AVZS Version: 01.01.00.01  
Set Server ...

Data Sources

Navigator

SQL

Discovery

NoSQL

Services

Web Services

- Create Directory
- /REST/
  - Create Service
- INSERT
- UPD
- Set Tree Filter
- Target System
- WSC
- Admin

Generated.sql

```
-- FILE LOCATION: wg31.washington.ibm.com/1200/SQL/Data/AVZS/V100
-- REMARKS: VSAMCICS - USER1.OFFICE.SUPPLIES
SELECT WS_ITEM_REF, WS_DESCRIPTION, WS_DEPARTMENT, WS_COST, WS_IN_
WS_ON_ORDER
FROM EXMPCAT LIMIT 1000;

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

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

Server Trace Console

WS_DESCRIPTION	WS_DEPARTMENT	WS_C
Mitch Johnson	10	002.
Mitch Johnson	10	002.
Mitch Johnson	10	002.

z/OS Connect REST Interface

DSN Name: AVZS

Execute Query

Refresh

Generate SAR File(s)

Columns Group: 1 or 1

3 rows SQL Messages

Columns per group: 25



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

Quick and easy **API mapping**.

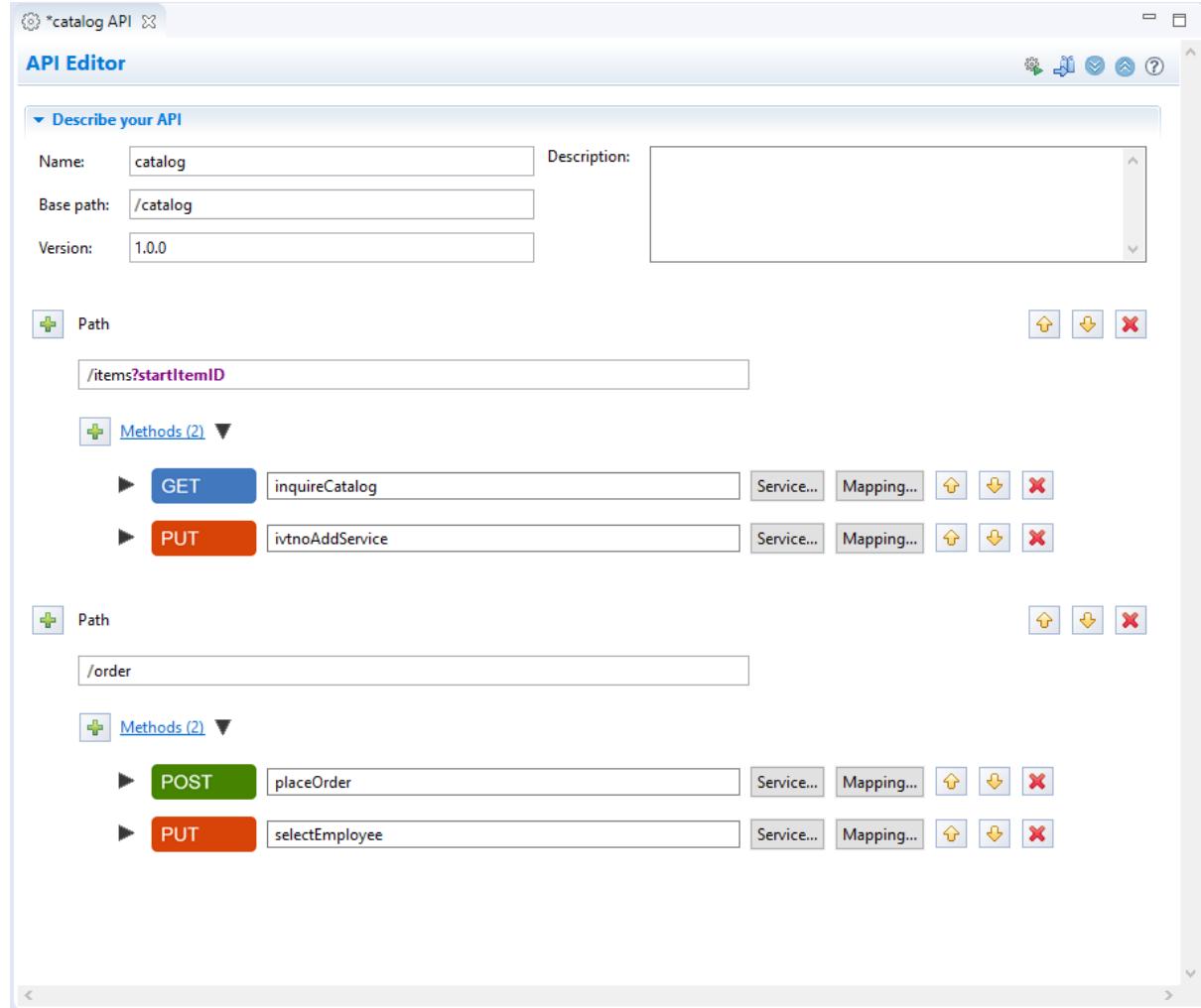
*Remember: All service archives files are functionally equivalent regardless of how they are created*



## /api\_toolkit/api\_editor

Quick and easy **API mapping**.

# API toolkit – API Editor



The screenshot shows the API toolkit API Editor interface. It displays two API definitions:

- catalog API**:
  - Path**: /items?startItemID
  - Methods (2)**:
    - ▶ **GET** inquireCatalog
    - ▶ **PUT** ivtnoAddService
- order**:
  - Path**: /order
  - Methods (2)**:
    - ▶ **POST** placeOrder
    - ▶ **PUT** selectEmployee

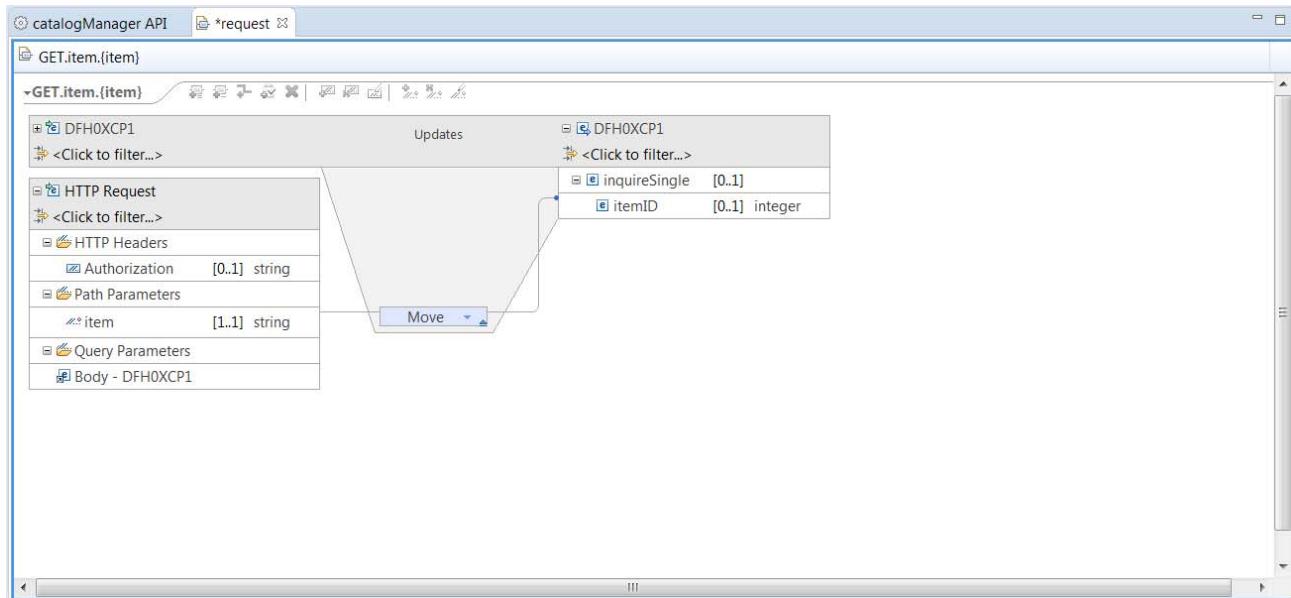
The **API toolkit** is designed to encourage RESTful API design.

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

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

# API toolkit – API Editor

## API mapping: Point-and-click interface



Map both the request and response for each API.

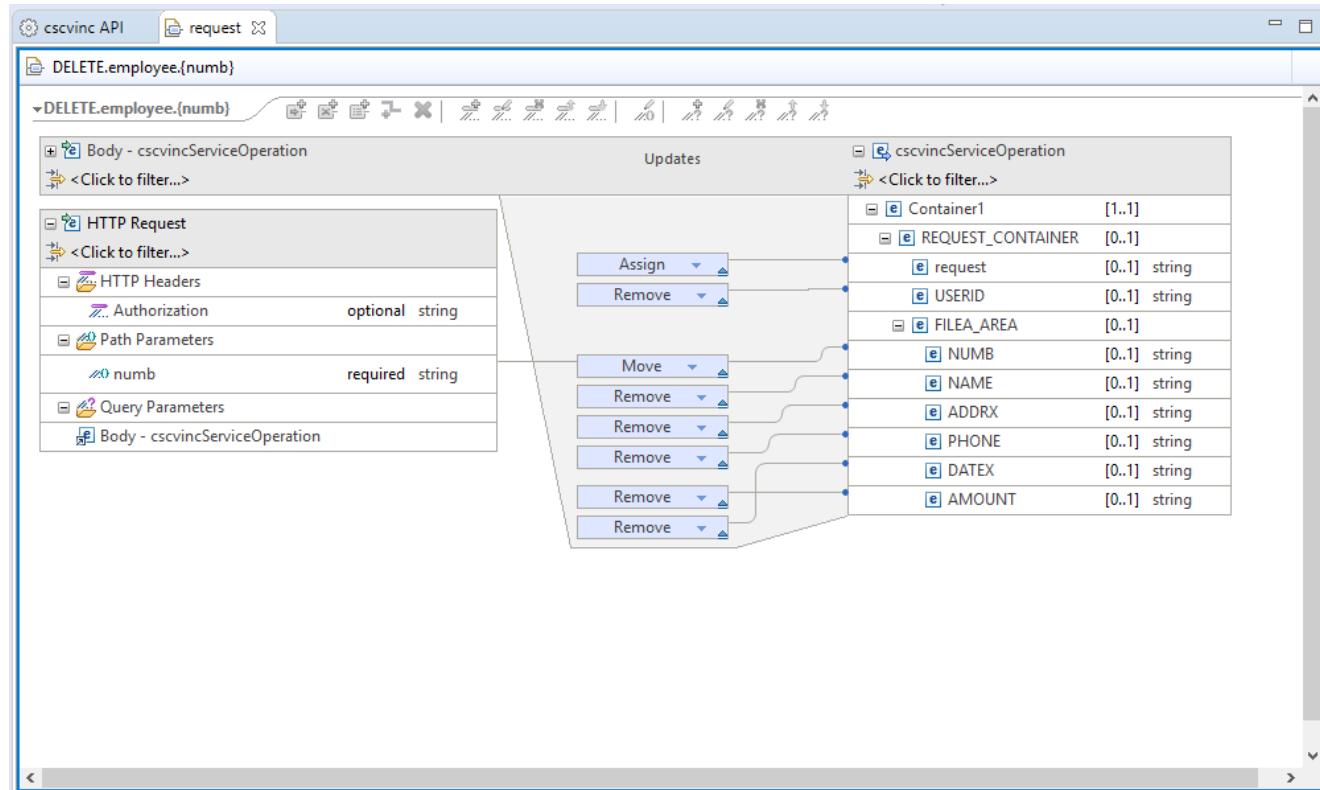
Map path and query parameters to native data structures.

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

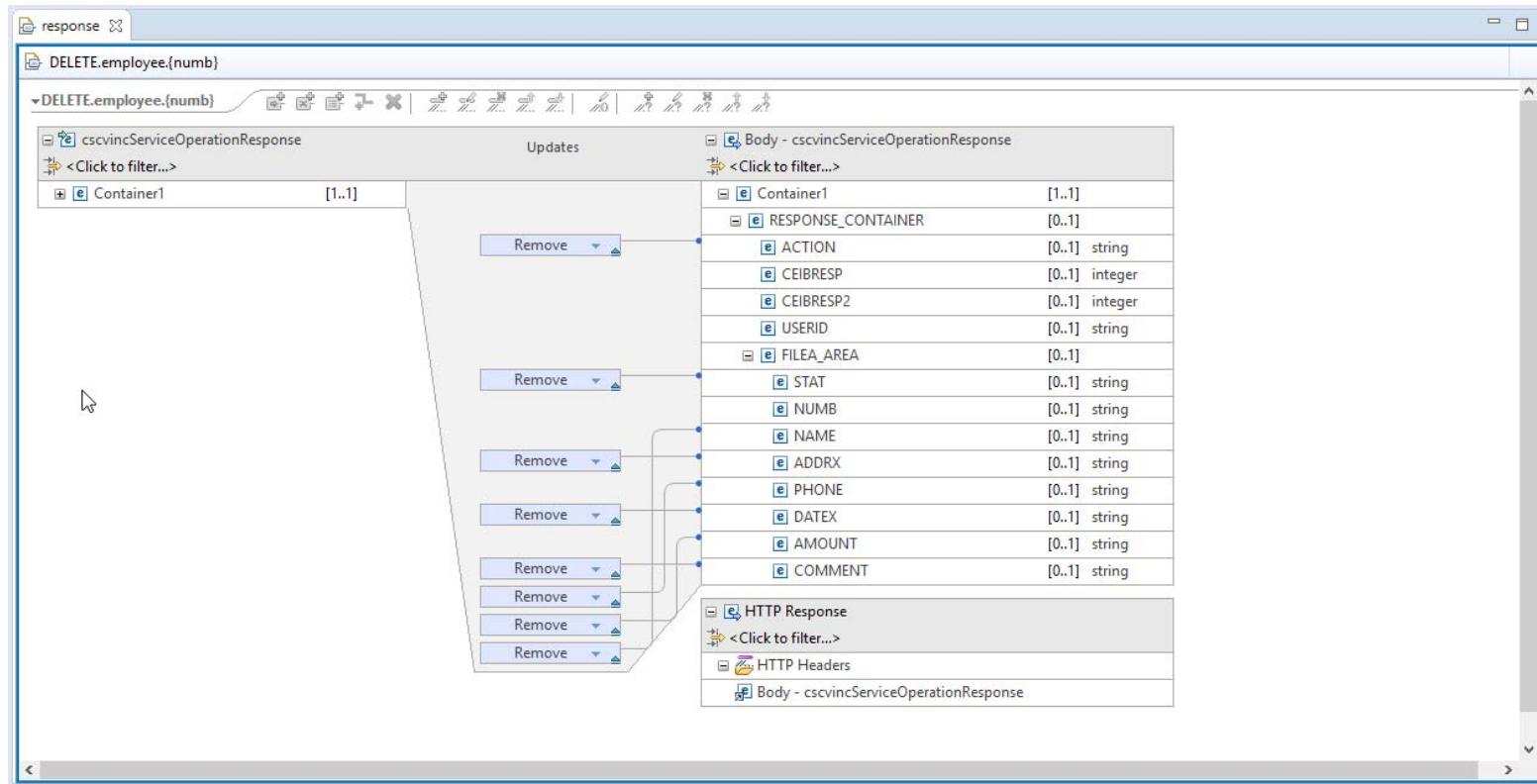
API mapping: Point-and-click interface



# API toolkit – API Editor

## API mapping: Point-and-click interface

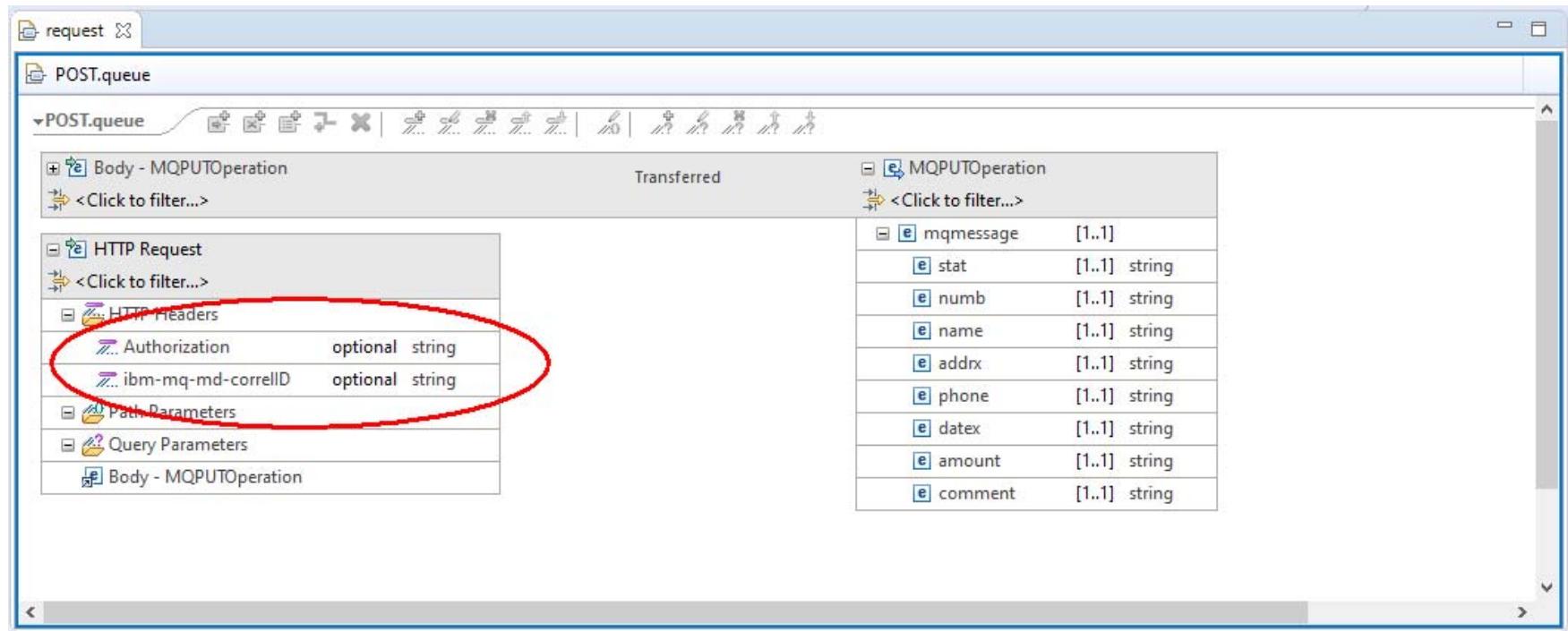
Allows the API Developer to remove fields from the response to simplify the API



# API toolkit – API Editor

## API mapping: Adding HTTP header properties

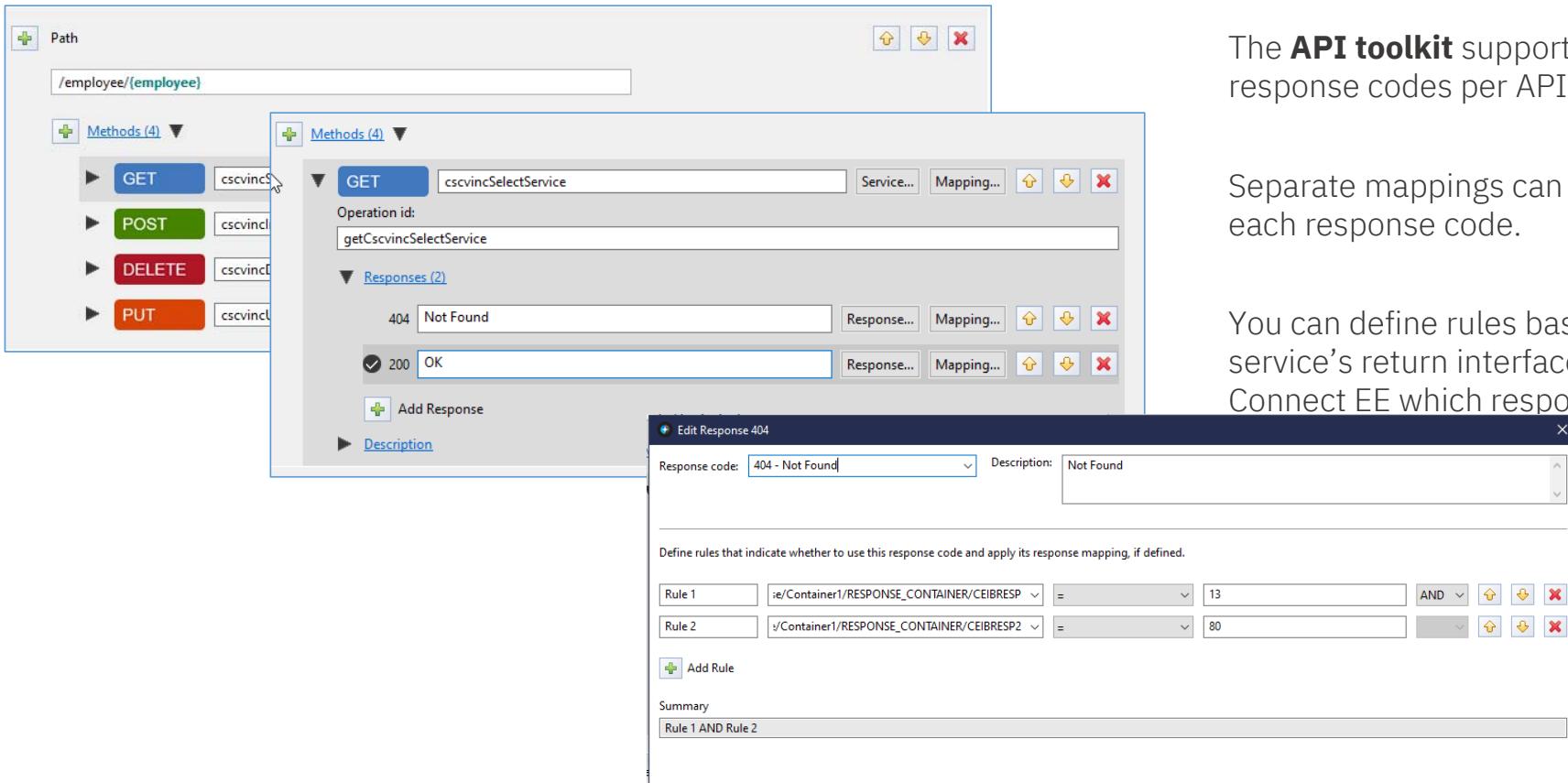
Allows the API Developer to remove fields to the request.



The screenshot shows the API Editor interface with the 'request' tab selected. The left pane displays the request structure for a 'POST.queue' operation. It includes sections for 'Body - MQPUTOperation', 'Transferred', and 'MQPUTOperation'. Under 'MQPUTOperation', there is a list of fields: mqmessage [1..1], stat [1..1] string, numb [1..1] string, name [1..1] string, addrx [1..1] string, phone [1..1] string, datex [1..1] string, amount [1..1] string, and comment [1..1] string. The 'HTTP Request' section contains an 'HTTP Headers' section, which is highlighted with a red oval. This section lists two optional string fields: 'Authorization' and 'ibm-mq-md-correlID'. Other sections in the request pane include 'Path Parameters' and 'Query Parameters'. The right pane shows the transferred message structure, which is identical to the MQPUTOperation structure on the left.

# API toolkit

## API definition with multiple response codes



The screenshot shows the API toolkit interface for defining an API operation. The path is set to `/employee/{employee}`. The Methods section lists four methods: GET, POST, DELETE, and PUT, each associated with a service component like `cscvincS`, `cscvincI`, and `cscvincD`.

For the GET method, the operation ID is `getCscvincSelectService`. The Responses section defines two response codes: 404 Not Found and 200 OK. A modal window titled "Edit Response 404" is open, showing the response code as "404 - Not Found" and the description as "Not Found". Below this, rules are defined to apply this response code based on specific conditions in the service's return interface.

**Responses (2)**

- 404 Not Found
- 200 OK

**Edit Response 404**

Response code: 404 - Not Found | Description: Not Found

Define rules that indicate whether to use this response code and apply its response mapping, if defined.

Rule 1	:e/Container1/RESPONSE_CONTAINER/CEIBRESP	=	13	AND
Rule 2	:y/Container1/RESPONSE_CONTAINER/CEIBRESP2	=	80	OR

Add Rule

Summary: Rule 1 AND Rule 2

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

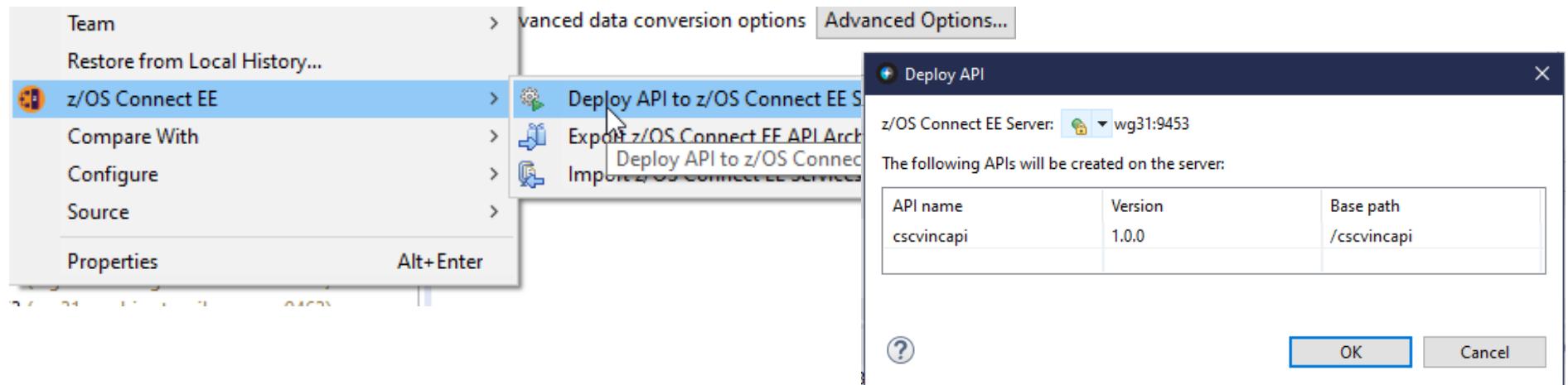
Separate mappings can be defined for each response code.

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:

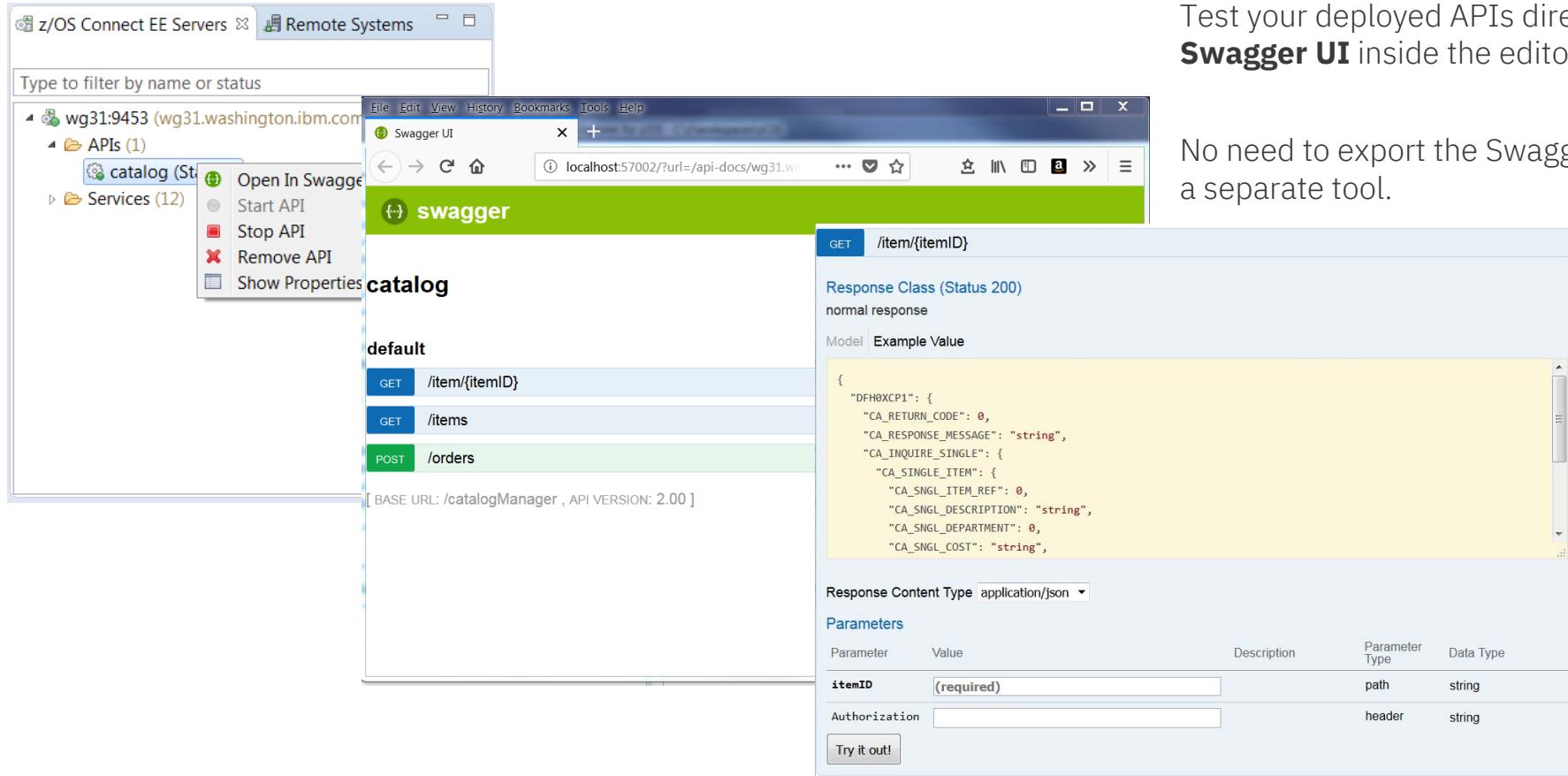


**Right-click deploy to server** enables developers to quickly deploy, test, and iterate on their APIs.

**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



The screenshot shows the z/OS Connect EE API Editor interface. On the left, there's a sidebar with 'z/OS Connect EE Servers' and 'Remote Systems' tabs, and a search bar. Below that, it lists 'APIs (1)' containing a 'catalog' entry, and 'Services (12)'. A context menu is open over the 'catalog' entry with options: 'Open In Swagger', 'Start API', 'Stop API', 'Remove API', and 'Show Properties'. The main area is titled 'swagger' and shows a 'catalog' section. It lists 'default' with three operations: 'GET /item/{itemID}', 'GET /items', and 'POST /orders'. The 'POST /orders' operation is highlighted with a green background. Below the operations, it says '[ BASE URL: /catalogManager , API VERSION: 2.00 ]'. To the right, a detailed view of the 'POST /orders' operation is shown. It has a 'Model' tab selected, displaying a JSON schema:

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

Below the model, the 'Response Content Type' is set to 'application/json'. Under 'Parameters', there are two entries: 'itemID' (required, path, string) and 'Authorization' (header, string). A 'Try it out!' button is at the bottom.

Test your deployed APIs directly with **Swagger UI** inside the editor.

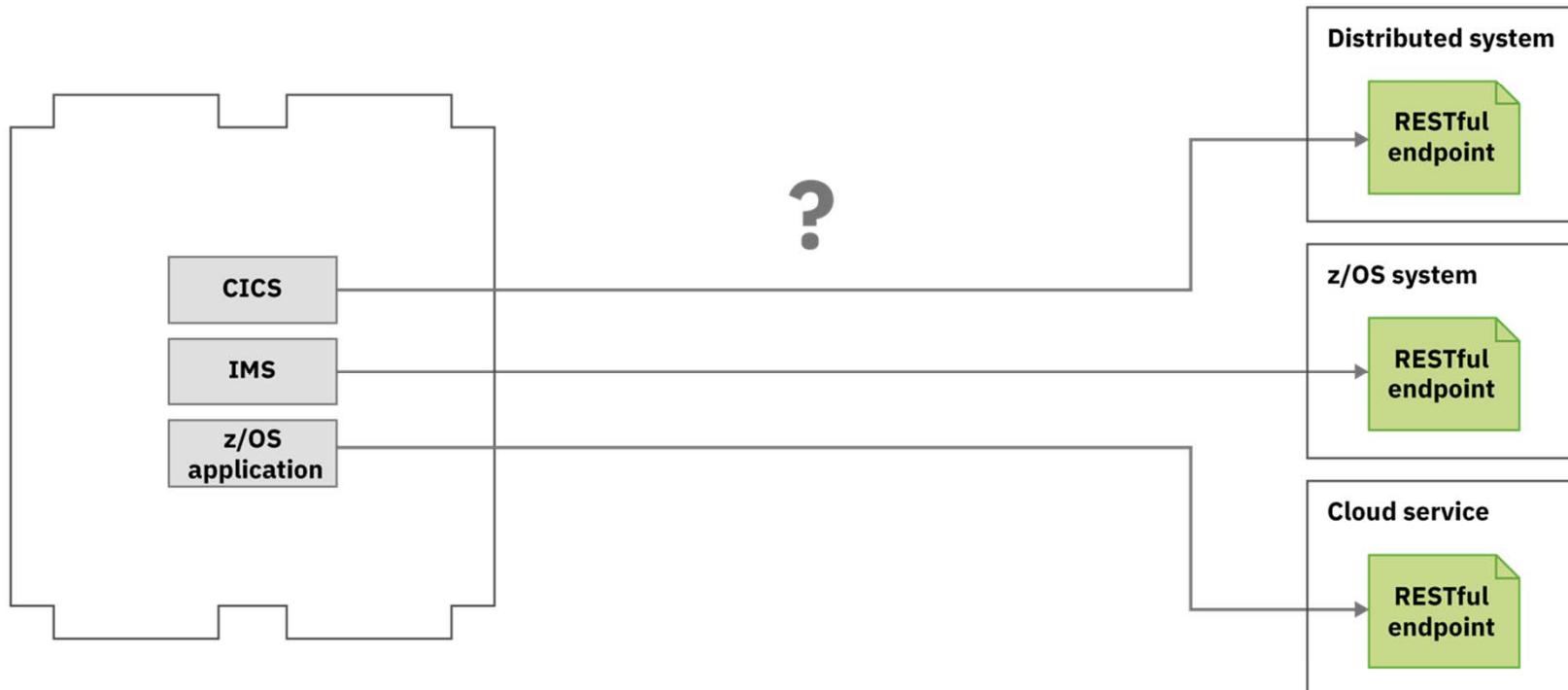
No need to export the Swagger doc to a separate tool.



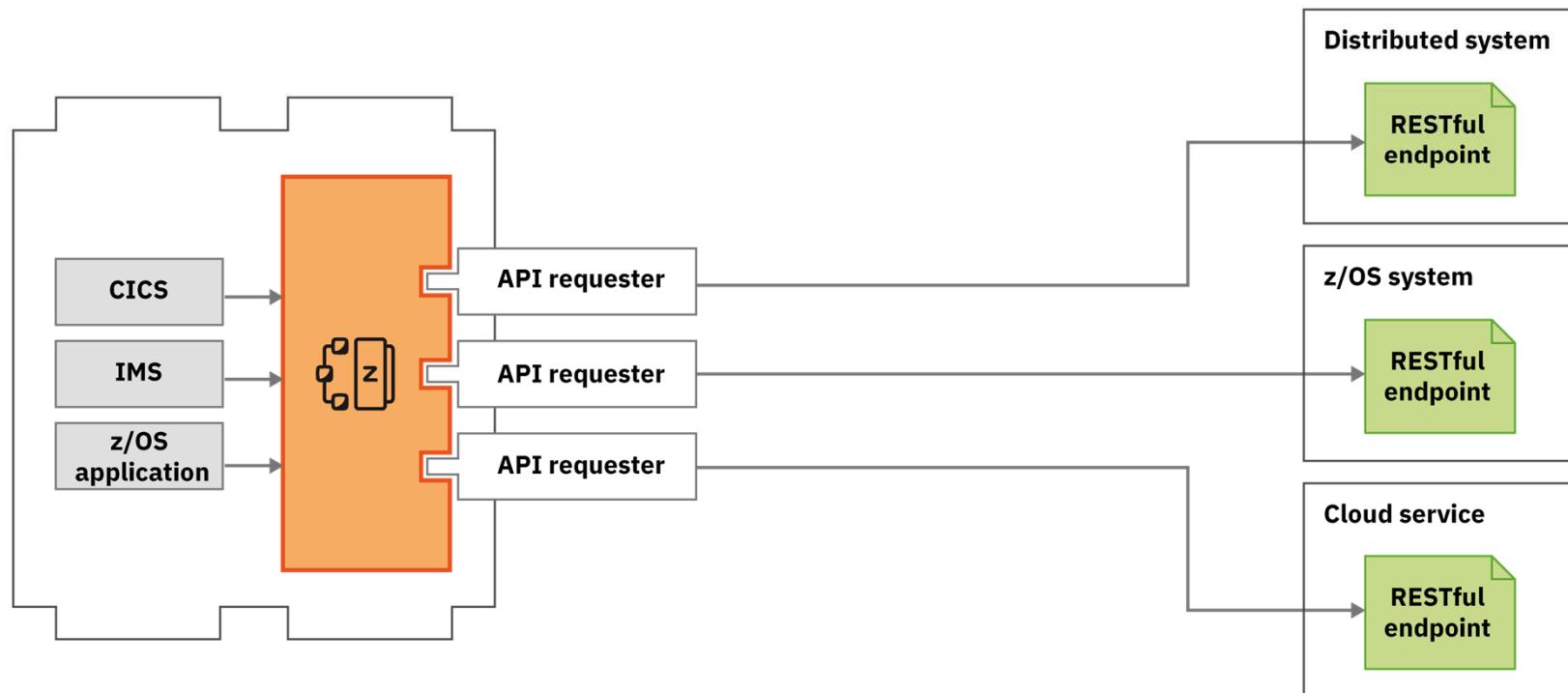
## **/api\_toolkit/apiRequesters**

Quick and easy **API mapping**.

# What about calling external APIs from my z/OS assets?



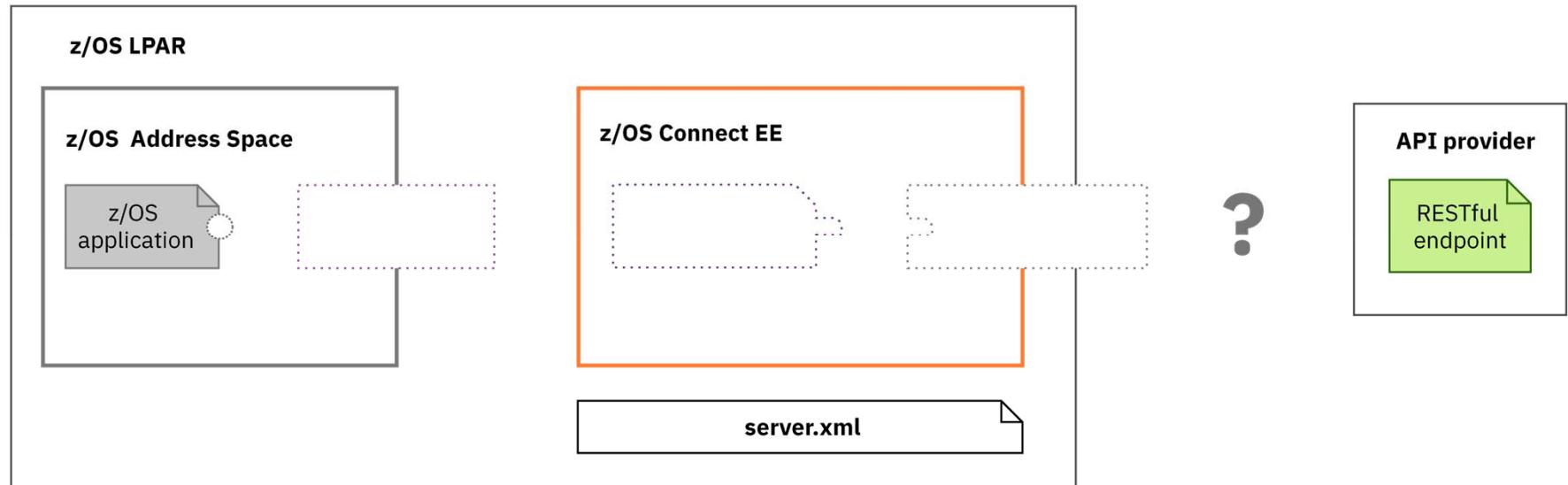
# Use API requester to call external APIs from z/OS assets





# Steps to calling an external API

Starting point

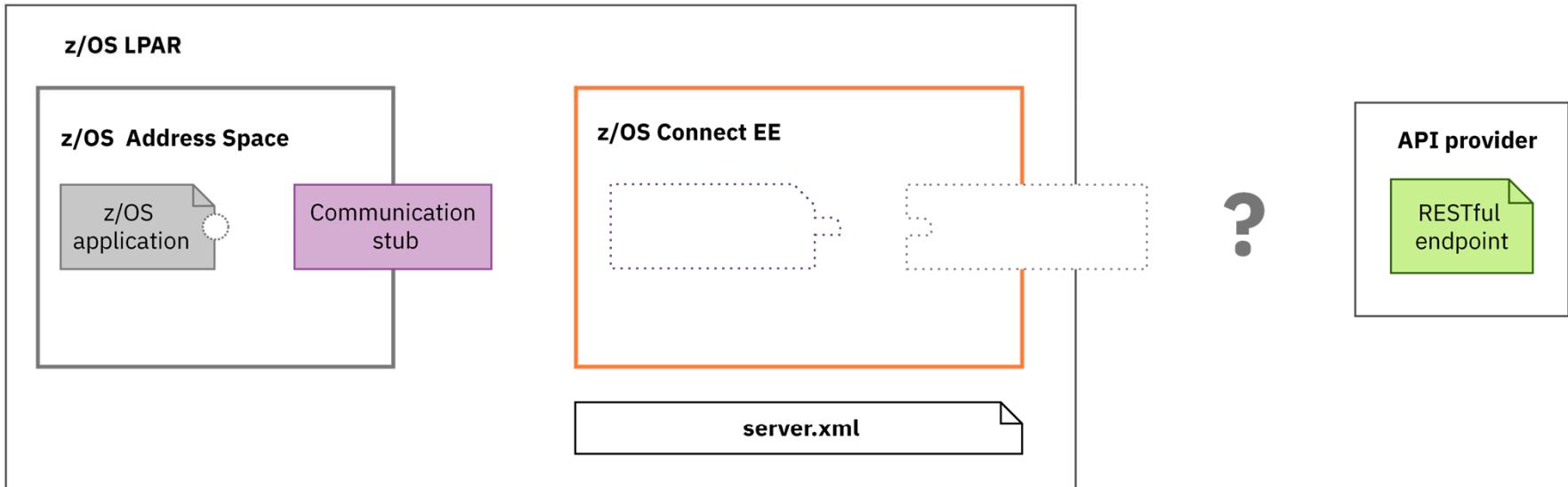




z/OS Connect EE

# Steps to calling an external API

## Step 1. Configure communication stub



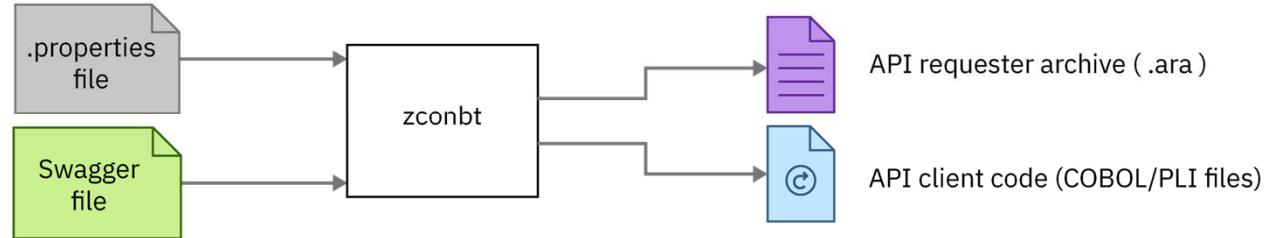
Configure a communication stub.

- Once per CICS region system using a URIMAP resource
- For non CICS client the configuration is done via environment variables

 [ibm.biz/zosconnect-configure-comms-stub](http://ibm.biz/zosconnect-configure-comms-stub)

# Steps to calling an external API

## Step 2. Generate API requester archive from Swagger

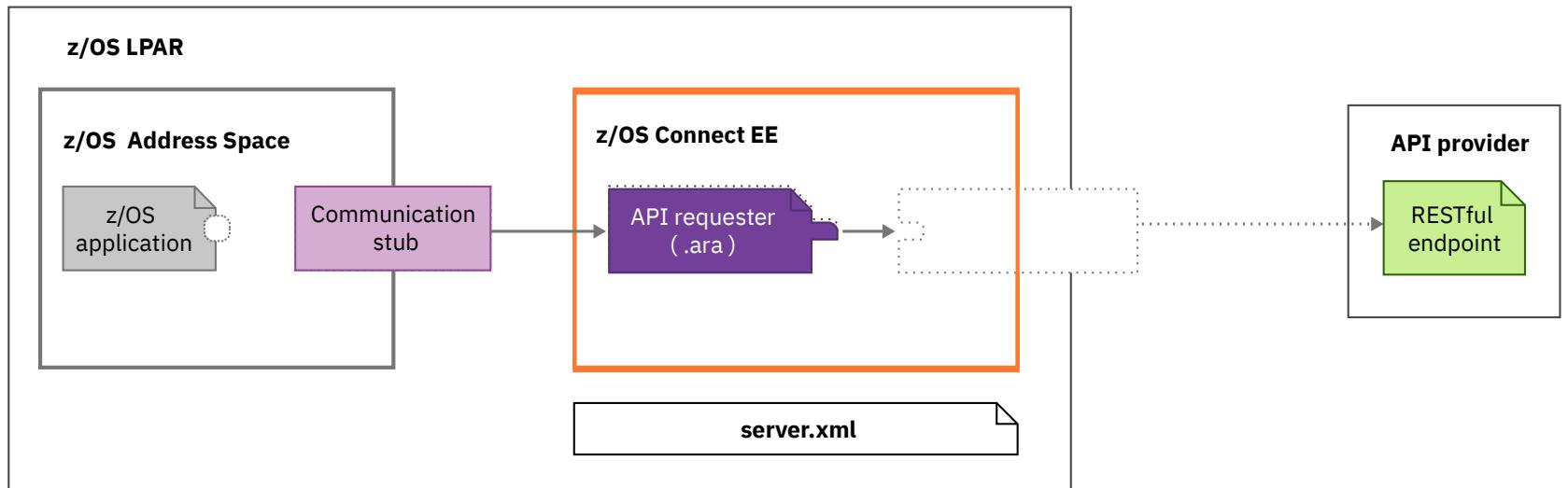


Generate the API requester archive file, and API client code.

 [ibm.biz/zosconnect-generate-ara](http://ibm.biz/zosconnect-generate-ara)

# Steps to calling an external API

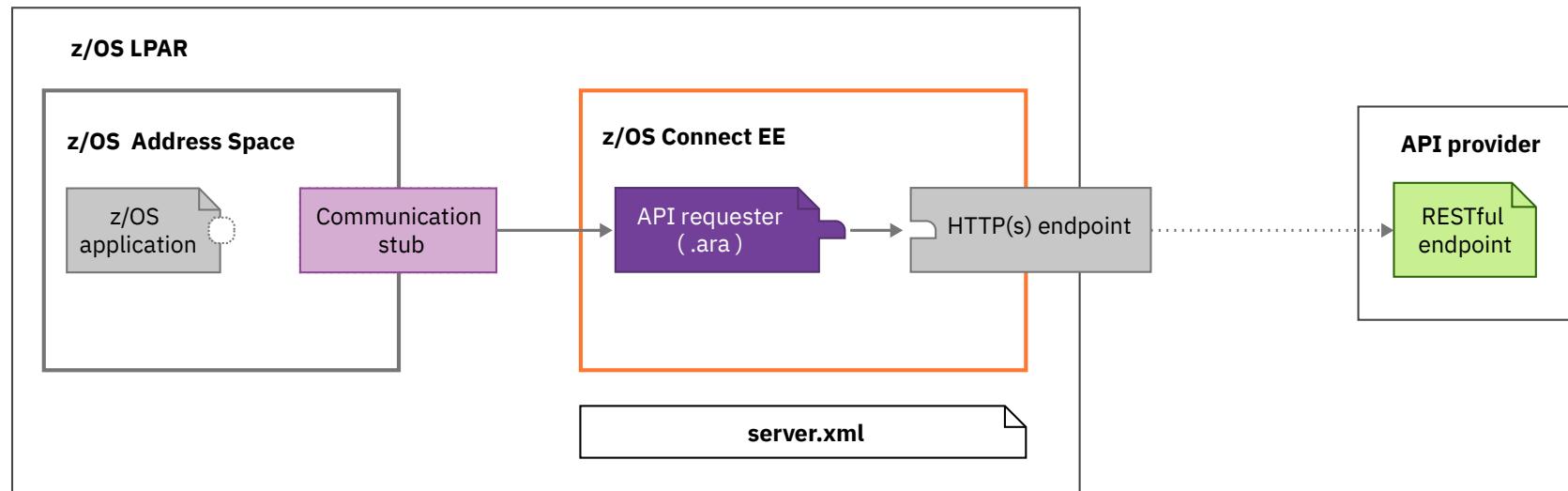
## Step 3. Deploy API requester (.ara) archive



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

# Steps to calling an external API

## Step 4. Configure HTTP(S) endpoint



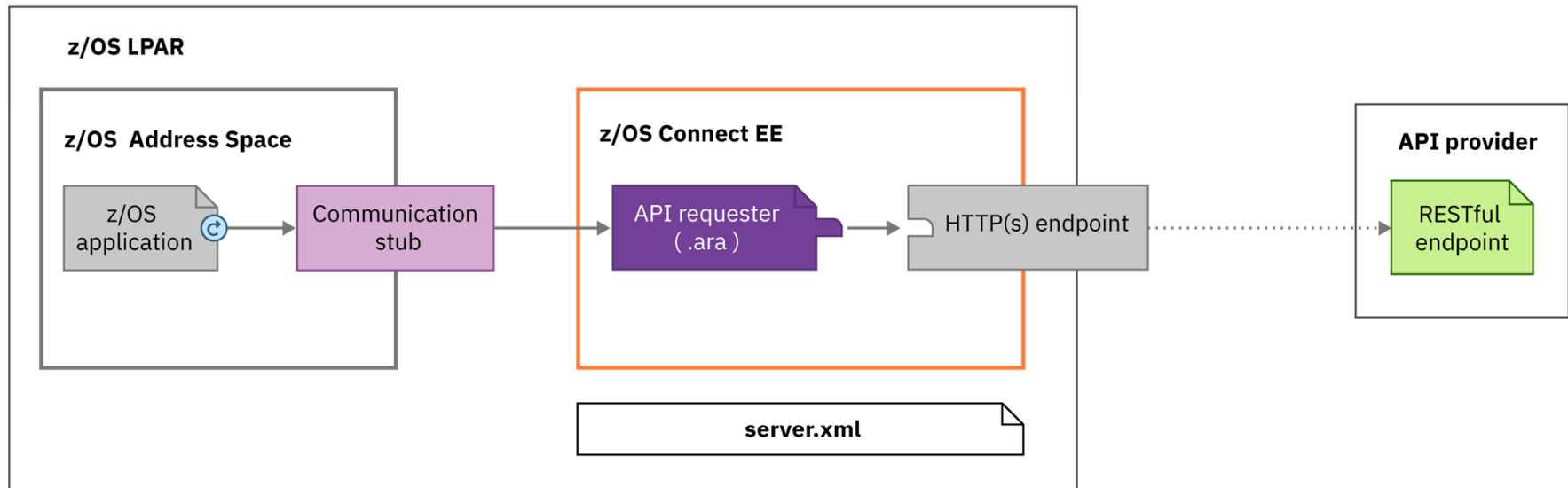
Configure the connection between z/OS Connect EE and the external API.

 [ibm.biz/zosconnect-configure-endpoint-connection](http://ibm.biz/zosconnect-configure-endpoint-connection)

© 2018, 2020 IBM Corporation

# Steps to calling an external API

## Step 5. Update z/OS application



Finally, add the generated API client code to your existing application and use it to make the external API call.

 [ibm.biz/zosconnect-configure-requester-zos-application](http://ibm.biz/zosconnect-configure-requester-zos-application)

# Steps to calling an external API

Step 5a. Update the z/OS application to include new copy books

The screenshot shows three windows in the Rational Application Developer interface:

- GETAPI**: A code editor window showing an AS/400 copybook structure. It includes fields for ERROR-MSG, EM-ORIGIN, EM-CODE, and EM-DETAIL, and a note about a required copybook: \* Copy API Requester required copybook COPY BAQRINFO.
- apis.xml**: An XML configuration file for the ZOS Connect API Requester. It defines a server with a featureManager and a zosconnect\_apiRequesters section. A specific endpoint connection is highlighted with a red oval, showing details like host="http://wg31.washington.ibm.com", port="9120", and basicAuthRef="myBasicAuth".
- CSC02I01**: A code editor window showing an AS/400 copybook structure for an API request. It includes fields for BAQ-APINAME, BAQ-APINAME-LEN, BAQ-APIPATH, BAQ-APIPATH-LEN, BAQ-APIMETHOD, and BAQ-APIMETHOD-LEN. A red arrow points from the 'COPY CSC02I01.' line in the GETAPI window to the 'COPY CSC02I01.' line in this window.

A callout box highlights several configuration parameters in the apis.xml file:

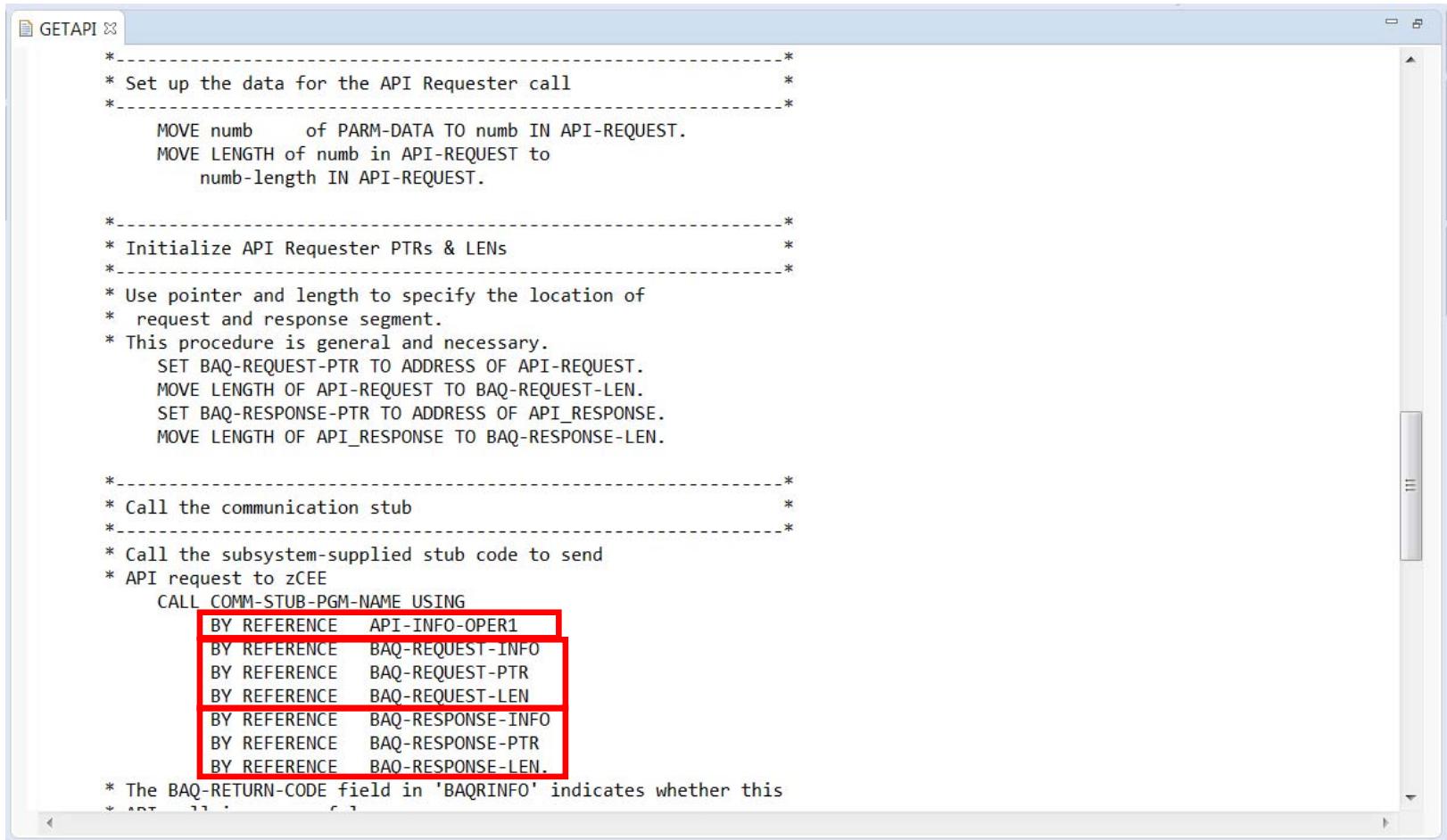
- host="http://wg31.washington.ibm.com"
- port="9120"
- basicAuthRef="myBasicAuth"
- connectionTimeout="10s"
- receiveTimeout="20s" />
- user="Fred"
- password="fredpwd" />

Another callout box highlights parameters in the configuration panel:

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

# Steps to calling an external API

## Step 5b. Update the z/OS application to call the stub



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

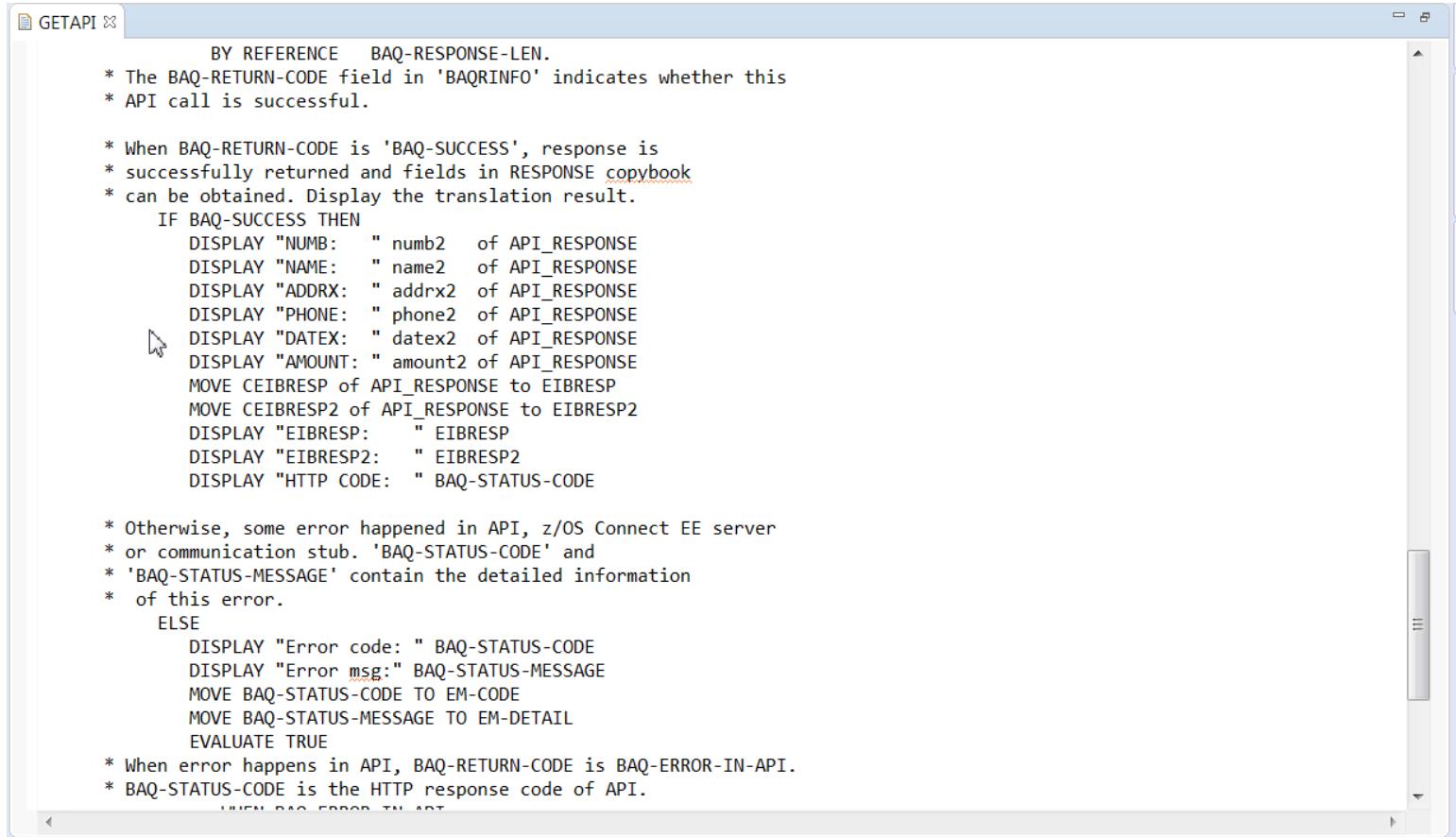
*-----*
* Initialize API Requester PTRs & LENs          *
*-----*
* Use pointer and length to specify the location of
*   request and response segment.
* This procedure is general and necessary.
      SET BAQ-REQUEST-PTR TO ADDRESS OF API-REQUEST.
      MOVE LENGTH OF API-REQUEST TO BAQ-REQUEST-LEN.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

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

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

# Steps to calling an external API

## Step 5c. Update the z/OS application to access the results

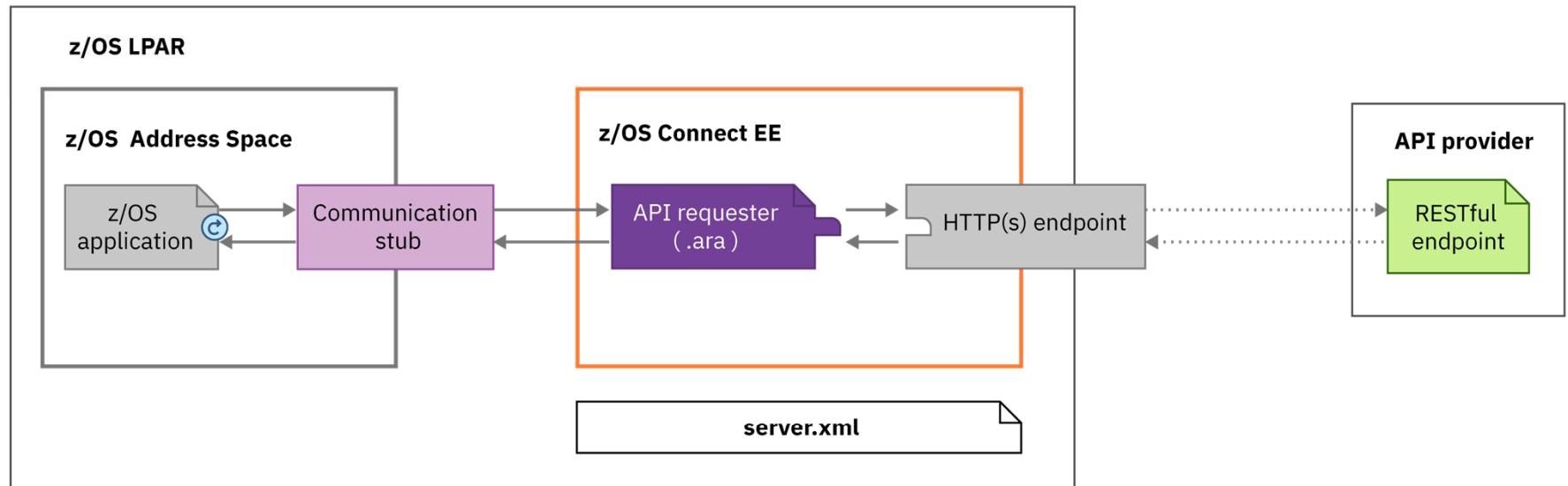


The screenshot shows the GETAPI editor window with the following AS/400 JCL code:

```
BY REFERENCE BAQ-RESPONSE-LEN.  
* The BAQ-RETURN-CODE field in 'BAQRINFO' indicates whether this  
* API call is successful.  
  
* When BAQ-RETURN-CODE is 'BAQ-SUCCESS', response is  
* successfully returned and fields in RESPONSE copybook  
* can be obtained. Display the translation result.  
IF BAQ-SUCCESS THEN  
  DISPLAY "NUMB: " numb2 of API_RESPONSE  
  DISPLAY "NAME: " name2 of API_RESPONSE  
  DISPLAY "ADDRX: " addrx2 of API_RESPONSE  
  DISPLAY "PHONE: " phone2 of API_RESPONSE  
  DISPLAY "DATEX: " datex2 of API_RESPONSE  
  DISPLAY "AMOUNT: " amount2 of API_RESPONSE  
  MOVE CEIBRESP of API_RESPONSE to EIBRESP  
  MOVE CEIBRESP2 of API_RESPONSE to EIBRESP2  
  DISPLAY "EIBRESP: " EIBRESP  
  DISPLAY "EIBRESP2: " EIBRESP2  
  DISPLAY "HTTP CODE: " BAQ-STATUS-CODE  
  
* Otherwise, some error happened in API, z/OS Connect EE server  
* or communication stub. 'BAQ-STATUS-CODE' and  
* 'BAQ-STATUS-MESSAGE' contain the detailed information  
* of this error.  
ELSE  
  DISPLAY "Error code: " BAQ-STATUS-CODE  
  DISPLAY "Error msg: " BAQ-STATUS-MESSAGE  
  MOVE BAQ-STATUS-CODE TO EM-CODE  
  MOVE BAQ-STATUS-MESSAGE TO EM-DETAIL  
  EVALUATE TRUE  
  
* When error happens in API, BAQ-RETURN-CODE is BAQ-ERROR-IN-API.  
* BAQ-STATUS-CODE is the HTTP response code of API.  
  WHEN BAQ-ERROR-IN-API
```

# Steps to calling an external API

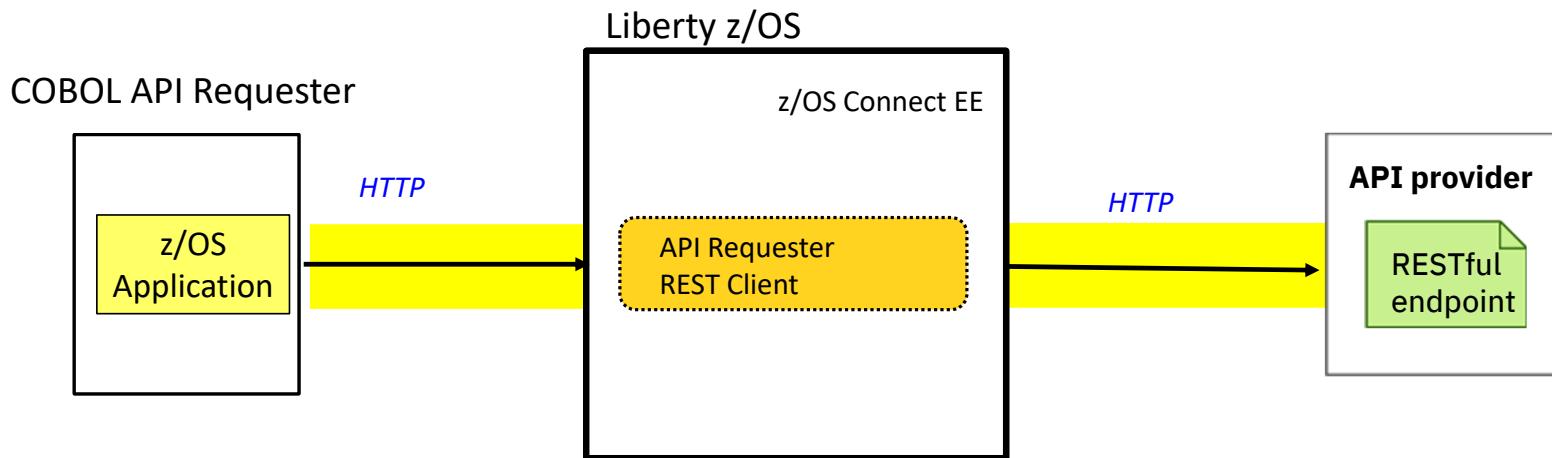
Done





z/OS Connect EE

## 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:

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

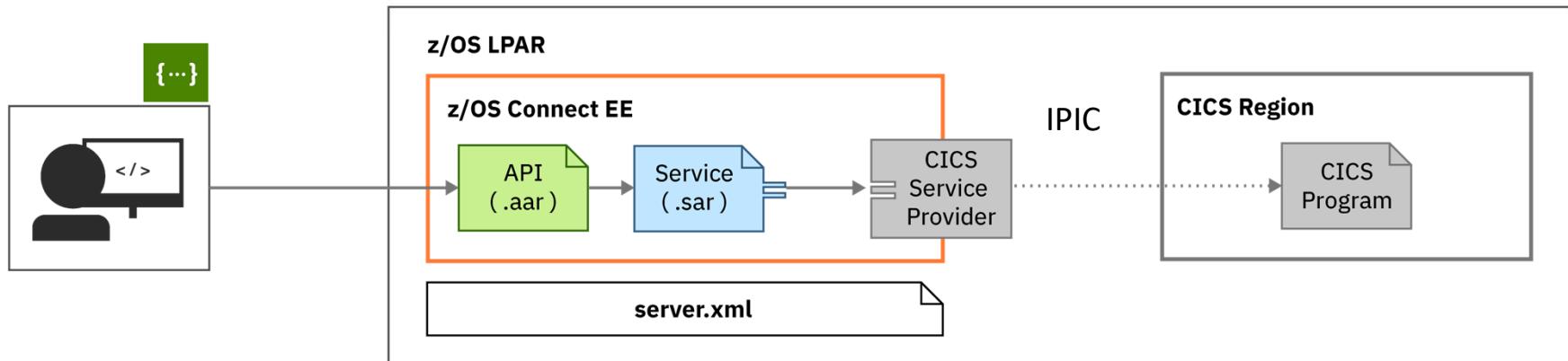


## /common\_scenarios

Typical connection patterns to different subsystems.

# Connections to CICS

## Topology



Connection to CICS is configured in `server.xml`.

An IPIC connection must be configured in CICS.

 [ibm.biz/zosconnect-scenarios](http://ibm.biz/zosconnect-scenarios)

## CICS IPIC (server.xml)



**z/OS Connect EE**

The server.xml file is the key configuration file:

The screenshot shows the 'inquireSingle Service' configuration interface. The top navigation bar includes tabs for 'Overview' and 'Configuration'. The main area is divided into 'Required Configuration' and 'Optional Configuration' sections.

**Required Configuration:**

- Coded character set identifier (CCSID): 37
- Connection reference: catalog

**Optional Configuration:**

- Transaction ID: (empty input field)
- Transaction ID usage: (dropdown menu)

Below the configuration tabs, there is a terminal window titled 'WG31' showing CICS release information and configuration details for a TCPIP service.

```
OVERTYPE TO MODIFY
CEDA ALTER TCpipservice( IPIC      )          CICS RELEASE = 0710
TCpipservice : IPIC
GR0up       : SYSPGRP
DEScripti0n ==> DFHISAIP
Urm         ==> 01491
P0rtnumber ==> 01491
SStatus     ==> Open
PR0tocol    ==> IPic
TRansaction ==> CISS
Backlog    ==> 00000
TSqprefix   :
Host        ==> ANY
(Mixed Case) ==>
Ipaddress   ==> ANY
SPeciftpcs ==>
SOcketclose ==> No
MAXPersist  ==> No
+ MAXDatalen ==> 000032
No | 0-240000 (HHMMSS)
No | 0-65535
3-524288

SYSID=CICS APPLID=CICS53Z
```

At the bottom, a footer bar displays PF keys (PF 1 to PF 12, HELP, COM, END), system status (6 CRSR, 7 SBH, 8 SFH, 9 MSG, 10 SB, 11 SF, 12 CNCL), and the date/time (06/022).

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

catalog.xml

## Design      Source

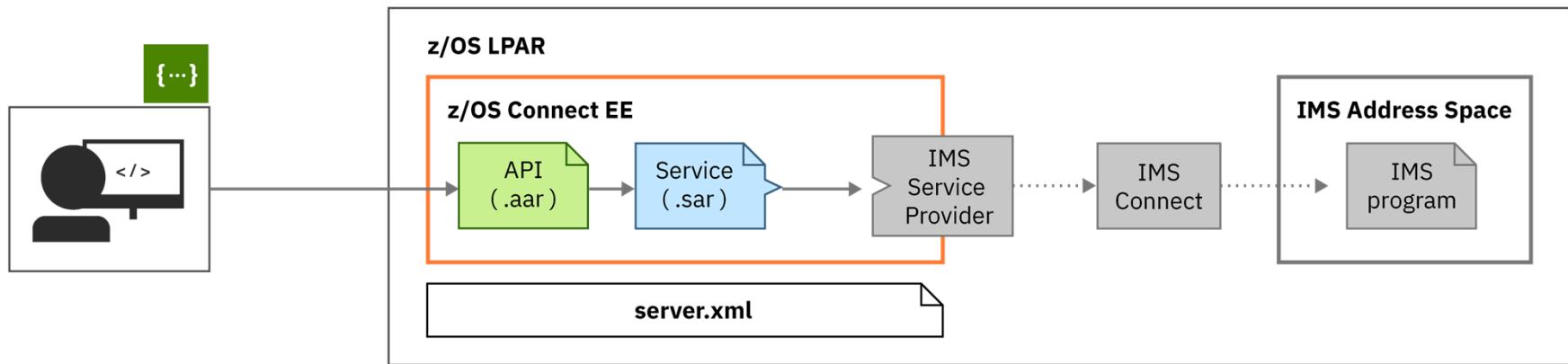
```
1<server description="CICS IPIC - catalog">
2
3<!-- Enable features --&gt;
4&lt;featureManager&gt;
5  &lt;feature&gt;zosconnect:cicsService-1.0&lt;/feature&gt;
6&lt;/featureManager&gt;
7
8&lt;zosconnect_cicsIpicConnection id="catalog"&gt;
9  host="wg31.washington.ibm.com"
10  port="1491"
11  transid="CSMI"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

## Define IPIC connection to CICS

# Connections to IMS TM



## Topology



Configure the connection to IMS through `ims-connections.xml` and `ims-interactions.xml` in the IMS service registry.

[ibm.biz/zosconnect-scenarios](http://ibm.biz/zosconnect-scenarios)

© 2018, 2020 IBM Corporation

# IMS Connections and Interactions



z/OS Connect EE

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 , TMEMBER=OTMAMEM )
IMSPLEX=( MEMBER=IMS14HWS , TMEMBER=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>

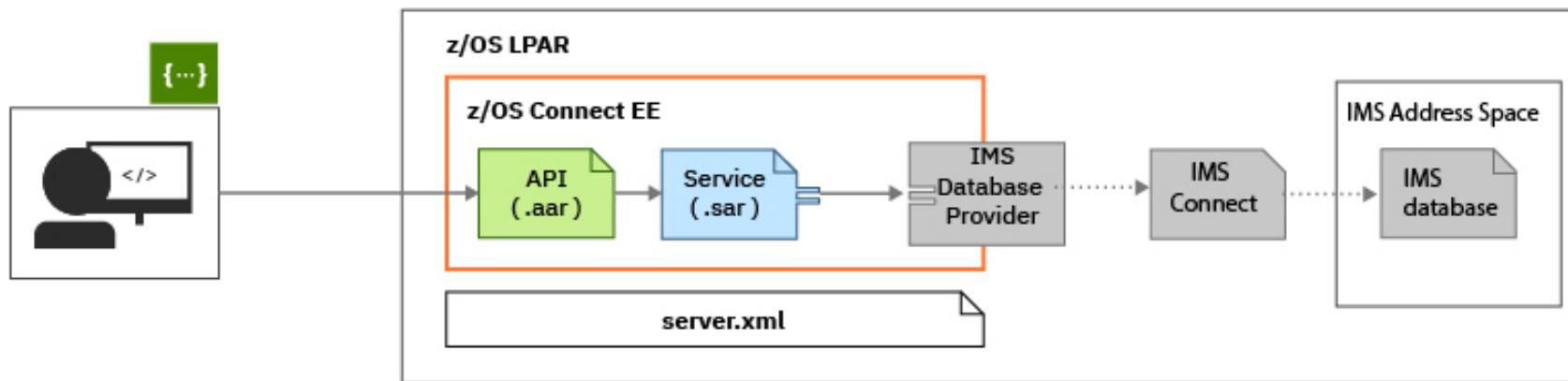
<authData id="Connection1_Auth" password="encryptedPassword1" user="userName1" />
</server>
```

## Interaction

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

# Connections to IMS DB

## Topology



Configure the connection to IMS using a Connection Factory in server.xml

Use the **API toolkit** to configure the service.

 [ibm.biz/zosconnect-scenarios](http://ibm.biz/zosconnect-scenarios)

# IMS Connection Factory



z/OS Connect EE

Service Project Editor: Configuration

Required Configuration

Enter the required configuration for this service.

Connection profile: DFSIVPACConn

## ConnectionFactory

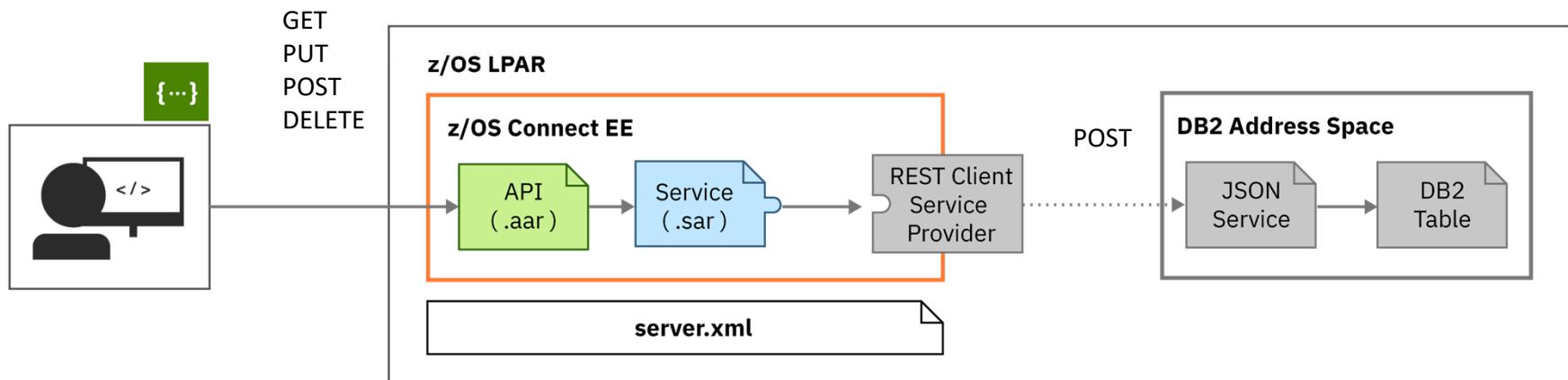
```
<connectionFactory id="DFSIVPACConn">
<properties.imsudbJLocal
  databaseName="DFSIVPA"
  datastoreName="IVP1"
  datastoreServer="wg31.washington.ibm.com"
  driverType="4"
  portNumber="5555"
  user="USER1"
  password="password"
  flattenTables="True" />
</connectionFactory>
```

## IMS Connect HWSCFG

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

# Connections to Db2

## Topology



Connection to the JSON Service is configured in **server.xml**.

A Db2 REST Service must be configured in DB2.

 [ibm.biz/zosconnect-db2-rest-services](http://ibm.biz/zosconnect-db2-rest-services)

# The server.xml File (Db2)



z/OS Connect EE

The server.xml file is the key configuration file:

The screenshot shows the Service Project Editor interface with the title "Service Project Editor: Configuration" and a tab for "Required Configuration". A connection reference "db2conn" is selected. On the left, there is a log output window showing various system messages. On the right, the "db2pass.xml" configuration file is displayed in a code editor with tabs for "Design" and "Source". The "Source" tab contains the XML code for the server definition.

```
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
```

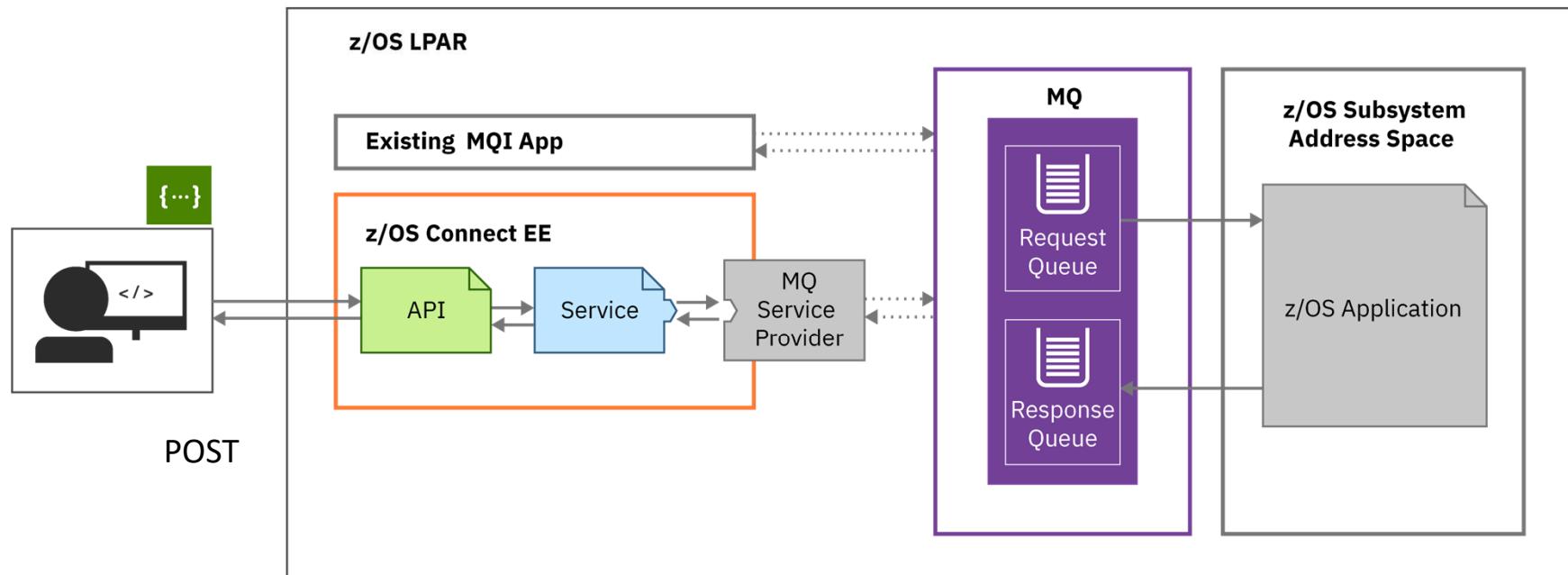
© 2018, 2020 IBM Corporation

Log Output (Left):

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

# Connections to MQ

Topology (Two-way service example)

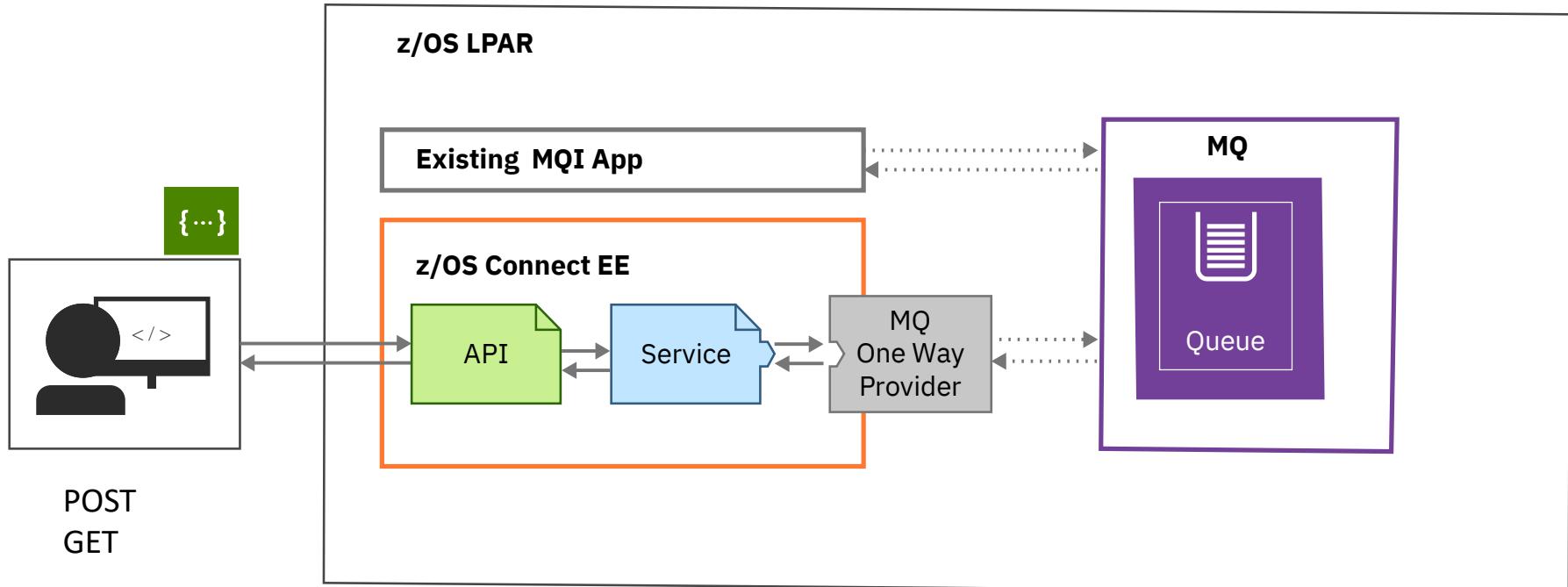


You can also configure one-way services.

 [ibm.biz/zosconnect-mq-service-provider](http://ibm.biz/zosconnect-mq-service-provider)

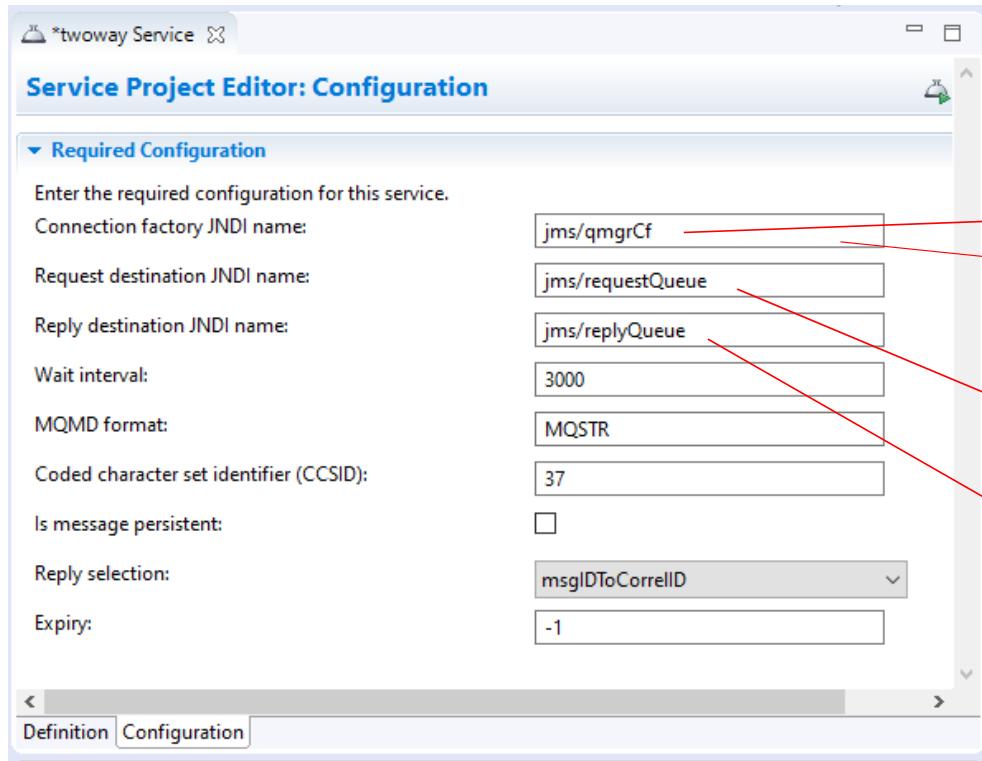
# Connections to MQ

Topology (One-way service example)



 [ibm.biz/zosconnect-mq-service-provider](http://ibm.biz/zosconnect-mq-service-provider)

# The server.xml File (MQ)

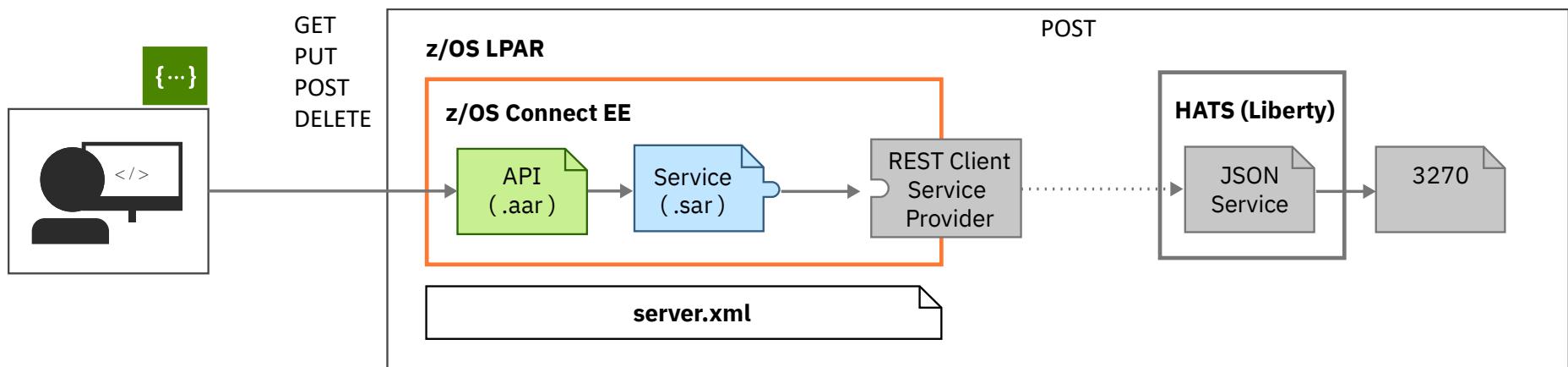


```
mq.xml
Read only Close
Design Source
2 <featureManager>
3   <feature>zosconnect:mqService-1.0</feature>
4 </featureManager>
5
6 <variable name="wmqJmsClient.rar.location"
7   value="/usr/lpp/mqm/V9R1M1/java/lib/jca/wmq.jmsra.rar"/>
8 <wmqJmsClient nativeLibraryPath="/usr/lpp/mqm/V9R1M1/java/lib"/>
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="QM21" />
16 </jmsConnectionFactory>
17
18 <jmsConnectionFactory id="qmgrCf2" jndiName="jms/qmgrCf2"
19   connectionManagerRef="ConMgr1">
20   <properties.wmqJMS transportType="CLIENT"
21     queueManager="ZMQ1"
22     channel="LIBERTY.DEF.SVRCONN"
23     hostName="wg31.washington.ibm.com"
24     port="1422" />
25 </jmsConnectionFactory>
26
27 <jmsQueue id="q1" jndiName="jms/default">
28   <properties.wmqJMS
29     baseQueueName="ZCONN2.DEFAULT.MQZCEE.QUEUE"
30     CCSID="37"/>
31 </jmsQueue>
32
33 <jmsQueue id="requestQueue" jndiName="jms/request">
34   <properties.wmqJMS
35     baseQueueName="ZCONN2.TRIGGER.REQUEST"
36     targetClient="MQ"
37     CCSID="37"/>
38 </jmsQueue>
39
40 <jmsQueue id="replyQueue" jndiName="jms/replyQueue">
41   <properties.wmqJMS
42     baseQueueName="ZCONN2.TRIGGER.RESPONSE"
43     targetClient="MQ"
44     CCSID="37"/>
45 </jmsQueue>
46
47
```

MQ V9.1.1 Added support for remote queue managers.

# Connection to HATS

## Topology

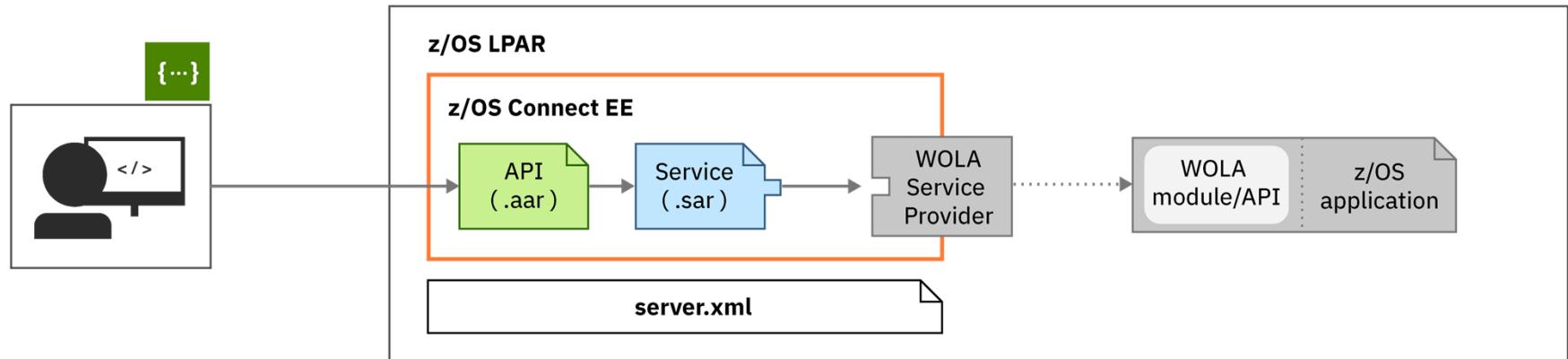


Connection to the HATS REST Service is configured in `server.xml`.

 [ibm.biz/zosconnect-db2-rest-services](http://ibm.biz/zosconnect-db2-rest-services)

# Connections to a MVS batch application

## Topology



Connection to WOLA is configured in `server.xml`.

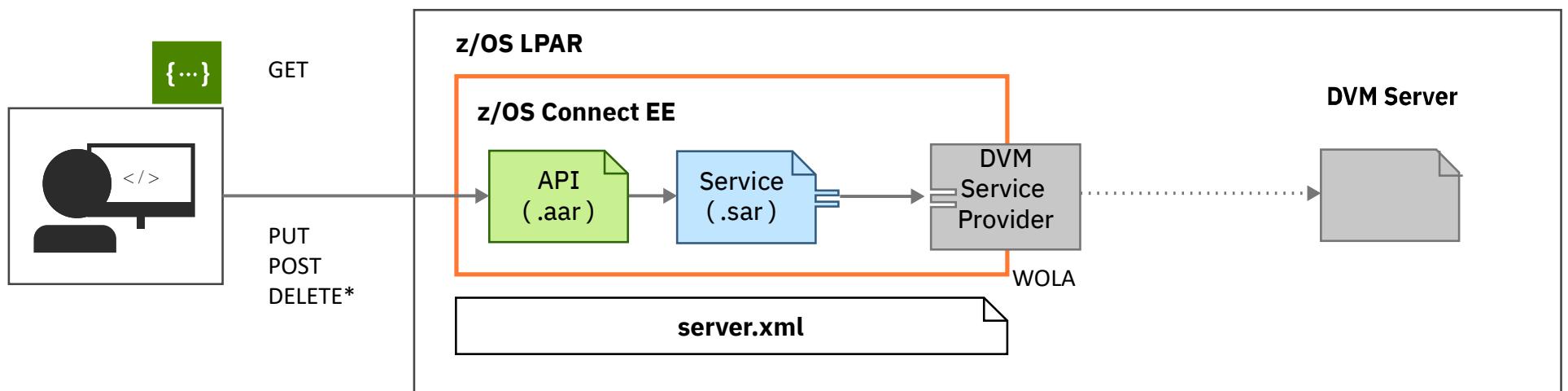
The z/OS application must be WOLA-enabled.

# Connections to DVM



z/OS Connect EE

## Topology



The DVM service provider uses WOLA

\* Requires a resource manager (e.g. RLS, VSAMCICS, etc.)

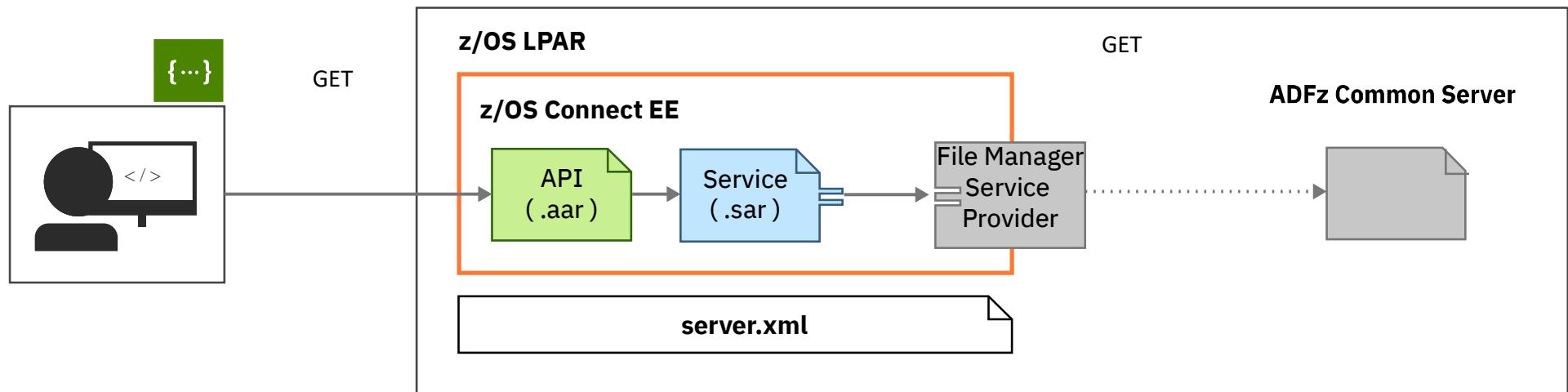
 [ibm.biz/zosconnect-db2-rest-services](http://ibm.biz/zosconnect-db2-rest-services)

# Connections to File Manager



z/OS Connect EE

## Topology



Connection to the Application Delivery Foundation for z (ADFz) common server is over TCP/IP

A File Manager Template is required .



## /miscellaneousTopics

performance, high availability, Liberty

# A Tour of Server Configuration Directories and Files



z/OS Connect EE

A z/OS Connect EE V3.0 server configuration structure looks like this:

```
/var/zosconnect
  /servers
    /zceesrv1
      /logs
        messages.log
  /resources
    /zosconnect
      /apis
      /apiRequesters
      /rules
      /services
        server.xml
        server.env
    /workarea
```

The messages.log file is the key output file for messages about Liberty and the processing taking place in the Liberty server.

The /zosconnect directory is where we will place the deployed APIs, services, and API requester files

The server.xml file is the key configuration file. It is here that z/OS Connect EE V3.0 definitions go which define the essential backend connectivity.

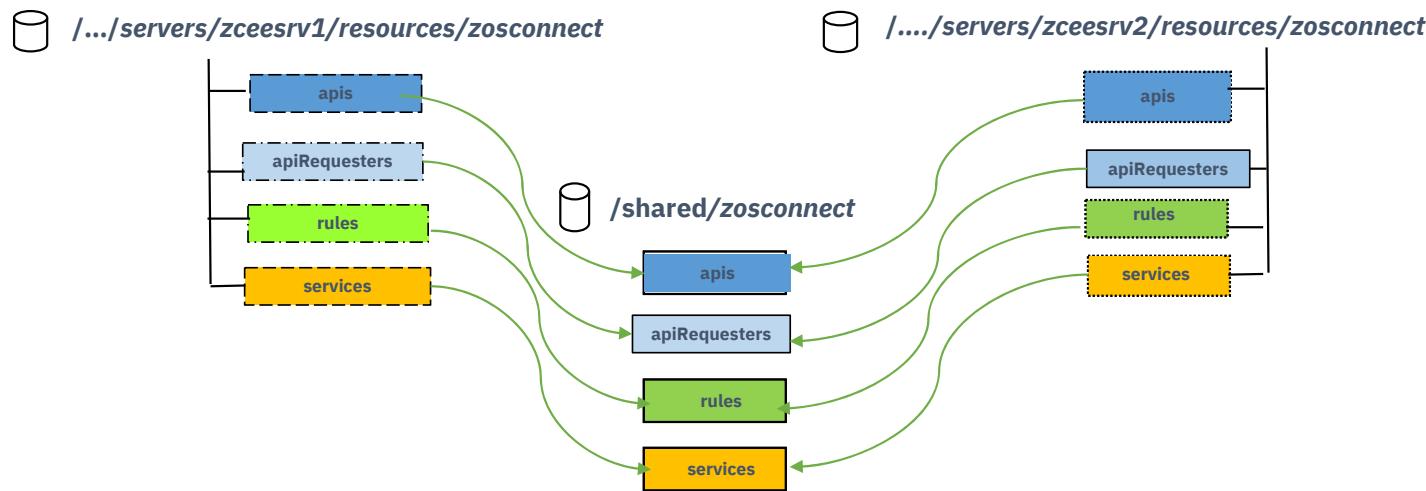
## server.xml

```
<server description="zCEE Server">
<include location="${server.config.dir}/includes/safSecurity.xml"/>
<include location="${server.config.dir}/includes/ipicIDProp.xml"/>
<include location="${server.config.dir}/includes/keyringOutboundMutual.xml"/>
<include location="${server.config.dir}/includes/groupAccess.xml"/>
<include location="${server.config.dir}/includes/shared.xml"/>
<include location="${server.config.dir}/includes/apiRequesterHTTPS.xml"/>
<include location="${server.config.dir}/includes/imsDatabase.xml"/>
```

-Dcom.ibm.ws.logging.log.directory=/u/johnson/logs

# Tour of Server Configuration Directories and Files

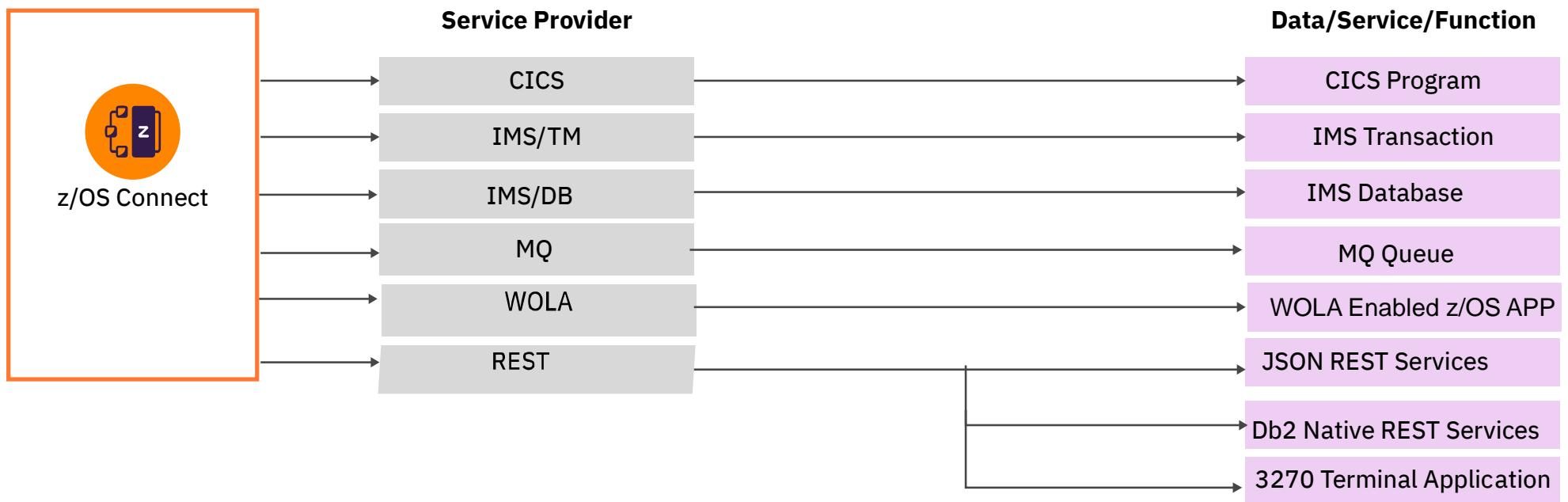
Using symbolic links to share application artifacts between servers:



```
cd /shared/zosconnect
ln -s ../../servers/zceesrv1/resources/zosconnect/apis apis
ln -s ../../servers/zceesrv1/resources/zosconnect/apiRequesters apiRequesters
ln -s ../../servers/zceesrv1/resources/zosconnect/rules rules
ln -s ../../servers/zceesrv1/resources/zosconnect/services services
ln -s ../../servers/zceesrv2/resources/zosconnect/apis apis
ln -s ../../servers/zceesrv2/resources/zosconnect/apiRequesters apiRequesters
ln -s ../../servers/zceesrv2/resources/zosconnect/rules rules
ln -s ../../servers/zceesrv2/resources/zosconnect/services services
```

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

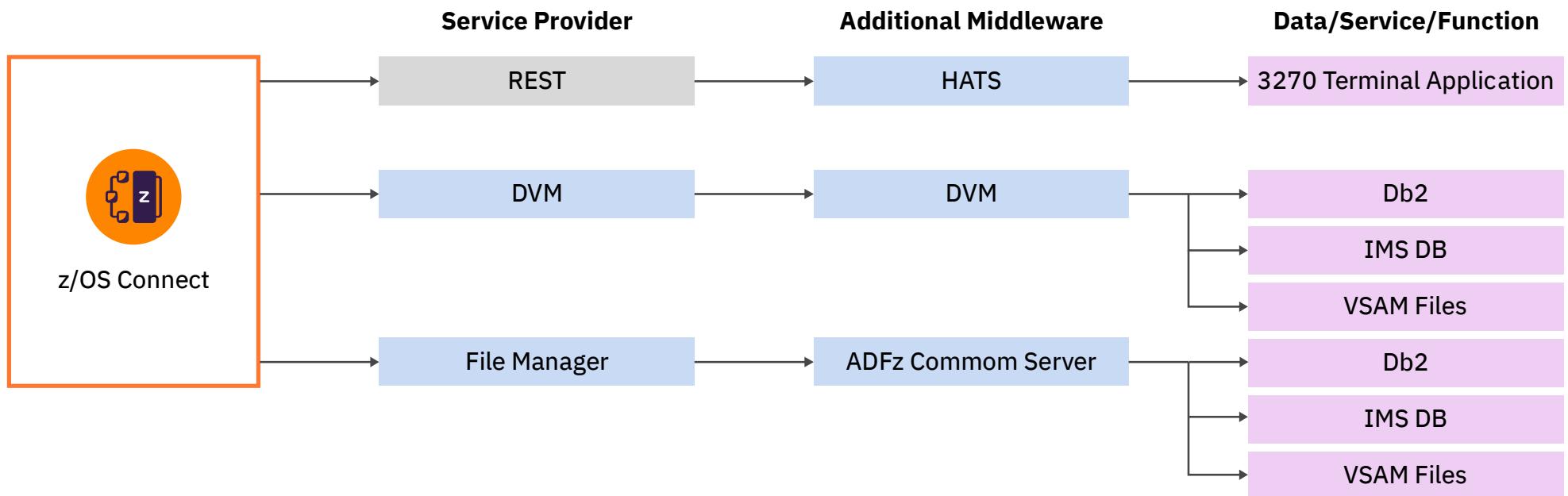
And which service provider could I use?



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

# Additional Middleware

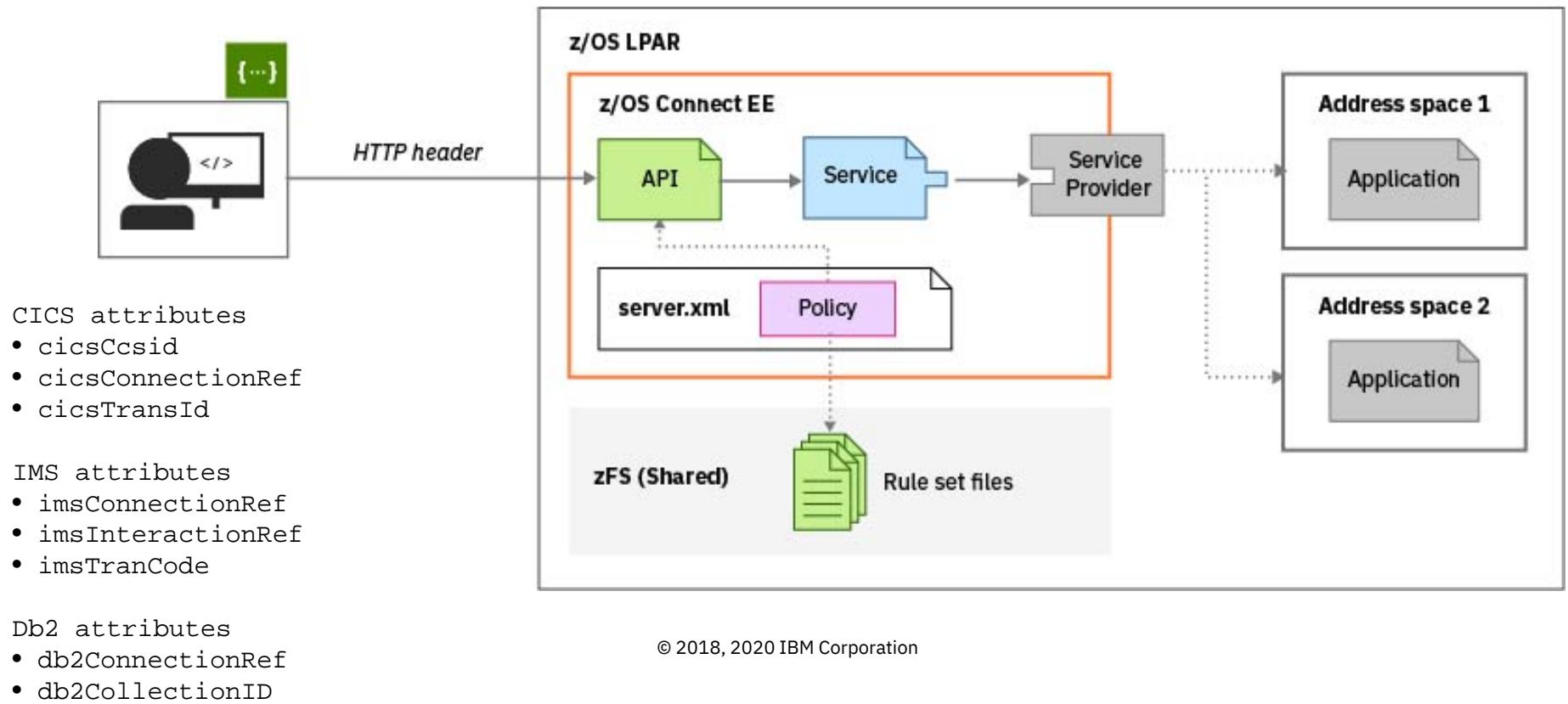
Additional value from the ecosystem



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) or Db2 (V3.0.36)
- Policies can be configured globally for every API in the server or for individual APIs (V3.0.11)



# A sample API Policies for CICS



z/OS Connect EE

```
<ruleset name="CICS rules">
    <rule name="csmi-rule">
        <conditions>
            <header name="cicsMirror" value="CSMI,MIJO"/>1
        </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...
```

<sup>1</sup>Transaction MIJO needs to be a clone of CSMI (e.g. invoke program DFHMIRS)



z/OS Connect EE

# Displaying zCEE messages on the console and/or spool

## server.xml

```
<zosLogging wtoMessage=
  "BAQR0657E,BAQR0658E,BAQR0660E,BAQR0686E,BAQR0687E"
  hardCopyMessage=
  "BAQR0657E,BAQR0658E,BAQR0660E,BAQR0686E,BAQR0687E"/>
```

## MVS Console

```
18.12.02 STC00137 +BAQR0686E: Program CSCVINC is not available in the CICS region with
  811           connection ID cscvinc; service cscvincService failed.
18.12.02 STC00137 +BAQR0686E: Program CSCVINC is not available in the CICS region with
  812           connection ID cscvinc; service cscvincService failed.
19.07.12 STC00137 +BAQR0657E: Transaction abend MIJO occurred in CICS while using
  745           connection cscvinc and service cscvincService.
```

## STDERR

```
ÝERROR    " BAQR0686E: Program CSCVINC is not available in the CICS region with connection cscvinc and service cscvincService.
ÝERROR    " BAQR0686E: Program CSCVINC is not available in the CICS region with connection cscvinc and service cscvincService.
ÝERROR    " BAQR0657E: Transaction abend MIJO occurred in CICS while using CICS connection cscvinc and service cscvincService.
```



# Liberty's “adminCenter” Feature

Web browser interface to the server's configuration files

The image shows two side-by-side screenshots of the z/OS Connect adminCenter web interface.

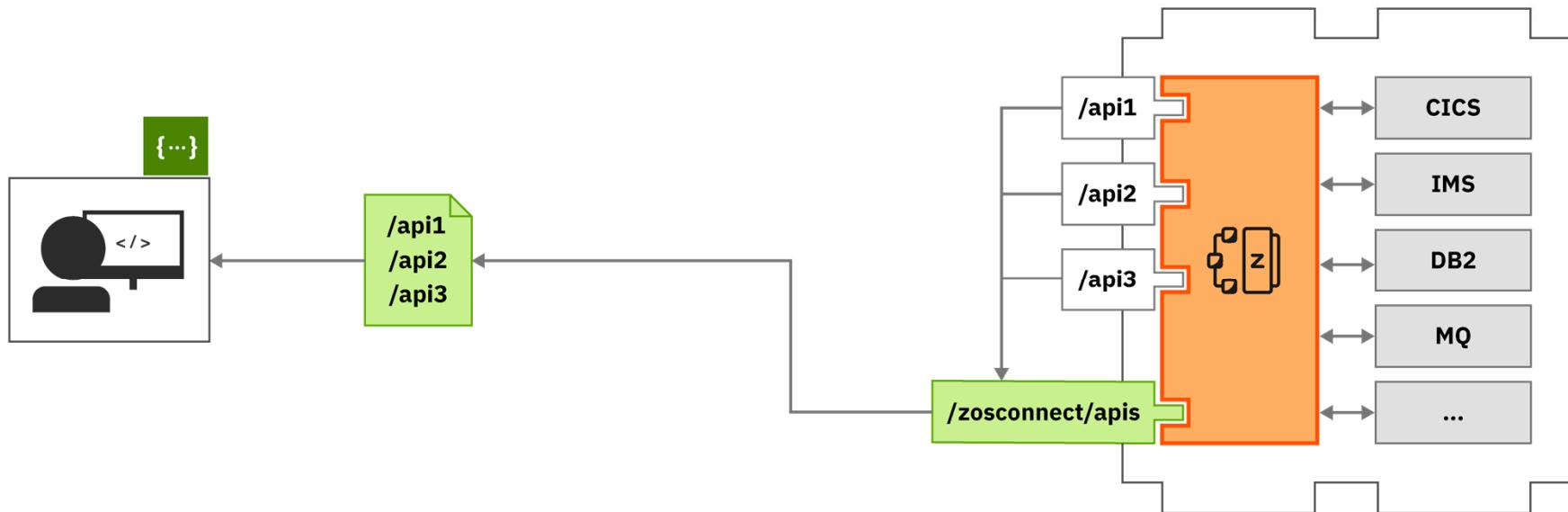
**Left Screenshot (server.xml):**

- Header:** Server Config
- Left Sidebar:** Design, Source, Includes (server.xml, wola.xml, hats.xml, ipic.xml, mq.xml, db2.xml, imsData.xml, adminC...), Feature Manager, z/OS Connect Manager, z/OS Logging, z/OS Connect policy name (cicsPolicy), omega... (omega...), z/OS Connect Interceptors (interceptor...), Cross-Origin Resource Sharing (default...), HTTP Endpoint (defaultHttpEndpoint), Configuration Management, z/OS Connect APIs, z/OS Connect Services, Application Monitoring.
- Right Content Area:**
  - Require request authentication:** false (Select)
  - Preserve JSON object payload order:** false (default) (Select)
  - Preserve JSON payload character format:** false (default) (Select)
  - Set response encoding to UTF-8:** false (default) (Select)
  - Return all errors in JSON format:** true (default) (Select)

**Right Screenshot (ipic.xml):**

- Header:** Server Config
- Left Sidebar:** Design, Source, Read only, Close, Settings gear icon.
- Right Content Area:**
  - z/OS Connect CICS IPIC connection:** Defines a connection that enables requests to call CICS programs via an IPIC connection.
  - Server:** Feature Manager
  - Feature:** zosconnect:cicsService-1.0
  - z/OS Connect CICS IPIC connection:** catalog, cscvinc, miniloan, miniloan1 (catalog is selected)
  - ID:** catalog
  - Host\***: wg31.washington.ibm.com
  - Port\***: 1491
  - Shared port:** false (default)
  - z/OS Connect APPLID:** (no value)
  - z/OS Connect network ID:** (no value)

# API Documentation



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

# 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`.

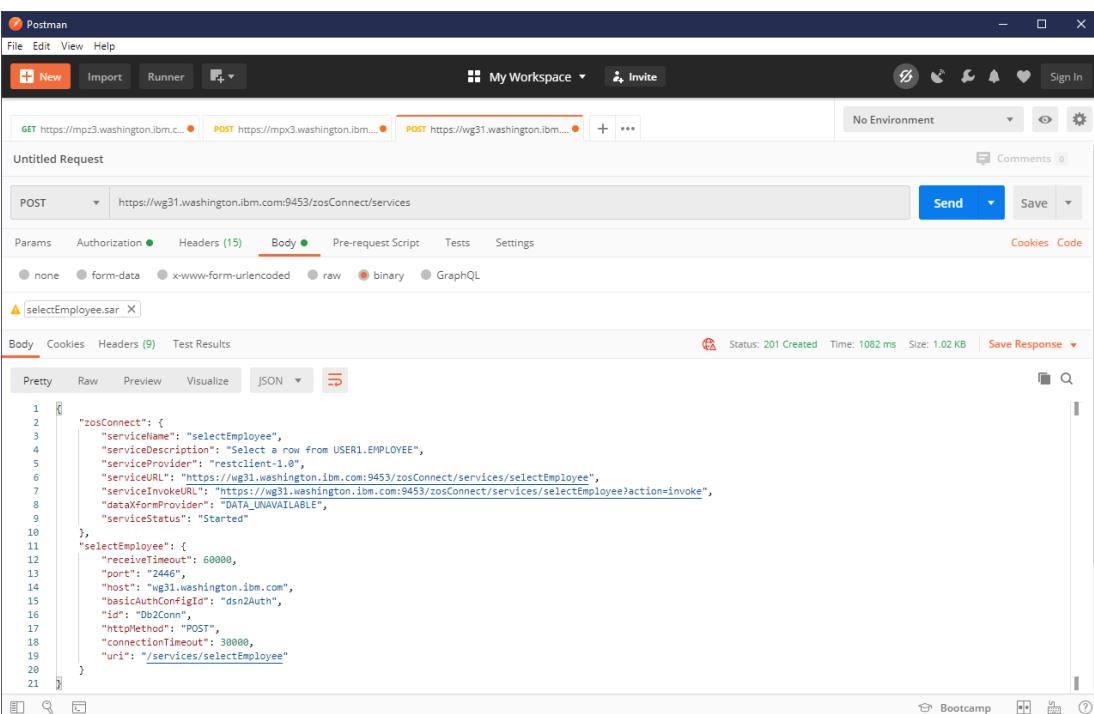
Most administration tasks are supported by the RESTful administration interface

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
```

# Deploying Service Archive options

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



The screenshot shows the Postman application interface. A POST request is being made to <https://wg31.washington.ibm.com:9453/zosConnect/services>. The 'Body' tab is selected, and a file named 'selectEmployee.sar' is attached. The file content is displayed in a code editor:

```

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

```

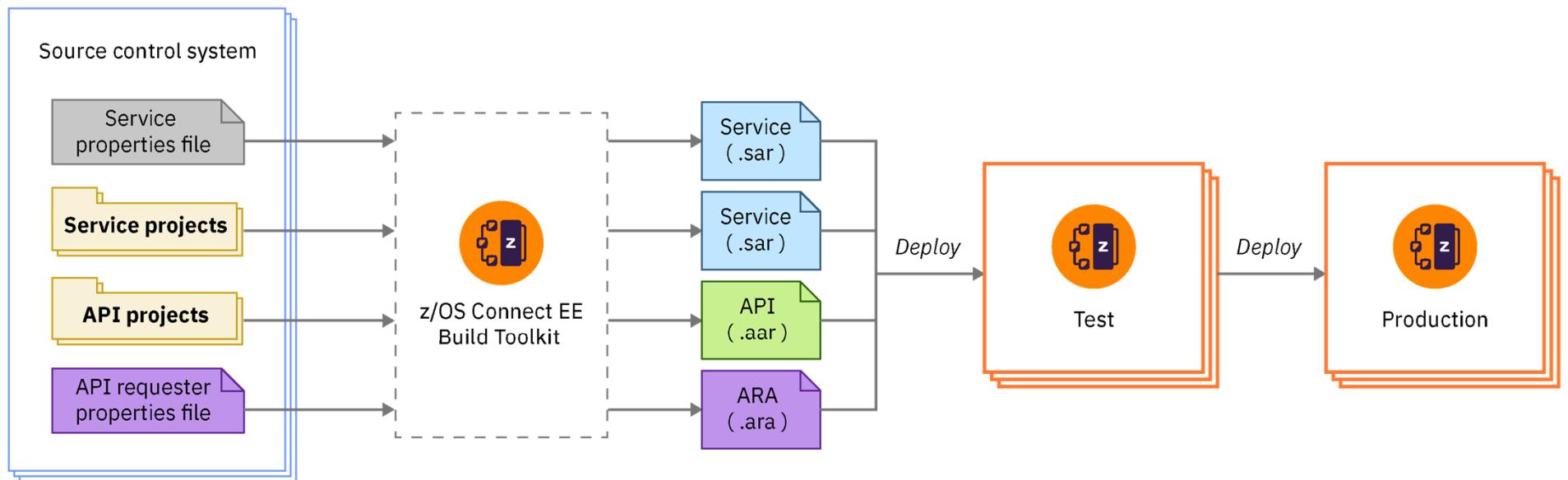
{
  "zosConnect": {
    "serviceName": "selectEmployee",
    "serviceDescription": "Select a row from USER1.EMPLOYEE",
    "serviceProvider": "IBM_ZOS_CONNECT_SERVICE_REST_CLIENT-1.0",
    "serviceURL": "https://mpxm:9453/zosConnect/services/selectEmployee",
    "serviceInvokeURL": "https://mpxm:9453/zosConnect/services/selectEmployee?action=invoke",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

```

# DevOps using z/OS Connect EE

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

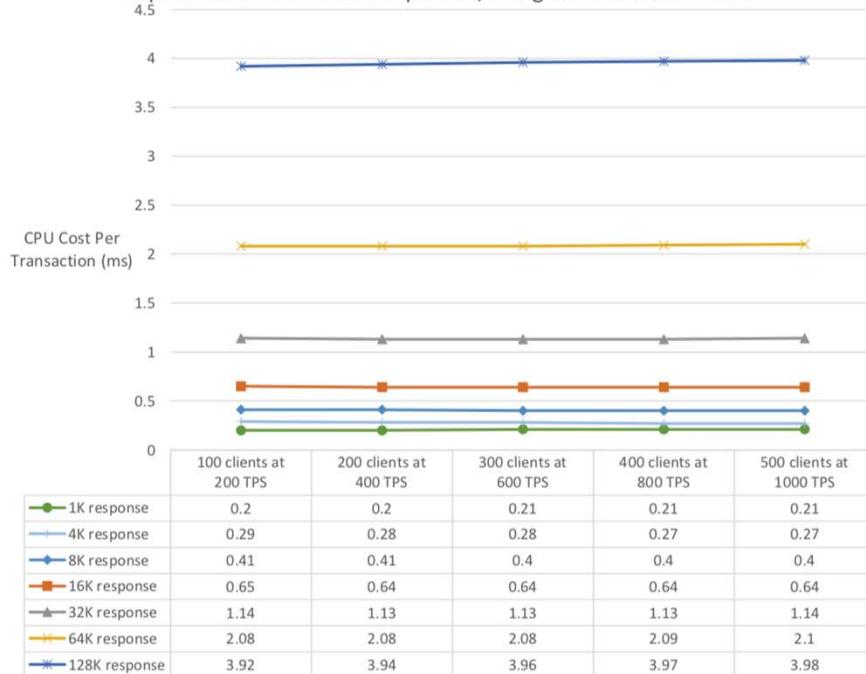
- The build toolkit supports the generation of service archives and API archives from projects created in the z/OS Connect EE API toolkit
- The build toolkit also supports the use of properties files to generate API requester archives
- Run the build toolkit from a build script to generate these archive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



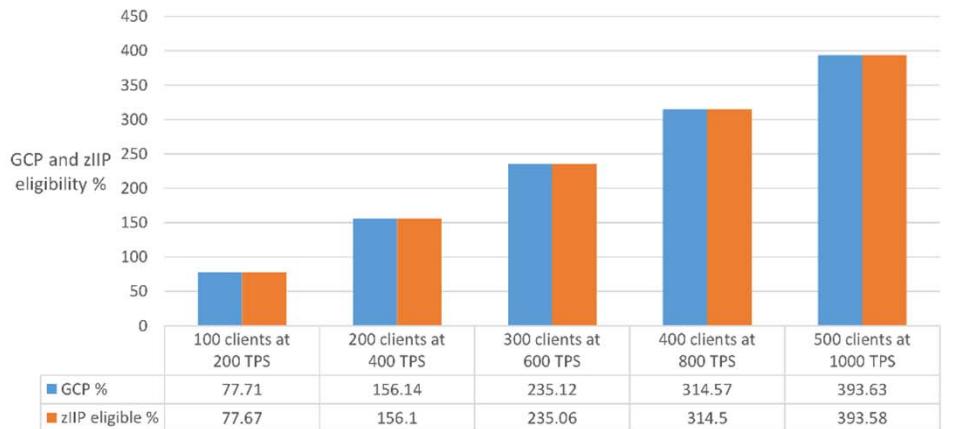
# Performance: API Provider

## High Speed, High Throughput, Low Cost

CPU Cost Per Transaction - increasing number of clients with 50 byte requests and 1K to 128K responses, using channels and CICS SP



zIIP eligibility - increasing number of clients with 50 byte requests and 128K responses, using channels and CICS SP

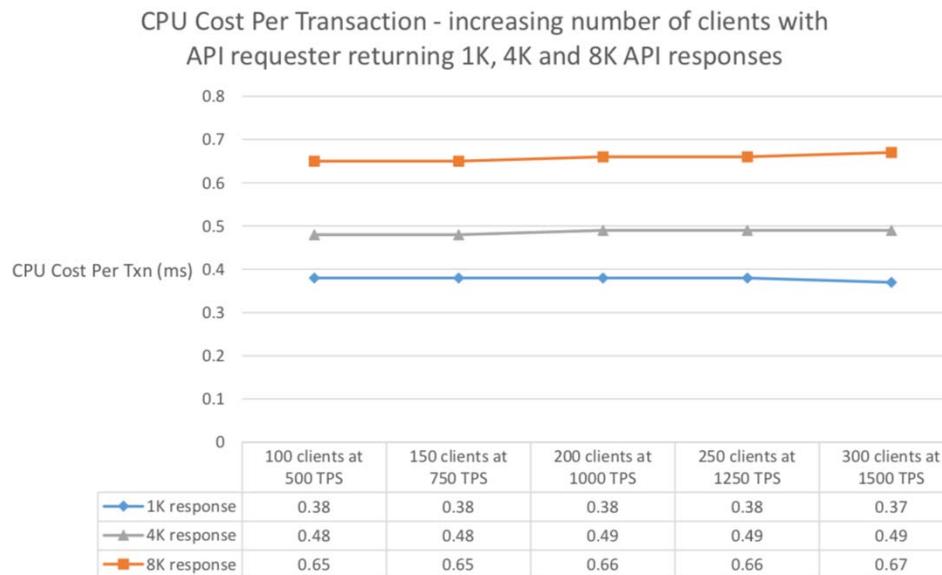


**z/OS Connect EE is a Java-based product:  
Over 99% of its MIPs are eligible for ZIIP offload.**



# Performance: API Requester

High Speed, High Throughput, Low Cost

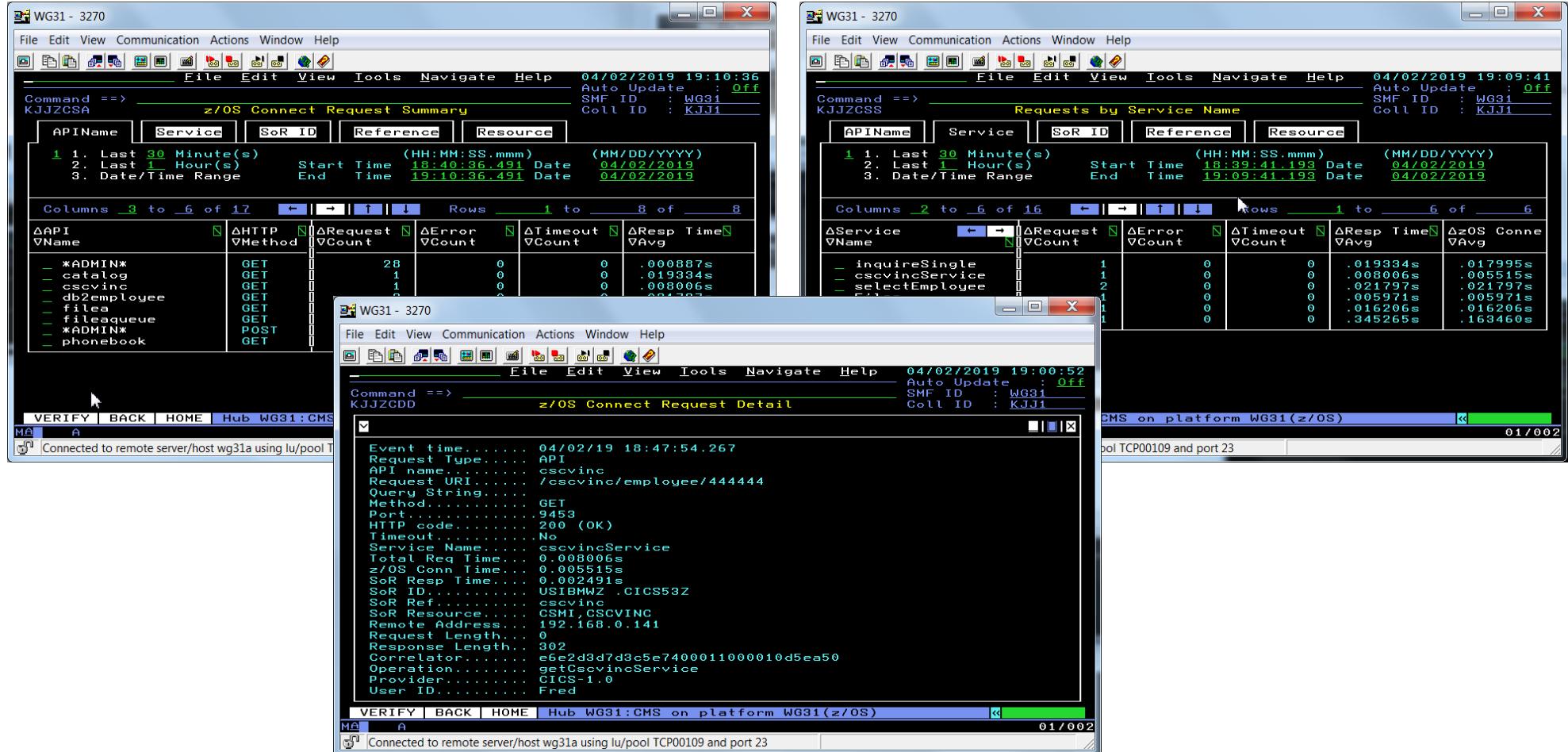


z/OS Connect EE is a Java-based product:  
Over **99%** of its MIPs are **eligible for ZIIP offload**.



[ibm.biz/zosconnect-performance-report](http://ibm.biz/zosconnect-performance-report)

# IBM z Omegamon for JVM



# IBM z Omegamon for JVM



WG31 - 3270

File Edit View Communication Actions Window Help

File Edit View Tools Navigate Help 04/02/2019 18:59:29  
Auto Update : Off  
SMF ID : WG31  
Coll ID : KJJ1

Command ==> z/OS Connect Request Detail

```

Event time..... 04/02/19 18:49:14.525
Request Type... API
API name..... filequeue
Request URI.... /filequeue/mq
Query String...
Method..... GET
Port..... 9453
HTTP code.... 200 (OK)
Timeout..... No
Service Name... FileaQueue
Total Req Time. 0.016206s
z/OS Conn Time. 0.016206s
SoR Resp Time. 0.000000s
SoR ID..... NONE
SoR Ref.... NONE
SoR Resource.. NONE
Remote Address.. 192.168.0.141
Request Length.. 0
Response Length. 191
Correlator.... e6e2d3d7d3c5e7400011000010d5ea51
Operation.... getFilea
Provider..... IBM MQ for z/OS
User ID..... Fred

```

VERIFY BACK HOME Hub WG31:CMS on platform WG31(z/OS) 01/002

Connected to remote server/host wg31a using lu/pool TCP00109 and port 23

```

Request Type... API
API name..... db2employee
Request URI.... /db2/employee/000020
Query String...
Method..... GET
Port..... 9453
HTTP code.... 200 (OK)
Timeout..... No
Service Name... selectEmployee
Total Req Time. 0.022592s
z/OS Conn Time. 0.022592s
SoR Resp Time. 0.000000s
SoR ID..... NONE
SoR Ref.... NONE
SoR Resource.. NONE
Remote Address.. 192.168.0.141
Request Length.. 0
Response Length. 326
Correlator.... e6e2d3d7d3c5e7400011000010d5ea50
Operation.... getSelectEmployee
Provider..... restclient-1.0
User ID..... Fred

```

VERIFY BACK HOME Hub WG31:CMS on platform WG31(z/OS) 01/002

Connected to remote server/host wg31a using lu/pool TCP00109 and port 23

WG31 - 3270

File Edit View Communication Actions Window Help

File Edit View Tools Navigate Help 04/02/2019 19:00:52  
Auto Update : Off  
SMF ID : WG31  
Coll ID : KJJ1

Command ==> z/OS Connect Request Detail

```

Event time..... 04/02/19 18:47:54.267
Request Type... API
API name..... cscvinc
Request URI.... /cscvinc/employee/444444
Query String...
Method..... GET
Port..... 9453
HTTP code.... 200 (OK)
Timeout..... No
Service Name... cscvincService
Total Req Time. 0.000006s
z/OS Conn Time. 0.005515s
SoR Resp Time. 0.002491s
SoR ID..... USIBMWZ.CICS532
SoR Ref.... cscvinc
SoR Resource.. CSM1,CSCVINC
Remote Address.. 192.168.0.141
Request Length.. 0
Response Length. 302
Correlator.... e6e2d3d7d3c5e7400011000010d5ea50
Operation.... getGscvincService
Provider..... CICS-1.0
User ID..... Fred

```

VERIFY BACK HOME Hub WG31:CMS on platform WG31(z/OS) 01/002

Connected to remote server/host wg31a using lu/pool TCP00109 and port 23

```

Request Type... API
API name..... phonebook
Request URI.... /phonebook/contacts/LAST1
Query String...
Method..... GET
Port..... 9453
HTTP code.... 200 (OK)
Timeout..... No
Service Name... ivtnoService
Total Req Time. 0.345265s
z/OS Conn Time. 0.163460s
SoR Resp Time. 0.181805s
SoR ID..... IVP1
SoR Ref.... IMSCONN
SoR Resource.. IVTNO
Remote Address.. 192.168.0.141
Request Length.. 0
Response Length. 158
Correlator.... e6e2d3d7d3c5e7400011000010d5ea55
Operation.... getPhoneBookService1
Provider..... imsmobile-2.0
User ID..... Fred

```

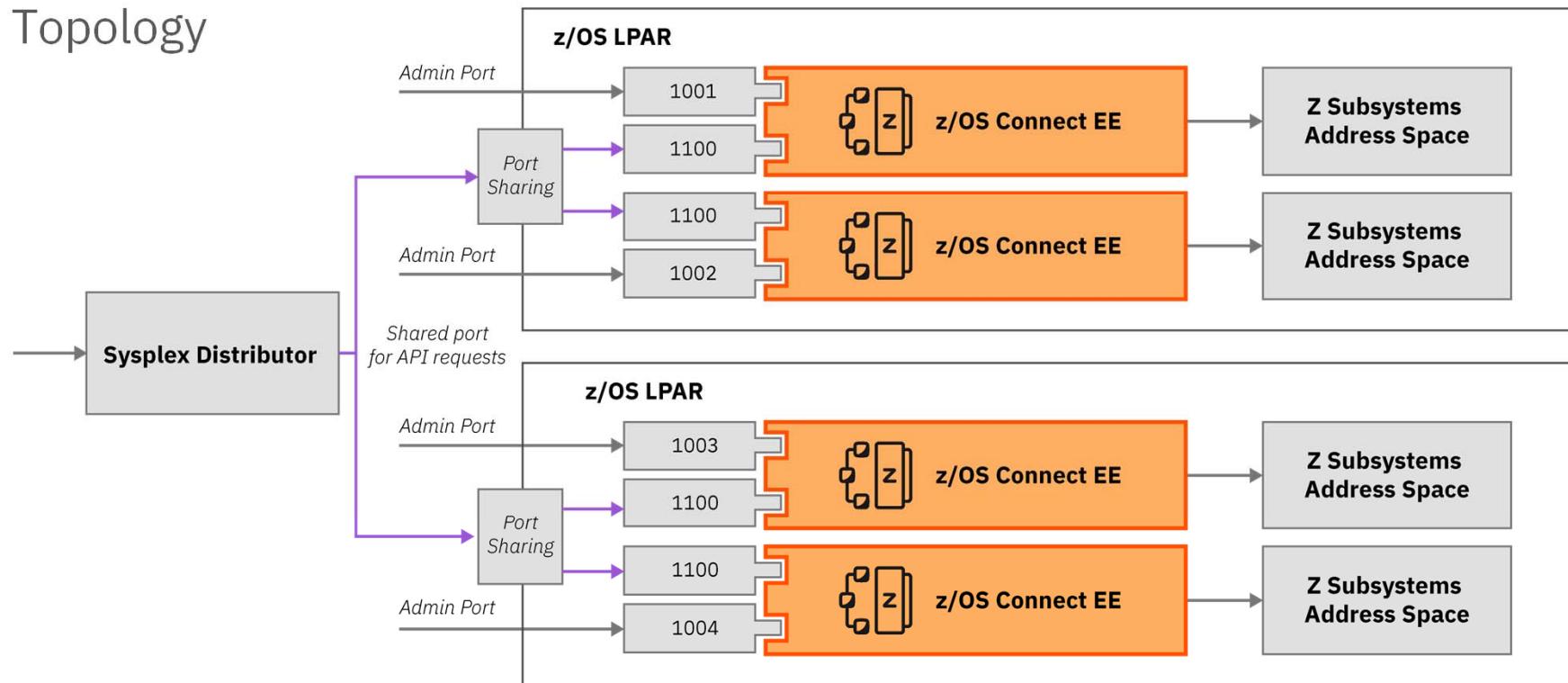
VERIFY BACK HOME Hub WG31:CMS on platform WG31(z/OS) 01/002

Connected to remote server/host wg31a using lu/pool TCP00109 and port 23



# High Availability

## Topology



**i** [ibm.biz/zosconnect-ha-concepts](http://ibm.biz/zosconnect-ha-concepts)

**i** [ibm.biz/zosconnect-scenarios](http://ibm.biz/zosconnect-scenarios)



## /security

How is security implement?



# Common security challenges

- **End-to-end security** is hampered by the issue of how to provide secure access between middleware components that use disparate security technologies e.g. registries
  - › This is a driver for implementing open security models like OAuth and OpenID Connect and standard tokens like JWT
- Security when using z/OS Connect is implemented in many products including z/OS Connect, WebSphere Liberty Profile on z/OS, SAF/RACF\*, CICS, IMS, Db2, MQ ...
  - › And these are all documented in different places
- Often security is at odds with **performance**, because the most secure techniques often involve the most processing overhead especially if not configured optimally

\*<https://knowledge.broadcom.com/external/article/128597/what-acf2-security-setup-is-needed-for-i.html>

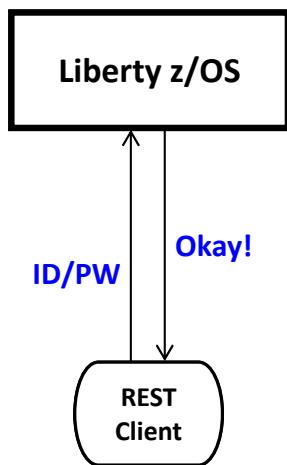
# Authentication



z/OS Connect EE

Several different ways this can be accomplished:

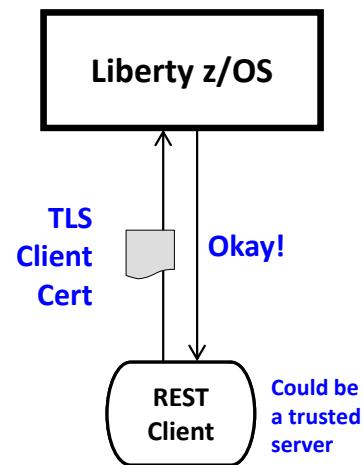
## Basic Authentication



Server prompts for ID/PW  
Client supplies ID/PW or ID/Passticket  
Server checks registry:

- Basic (server.xml)
- LDAP
- SAF

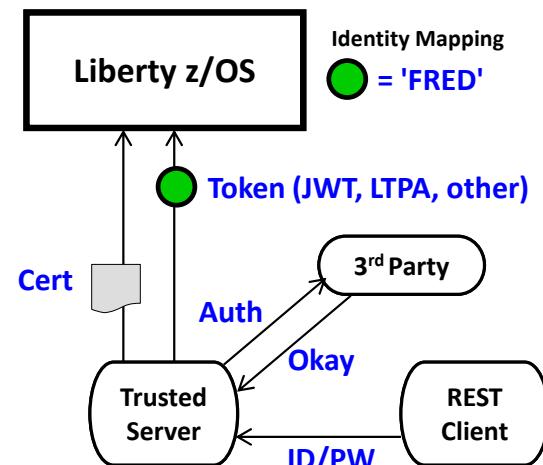
## Client Certificate



Server prompts for cert.  
Client supplies certificate  
Server validates cert and maps to an identity  
Registry options:

- LDAP
- SAF

## Third Party Authentication



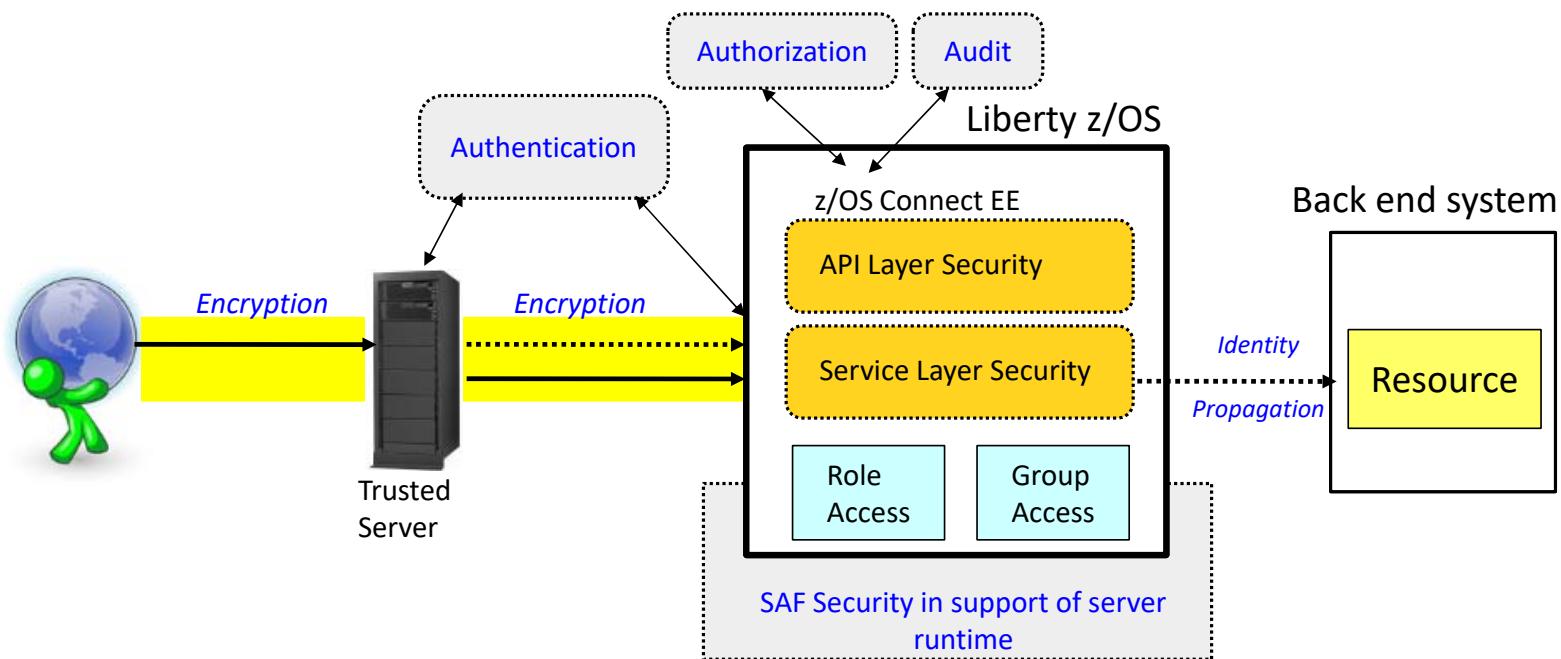
Client authenticates to 3<sup>rd</sup> party sever  
Client receives a trusted 3<sup>rd</sup> party token  
Token flows to Liberty z/OS and is mapped to an identity  
Registry options:

- LDAP
- SAF

# z/OS Connect EE API provider security overview



z/OS Connect EE

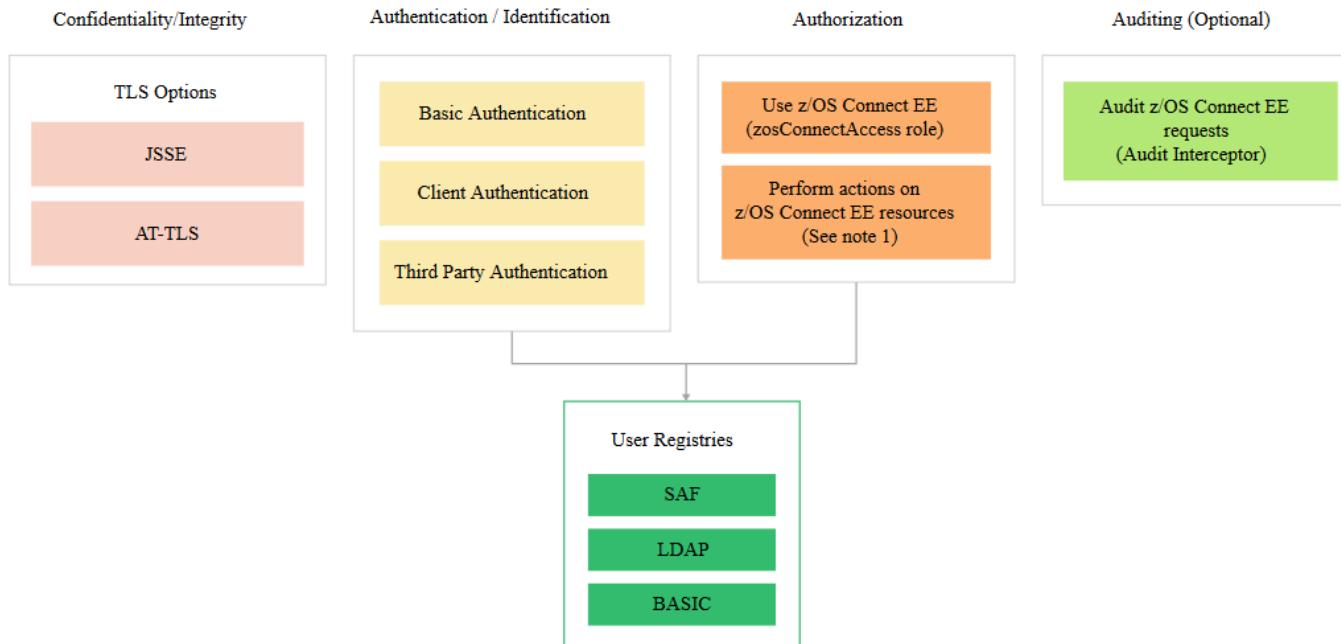


1. Authentication (basic, client certificates, 3<sup>rd</sup> party authentication)
2. Encryption (aka "SSL" or "TLS")
3. Authorization (role and group access)
4. Audit
5. Configuring security with SAF
6. Back end identity propagation (CICS, IMS, Db2, MQ)

See Dev Center article "Securing APIs with z/OS Connect EE" overview of z/OS Connect EE security

© 2018, 2020 IBM Corporation

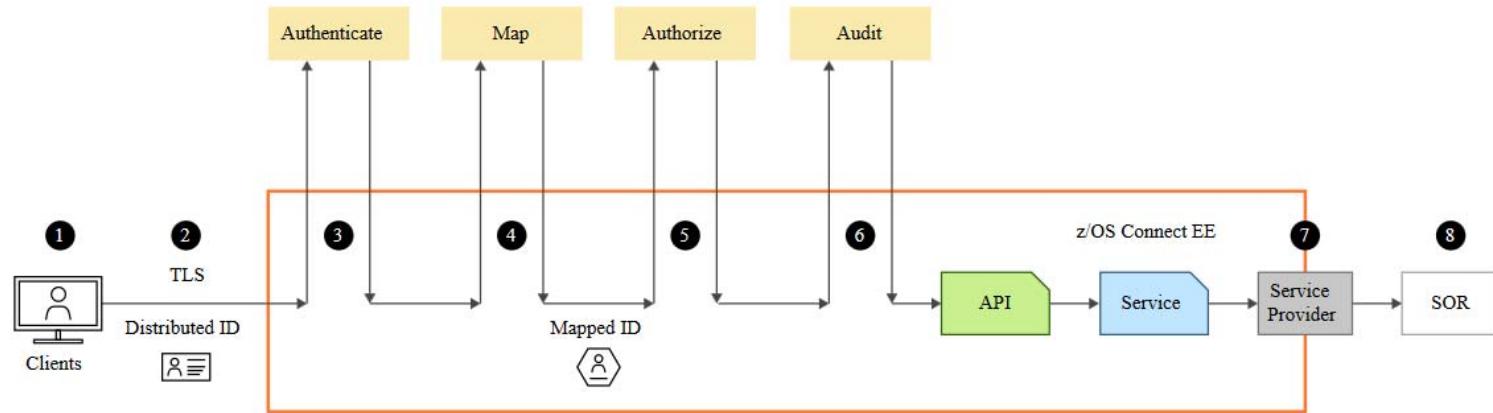
# z/OS Connect EE security options



 <http://ibm.biz/zosconnect-security>

The actions which can be controlled by authorization (see Note 1 in the diagram above) are: deploying, querying, updating, starting, stopping and deleting of APIs, services and API requesters.

## Typical z/OS Connect EE 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. Audit the API request
7. 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
8. The program or database request may run in the SoR under the mapped ID

# Security token types by z/OS Connect EE



z/OS Connect EE

Token type	How used	Pros	Cons
LTPA	Authentication technology used in IBM WebSphere	<ul style="list-style-type: none"><li>Easy to use with WebSphere and DataPower</li></ul>	<ul style="list-style-type: none"><li>IBM Proprietary token</li></ul>
SAML	XML-based security token and set of profiles	<ul style="list-style-type: none"><li>Token includes user id and claims</li><li>Used widely with SoR applications</li></ul>	<ul style="list-style-type: none"><li>Tokens can be heavy to process</li><li>No refresh token</li></ul>
OAuth 2.0 access token	Facilitates the authorization of one site to access and use information related to the user's account on another site	<ul style="list-style-type: none"><li>Used widely for SoE applications e.g with Google, Facebook, Microsoft, Twitter ...</li></ul>	<ul style="list-style-type: none"><li>Needs introspection endpoint to validate token</li></ul>
JWT	JSON security token format	<ul style="list-style-type: none"><li>More compact than SAML</li><li>Ease of client-side processing especially mobile</li></ul>	

See the YouTube video *OAuth 2.0 and OpenID Connect (in plain English)*

<https://www.youtube.com/watch?v=996OjexHze0> © 2018, 2020 IBM Corporation



# JWT (JSON Web Token)

- JWT is a compact way of representing claims that are to be transferred between two parties
- Normally transmitted via HTTP header
- Consists of three parts
  - Header
  - Payload
  - Signature

The screenshot shows the jwt.io website interface. On the left, under 'Encoded', is a long string of characters: eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiJkaXN0dXNlciIsInRva2VuX3R5cGUiOiJCZWFyZXIiLCJhenAiOiJycFNzbCIIsImlzcyI6Imh0dHBzOi8vd2czMS53YXNoaW5ndG9uLmlibSSjb206MjYyMTMvb21kYy9lbnRwb2ludC9PUHNzbCisImF1ZC16Im15WnN1ZSIsImV4cCI6MTU2OTY3NTY50CwiaWF0IjoxNTY5NjctInjM4LCJyZWFsbU5hbWUiOj6Q0VFUmVhbG0iLCJ1bmIxWVTZWN1cm10eU5hbWUiOjJkaXN0dXNlciJ9.a\_G\_1f\_vhD1u6\_z6-Kg5cprxPqkVe1K6-ngswuv4ZGecRdF9AU3E5Ig4o8qo0vmk\_0msqHu65Xe-Lxp0lo6Do1YZHI-cJg1jrrrnE6WMwdIUDi\_ayTTtGnMRYMeDdweE9eljgKZ3X2ChoYnm13p8V\_A7mIxuyomD6Vnsx4R1H0rwF4AS2RYHq3mBucK8vD-KZN0Z3eh307tg0ikD8Hadg3vhGvxd0Poh09ijQ7xgD0jg2ZG2Vuy0AdNeTRB2Gp5DP17s3aPzidMaDEFrUfn3GWknE07ylzW45NB1b1xSuVPP3HYssa3rvHBqrnl3bekIL2aF12dauaFogq7i-Cg. On the right, under 'Decoded', are the three parts:

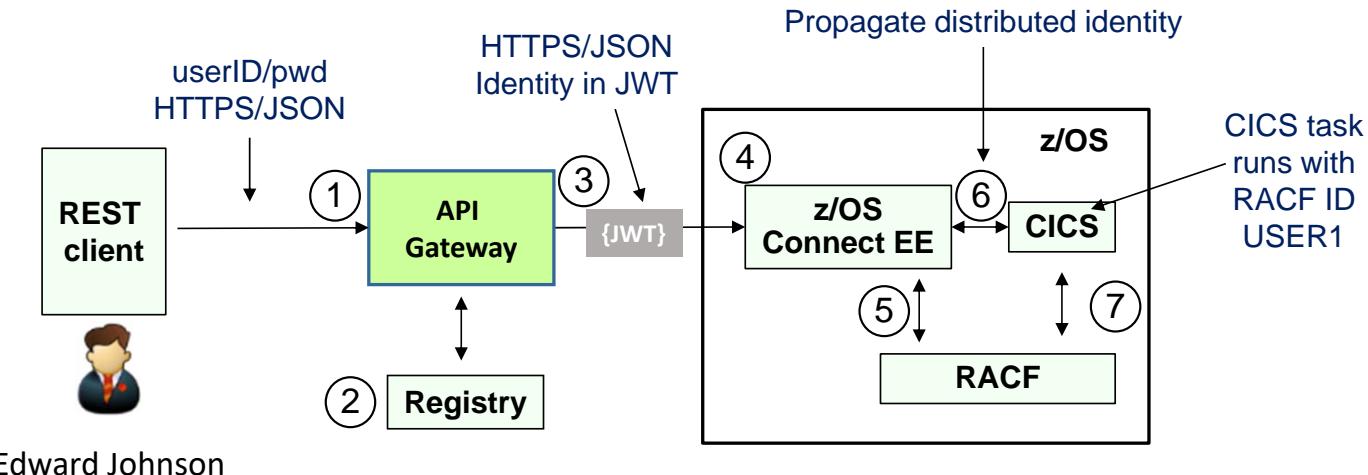
```

HEADER: ALGORITHM & TOKEN TYPE
{
  "alg": "RS256",
  "typ": "JWT"
}

PAYLOAD: DATA
{
  "sub": "distuser",
  "token_type": "Bearer",
  "azp": "rp0s1",
  "iss": "https://wg31.washington.ibm.com:26213/.oidc/endpoint/0/pss1",
  "aud": "myZee",
  "exp": 1569675698,
  "iat": 1569675638,
  "realmName": "zCEERealm",
  "uniqueSecurityName": "distuser"
}

VERIFY SIGNATURE
RSASHA256(
base64UrlEncode(header) + "." +
base64UrlEncode(payload),
-----BEGIN PUBLIC KEY-----
MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIIBCgKCAQEAmyz1sTjfNBbbbgKFMSv
vkTrwlvRsaIn7S5wA+kzeVOvnVWwk
-----END PUBLIC KEY-----
)
  
```

## Example scenario – security flow



```
RACMAP ID(USER1) MAP USERIDFILTER(NAME('auser')) REGISTRY(NAME('*'))
```

1. User authenticates with the managed API using a "distributed" identity and a password
2. An external registry is used as the user registry for distributed users and groups
3. API Gateway generates a JWT and forwards the token with the request to z/OS Connect EE
4. z/OS Connect EE validates JWT
5. z/OS Connect EE calls RACF to map distributed ID to RACF user ID and authorizes access to API
6. z/OS Connect EE CICS service provider propagates distributed ID to CICS
7. CICS calls RACF to map distributed ID to RACF user ID and performs resource authorization checks

## JWT used in scenario

```
{  
  "alg": "RS256"  
}  
{  
  "sub": "auser",  
  "token_type": "Bearer",  
  "azp": "rpSsl",  
  "iss": "https://wg31.washington.ibm.com:26213/oidc/endpoint/OP",  
  "aud": "myZcee",  
  "realmName": "zCEERealm",  
  "uniqueSecurityName": "auser"  
}  
RSASHA256(base64UrlEncode(header)+ base64UrlEncode(payload))
```

- The header contains an **alg** (algorithm) element value **RS256**
  - **RS256** (RSA Signature with SHA-256) is an asymmetric algorithm which uses a **public/private** key pair
  - **ES512** (Elliptic Curve Digital Signature Algorithm with SHA-512) [link for more info](#)
  - **HS256** (HMAC with SHA-256) is a symmetric algorithm with only one (**secret**) key
- The **iss** (issuer) claim identifies the principal that issued the JWT
- The **sub** (subject) claim **distuser** identifies the principal that is the subject of the JWT
- The **aud** (audience) claim **myZcee** identifies the recipients for which the JWT is intended



## Configuring authentication with JWT

z/OS Connect EE can perform user authentication with JWT using the support that is provided by the *openidConnectClient-1.0* feature. The **<openidConnectClient>** element is used to accept a JWT token as an authentication token

```
<openidConnectClient id="ATS" inboundPropagation="required"
    signatureAlgorithm="RS256" trustAliasName="JWT-Signer"
    trustStoreRef="jwtTrustStore"
    userIdentityToCreateSubject="sub" mapIdentityToRegistryUser="true"
    issuerIdentifier="https://wg31.washington.ibm.com:26213/oidc/endpoint/OPssl"
    authnSessionDisabled="true" audiences="myZcee" />
```

- ***inboundPropagation*** is set to required to allow z/OS Connect EE to use the received JWT as an authentication token
- ***signatureAlgorithm*** specifies the algorithm to be used to verify the JWT signature
- ***trustStoreRef*** specifies the name of the keystore element that defines the location of the validating certificate
- ***trustAliasName*** gives the alias or label of the certificate to be used for signature validation
- ***userIdentityToCreateSubject*** indicates the claim to use to create the user subject
- ***mapIdentityToRegistryUser*** indicates whether to map the retrieved identity to the registry user
- ***issuerIdentifier*** defines the expected issuer
- ***authnSessionDisabled*** indicates whether a WebSphere custom cookie should be generated for the session
- ***audiences*** defines a list of target audiences

See Dev Center article "Using a JWT with z/OS Connect EE" for full description of scenario



## Using authorization filters with z/OS Connect EE

Authentication filter can be used to filter criteria that are specified in the **authFilter** element to determine whether certain requests are processed by certain providers, such as OpenID Connect, for authentication.

```
<openidConnectClient id="RPssl" inboundPropagation="required"
    signatureAlgorithm="RS256" trustAliasName="JWT-Signer"
    trustStoreRef="jwtTrustStore"
    userIdentityToCreateSubject="sub" mapIdentityToRegistryUser="true"
    issuerIdentifier="https://wg31.washington.ibm.com:26213/oidc/endpoint/OP"
    authnSessionDisabled="true" audiences="myZcee"
    authFilterRef="JwtAuthFilter"/>
<authFilter id="API Gateway">
    <remoteAddress id="ApiAddress" ip="10.7.1.*" matchType="equals" />
</authFilter>
<authFilter id="Cscvinc">
    <requestUrl id="URL" urlPattern="/cscvinc/employee/*" matchType="equals" />
</authFilter>
<authFilter id="JwtAuthFilter" >
    <requestHeader id="authHeader" name="Authorization" value="Bearer" matchType="contains" />
</authFilter>
```

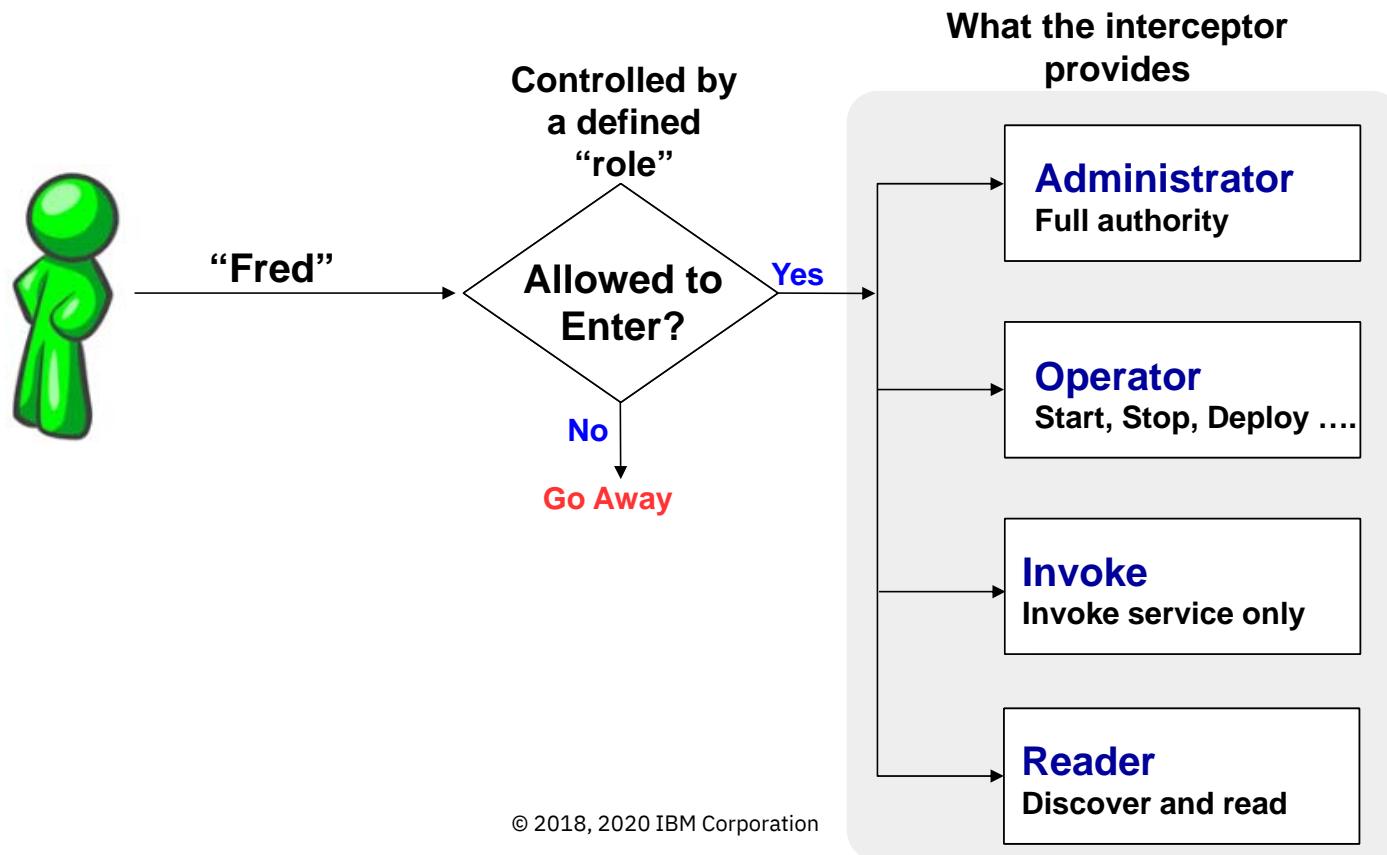
### Some alternative filter types

- A **remoteAddress** element is compared against the TCP/IP address of the client that sent the request.
- The **host** element is compared against the "Host" HTTP request header, which identifies the target host name of the request.
- The **requestUrl** element is compared against the URL that is used by the client application to make the request.



## Authorization interceptor

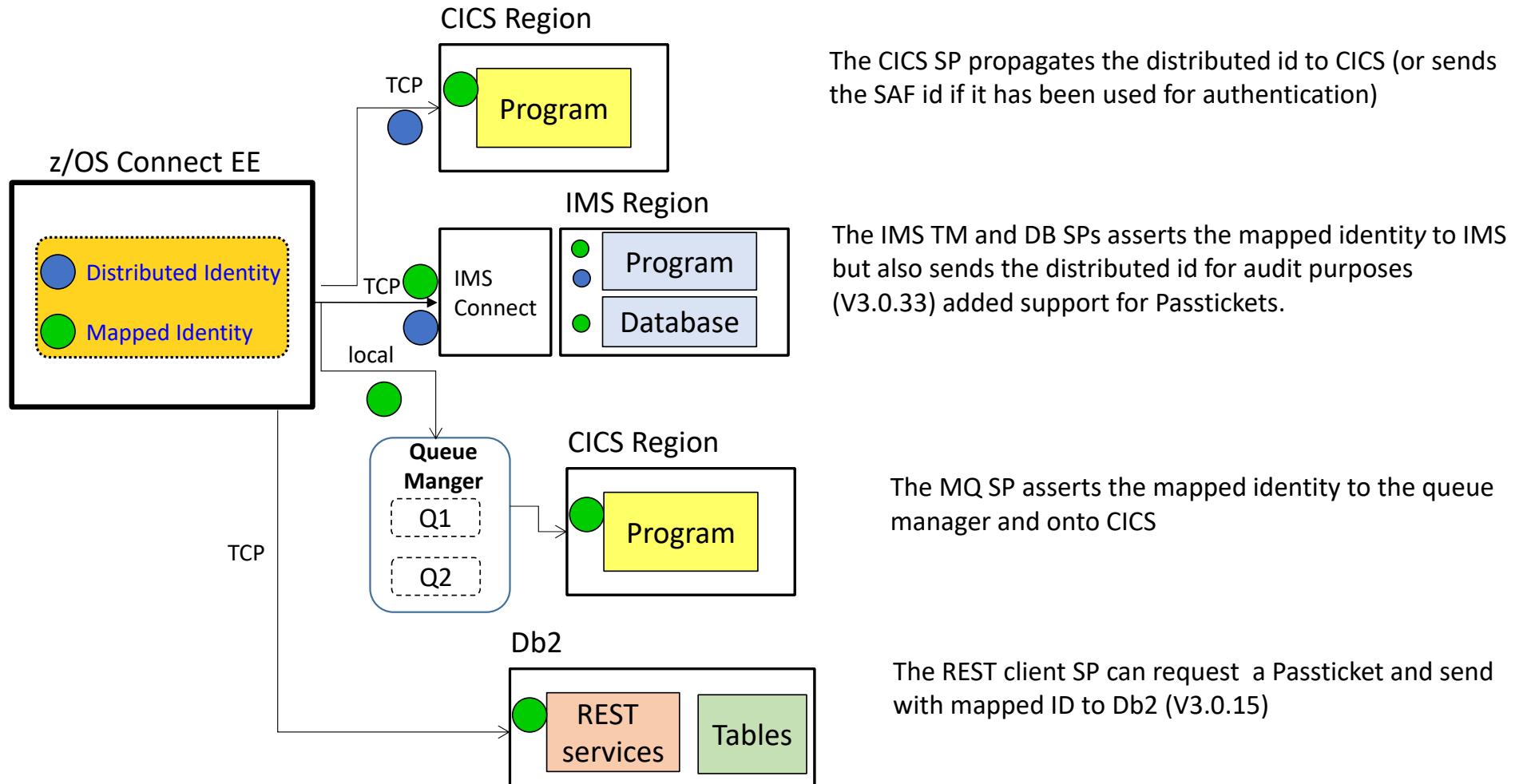
The “authorization interceptor” is a supplied piece of interceptor code that will check to see if the user has the authority to perform the action requested:



## Flowing an identity to the back end



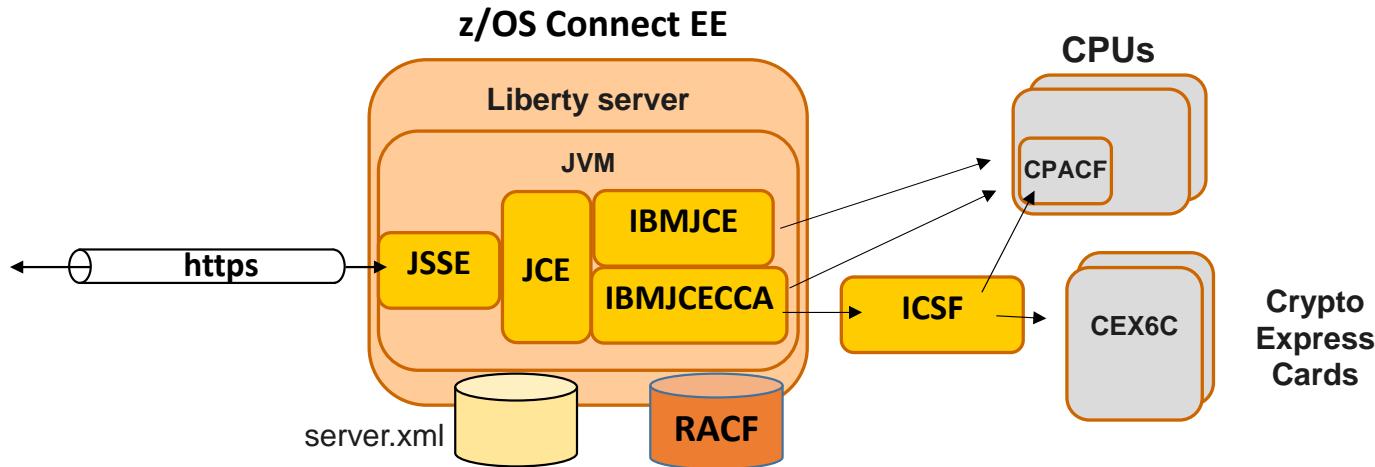
z/OS Connect EE



## Using JSSE with z/OS Connect EE

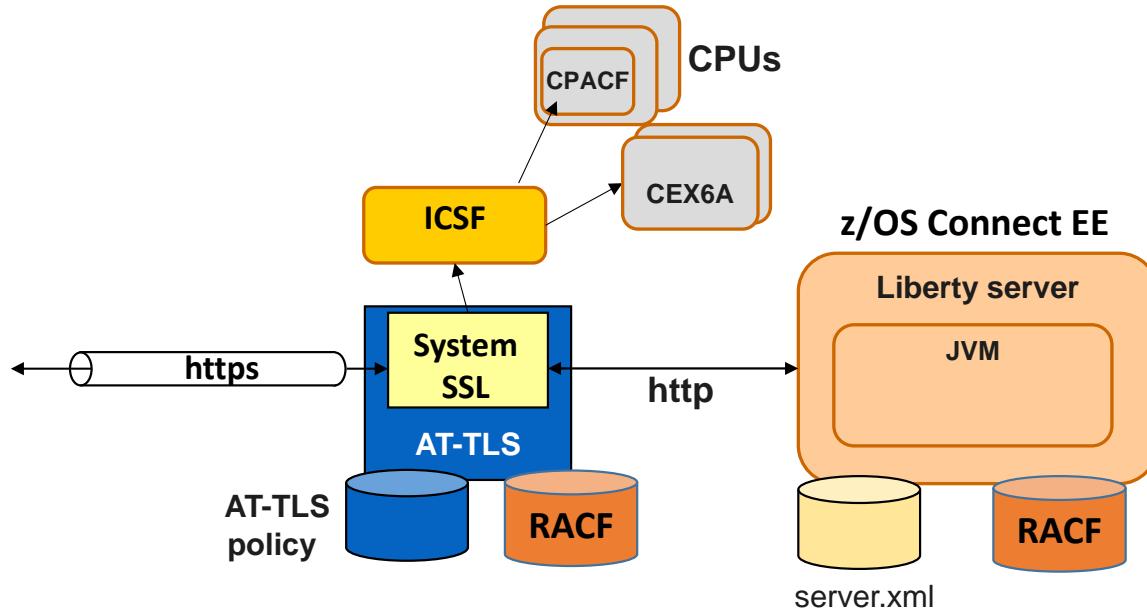


z/OS Connect EE



- z/OS Connect EE support for SSL/TLS is based on **Liberty server** support
- **Java Secure Socket Extension (JSSE)** API provides framework and Java implementation of SSL and TLS protocols used by Liberty HTTPS support
- **Java Cryptography Extension (JCE)** is standard extension to the Java Platform that provides implementation for cryptographic services
- **IBM Java SDK** for z/OS provides two different JCE providers, **IBMJCE** and **IBMJCECCA**

## Using AT-TLS with z/OS Connect EE



- **Application Transparent TLS (AT-TLS)** creates a secure session on behalf of z/OS Connect
- Only define http ports in **server.xml** (z/OS Connect does not know that TLS session exists)
- Define TLS protection for all applications (including z/OS Connect) in **AT-TLS policy**
- AT-TLS uses **System SSL** which exploits the CPACF and Crypto Express cards via ICSF

## JSSE and AT-TLS comparison



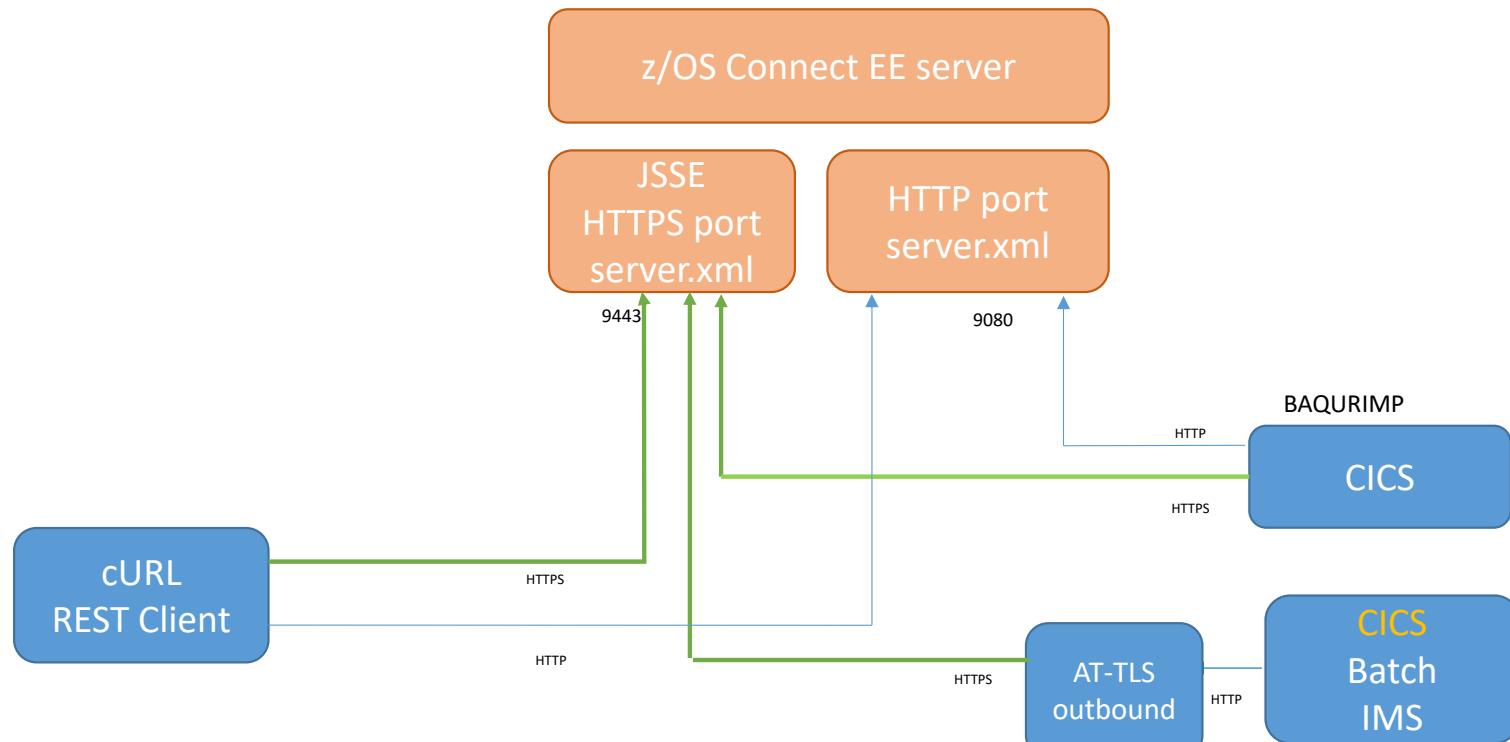
z/OS Connect EE

Capability	Description	JSSE	AT-TLS
1-way SSL	Verification of z/OS Connect certificate by client	Yes	Yes
2-way SSL	Verification of client certificate by z/OS Connect	Yes	Yes
SSL client authentication	Use of client certificate for authentication	Yes	No
Support for requireSecure option on APIs	Requires that API requests are sent over HTTPS	Yes	No
Persistent connections	To reduce number of handshakes	Yes	Yes
Re-use of SSL session	To reduce number of full handshakes	Yes	Yes
Shared SSL sessions	To share SSL sessions across cluster of z/OS Connect instances	No	Yes
zIIP processing	Offload TLS processing to zIIP	Yes	No
CPACF	Offload symmetric encryption to CPACF	Yes	Yes
CEX6	Offload asymmetric operations to Crypto Express cards	Yes	Yes



z/OS Connect EE

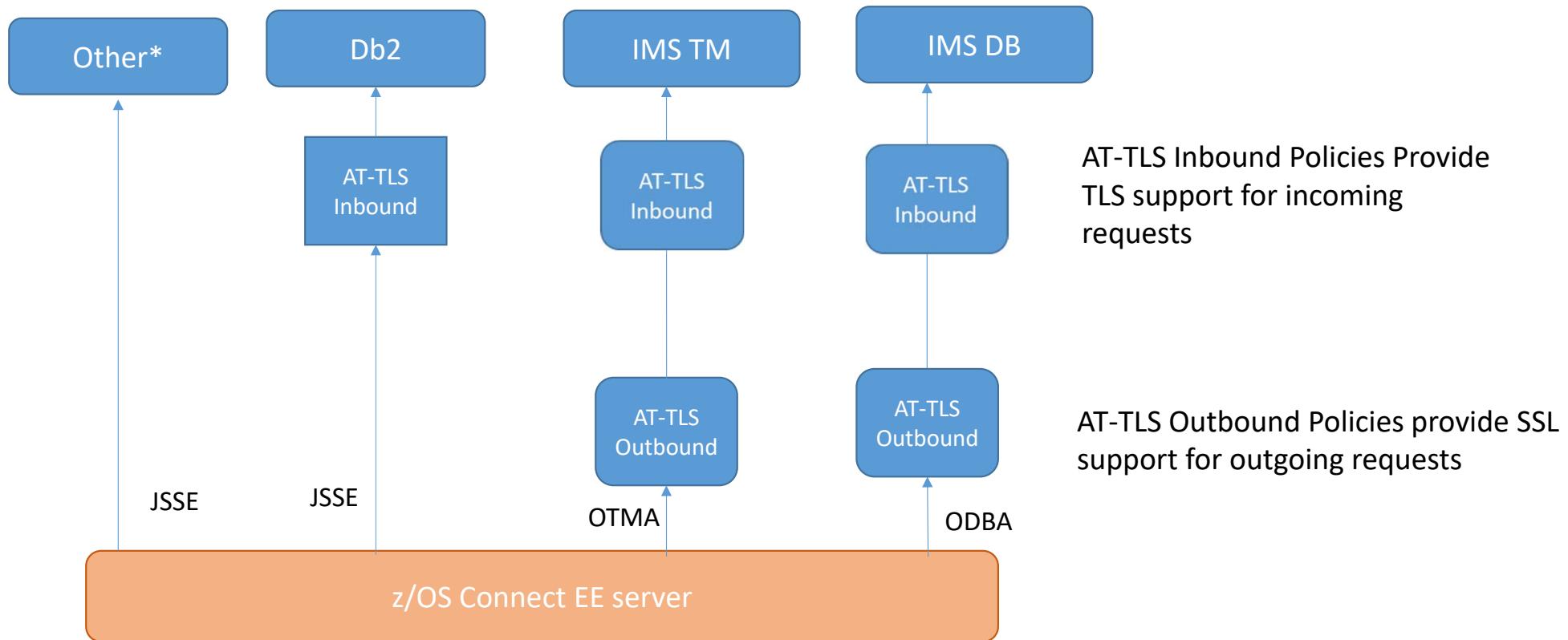
## TLS Inbound to zCEE Scenarios



# TLS Outbound from zCEE Scenarios (HTTPS/OTMA/ODBA)



z/OS Connect EE



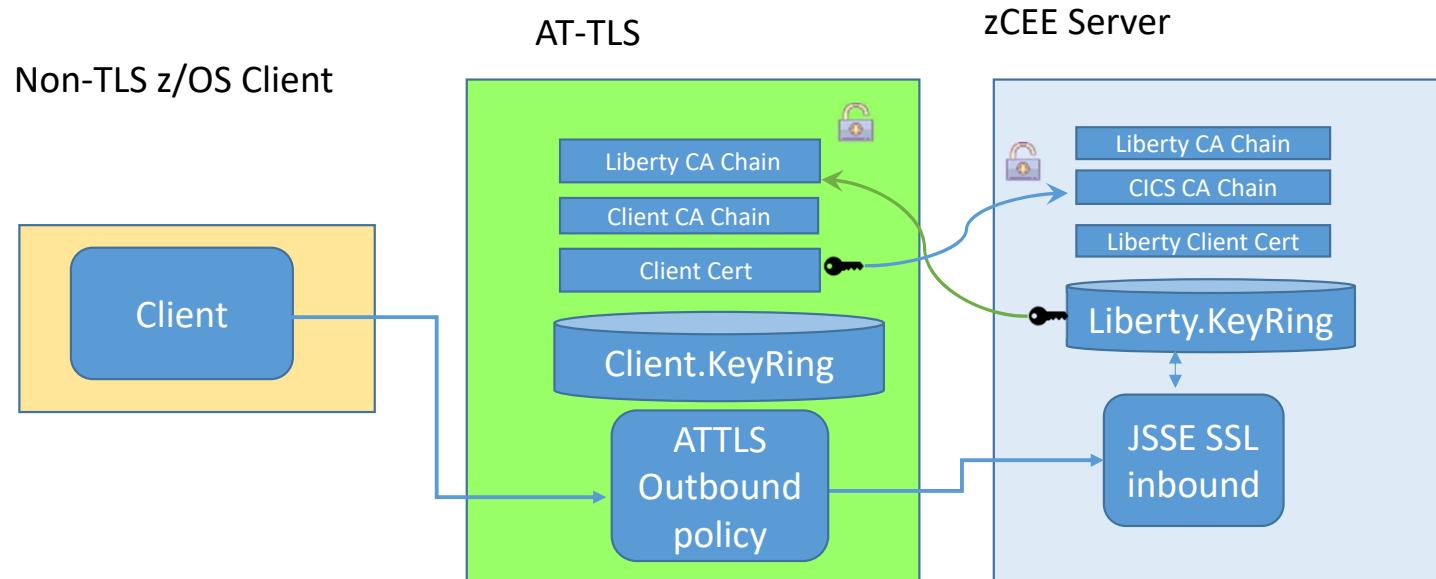
\* REST endpoints, CICS, MQ, etc.

© 2018, 2020 IBM Corporation

## Non-TLS Client with zCEE Inbound TLS Handshake



z/OS Connect EE

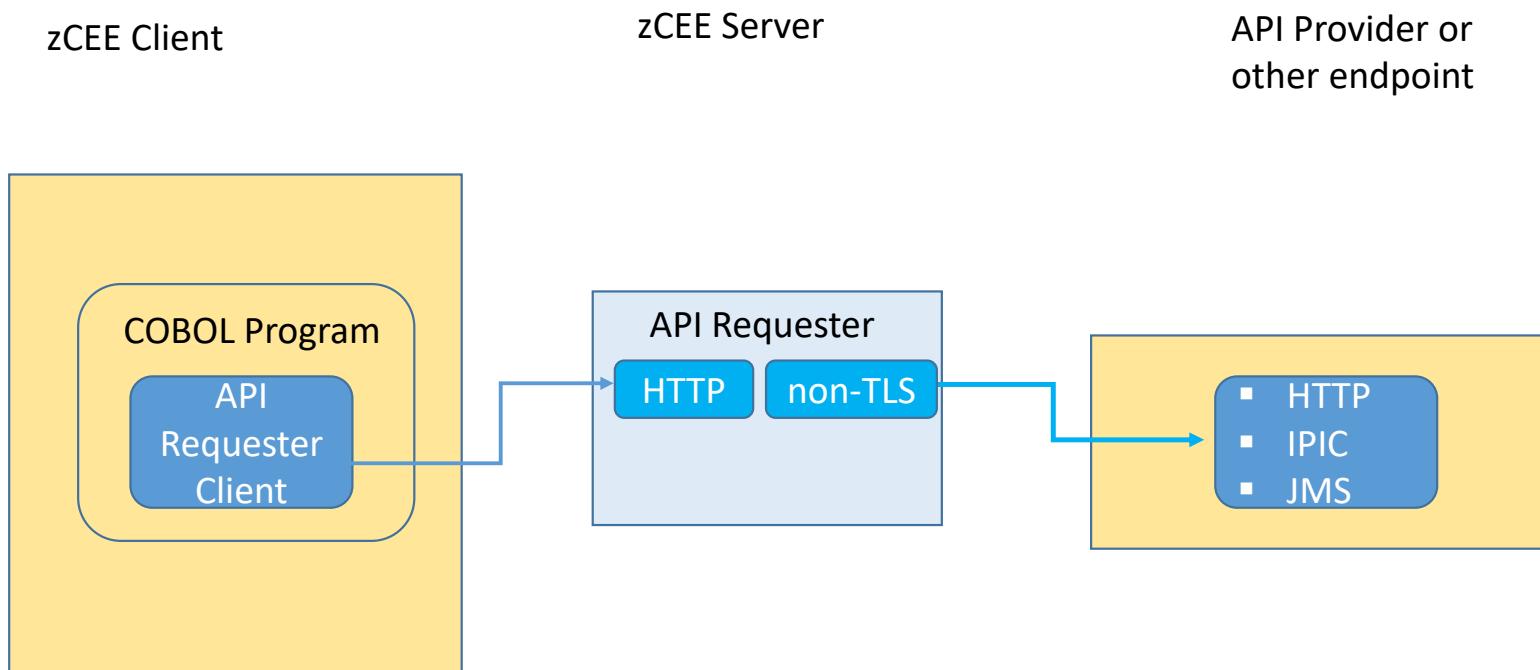


Outbound Policy acts a surrogate SSL client

## Non-TLS Flow



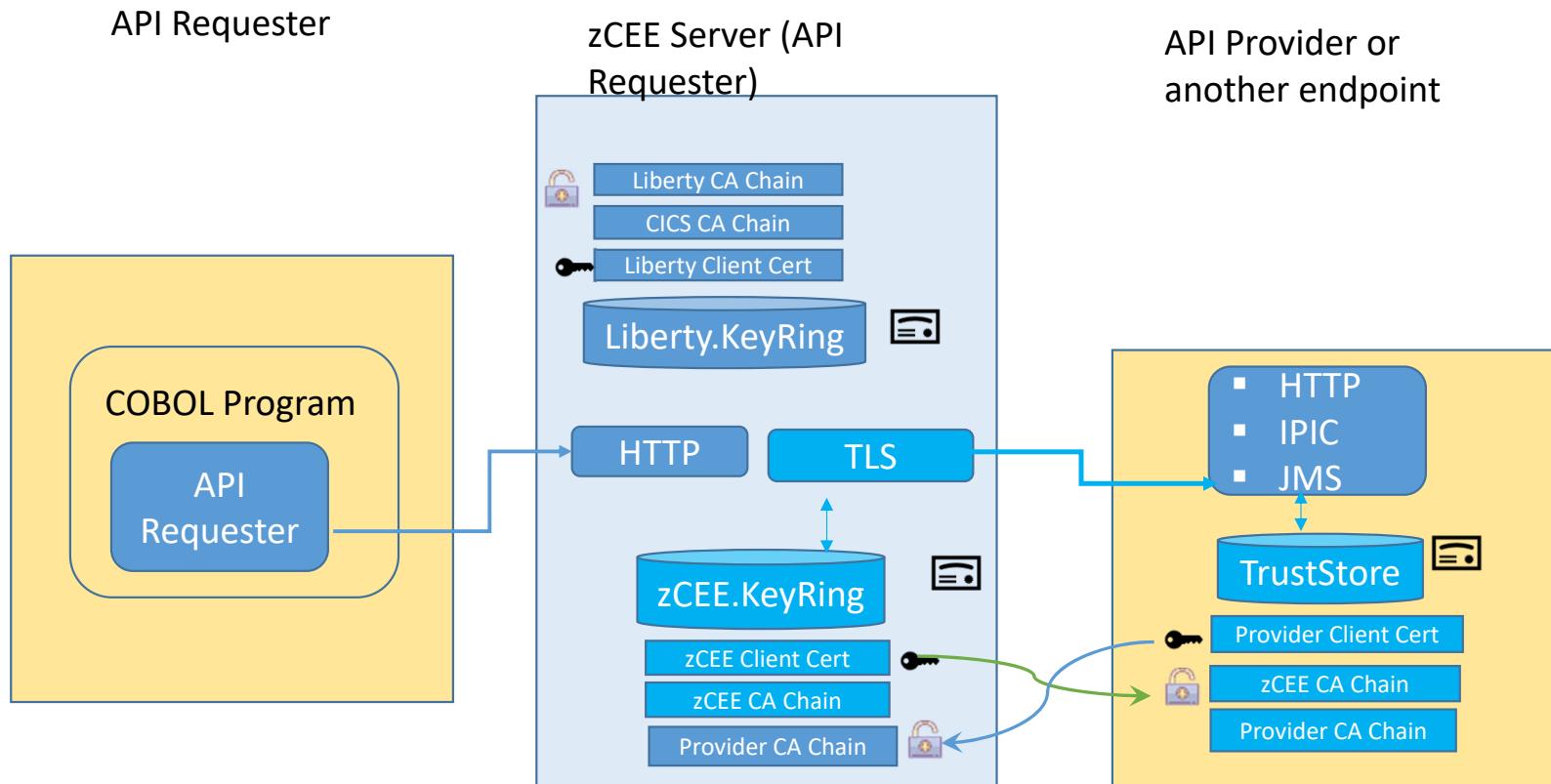
z/OS Connect EE



# Non-TLS Client with zCEE Outbound TLS Handshake



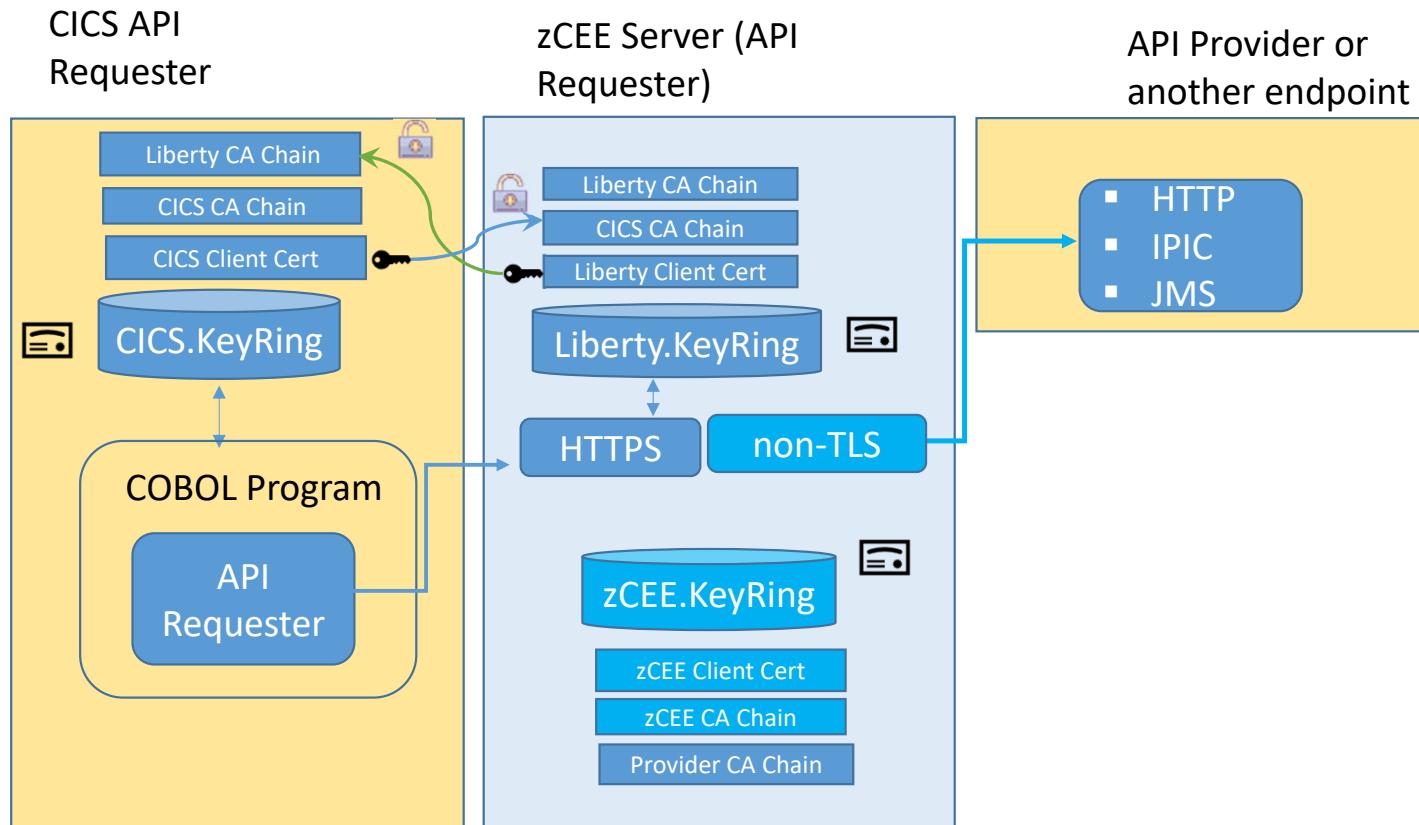
z/OS Connect EE



## TLS Client with zCEE non-TLS outbound



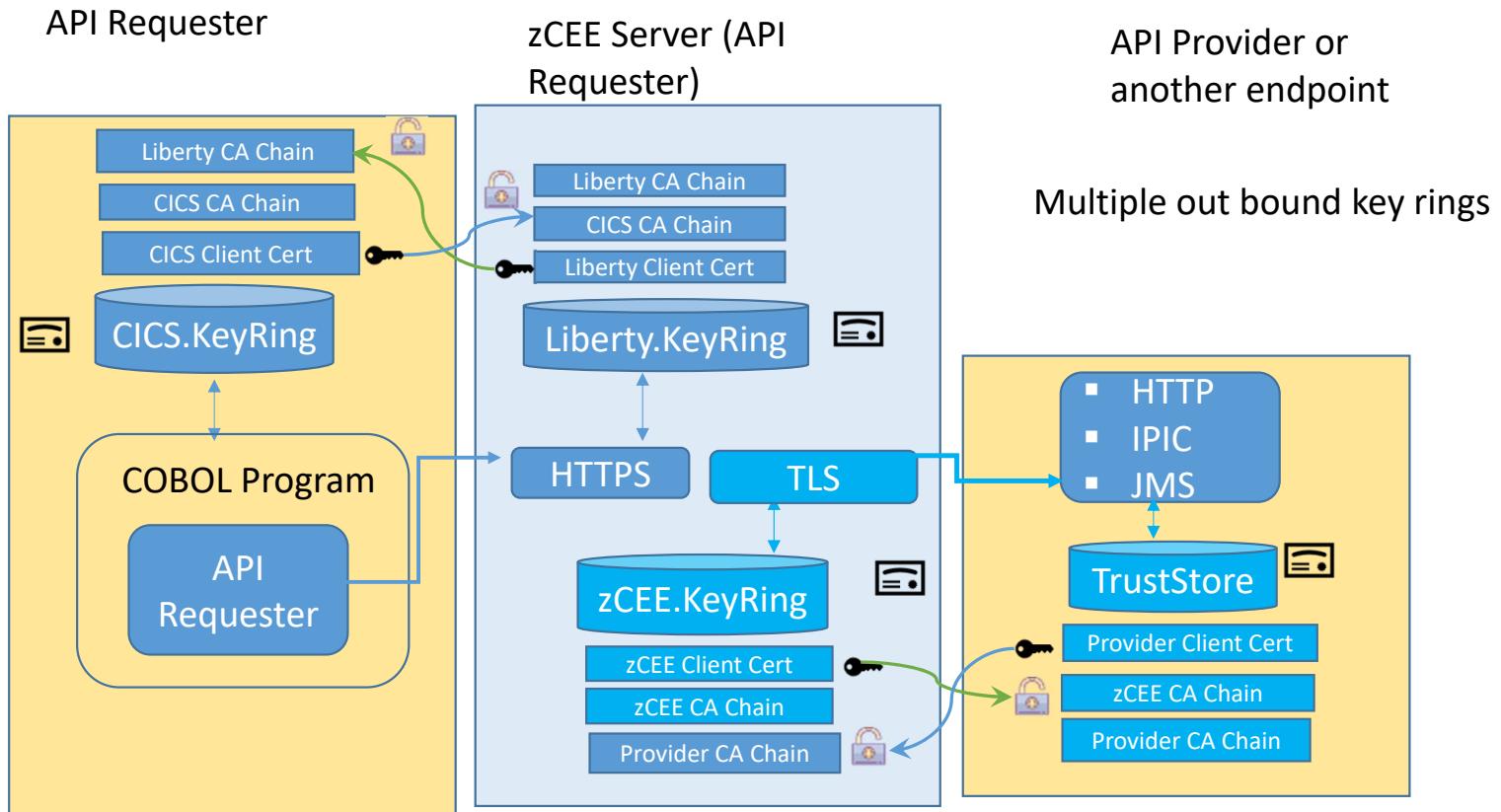
z/OS Connect EE



# TLS Client with zCEE inbound and outbound TLS Handshakes



z/OS Connect EE





## Creating a Key Ring

```
/* Import certificates obtained from a Certificate Authority
racdcert CERTAUTH withlabel('CICS CA') add('USER1.CICSCA.PEM')
racdcert id(cicsstc) withlabel('CICS Client Cert') add('USER1.CICS.P12') password('secret')
/* Create CICS key ring and connect CA and personal certificates */
racdcert id(cicsstc) addring(CICS.KeyRing)
racdcert id(cicsstc) connect(ring(CICS.KeyRing) label('CICS CA') certauth usage(certauth))
racdcert id(cicsstc) connect(ring(CICS.KeyRing) label('Liberty CA') certauth usage(certauth))
/* Connect default personal certificate */
racdcert id(cicsstc) connect(ring(CICS.KeyRing) label('CICS Client Cert') default
setropts raclist(digtring,digtnmap) refresh
```

### Broadcom Support web pages

Site of *What ACF2 security setup is needed for IBM's z/OS Connect Enterprise Edition V3.0?*

<https://knowledge.broadcom.com/external/article/128597/what-acf2-security-setup-is-needed-for-i.html>

Site of *ACF2 setup for z/OS Connect Enterprise Edition V3.0*

<https://knowledge.broadcom.com/external/article/142172/acf2-setup-for-zos-connect-enterprise-ed.html>

Site of *Setting up Liberty Server for z/OS with Top Secret*

<https://knowledge.broadcom.com/external/article/37272/setting-up-liberty-server-for-zos-with-t.html>



- During the TLS handshake, the TLS protocol and data exchange cipher are negotiated
- Choice of cipher and key length has an impact on performance
- You can restrict the protocol (SSL or TLS) and ciphers to be used
- Example setting server.xml file

```
<ssl id="DefaultSSLSettings" keyStoreRef="defaultKeyStore"  
sslProtocol="TLSv1.2" enabledCiphers="TLS_RSA_WITH_AES_256_CBC_SHA256  
TLS_RSA_WITH_AES_256_GCM_SHA384" />
```

- This configures use of TLS 1.2 and two supported ciphers
- It is recommended to control what ciphers can be used in the server rather than the client

## Persistent connections



z/OS Connect EE

- Persistent connections can be used to avoid too many handshakes
- Configured by setting the `keepAliveEnabled` attribute on the `httpOptions` element to **true**
- Example setting `server.xml` file

```
<httpEndpoint host="*" httpPort="80" httpsPort="443"  
id="defaultHttpEndpoint" httpOptionsRef="httpOpts" />  
  
<httpOptions id="httpOpts" keepAliveEnabled="true"  
maxKeepAliveRequests="500" persistTimeout="1m" />
```

- This sets the connection timeout to **1 minute** (default is 30 seconds) and sets the maximum number of persistent requests that are allowed on a single HTTP connection to **500**
- It is recommended to set a maximum number of persistent requests when connection workload balancing is configured
- It is also necessary to configure the client to support persistent connections



## SSL sessions

- When connections timeout, it is still possible to avoid the impact of full handshakes by reusing the SSL session id
- Configured by setting the `sslSessionTimeout` attribute on the `sslOptions` element to an amount of time
- Example setting `server.xml` file

```
<httpEndpoint host="*" httpPort="80" httpsPort="443"  
id="defaultHttpEndpoint" httpOptionsRef="httpOpts"  
sslOptionsRef="mySSLOptions"/>  
  
<httpOptions id="httpOpts" keepAliveEnabled="true"  
maxKeepAliveRequests="100" persistTimeout="1m"/>  
  
<sslOptions id="mySSLOptions" sslRef="DefaultSSLSettings"  
sslSessionTimeout="10m"/>
```

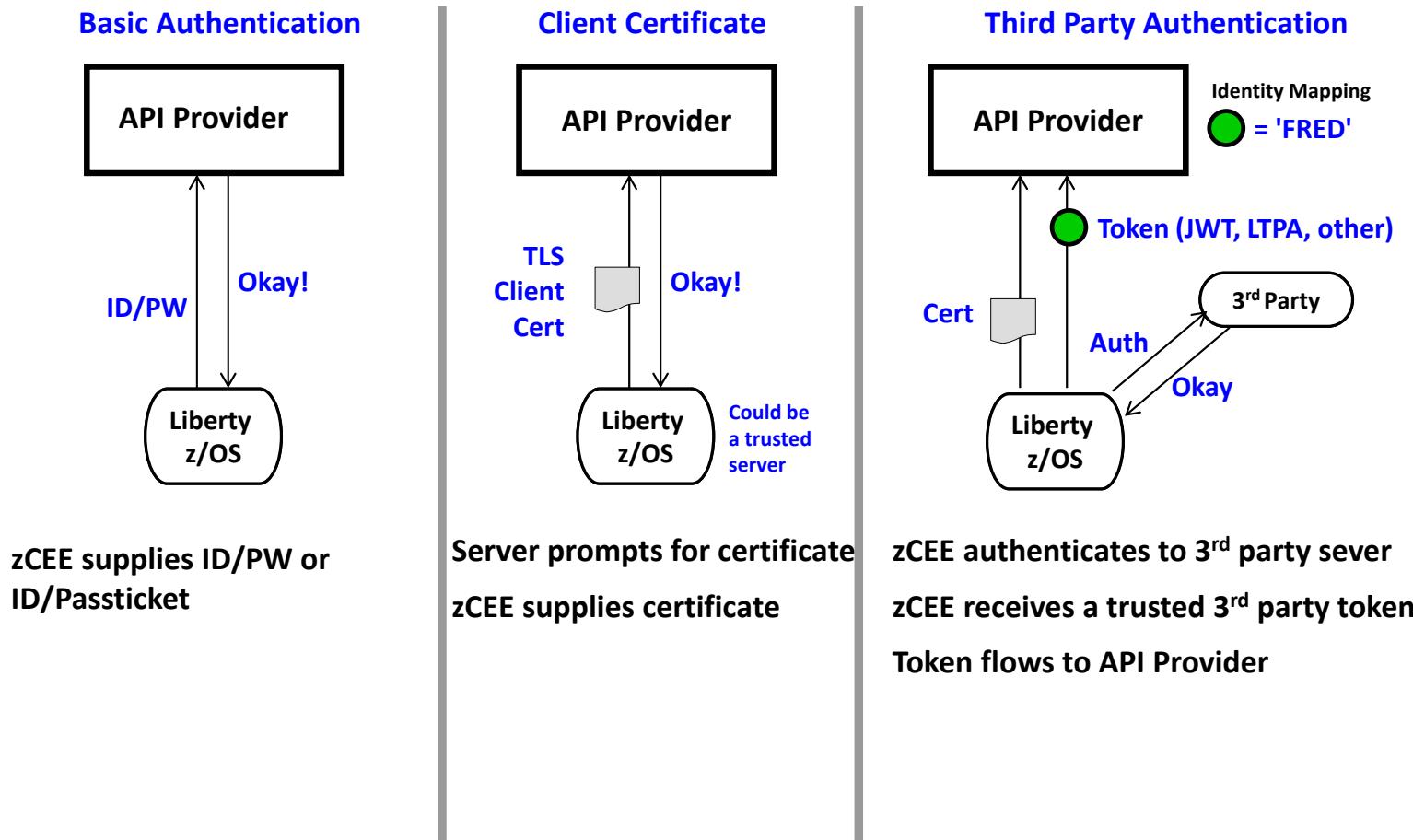
- This sets the timeout limit of an SSL session to **10 minutes** (default is 8640ms)
- SSL session ids are not shared across z/OS Connect servers

# API Provider Authentication



z/OS Connect EE

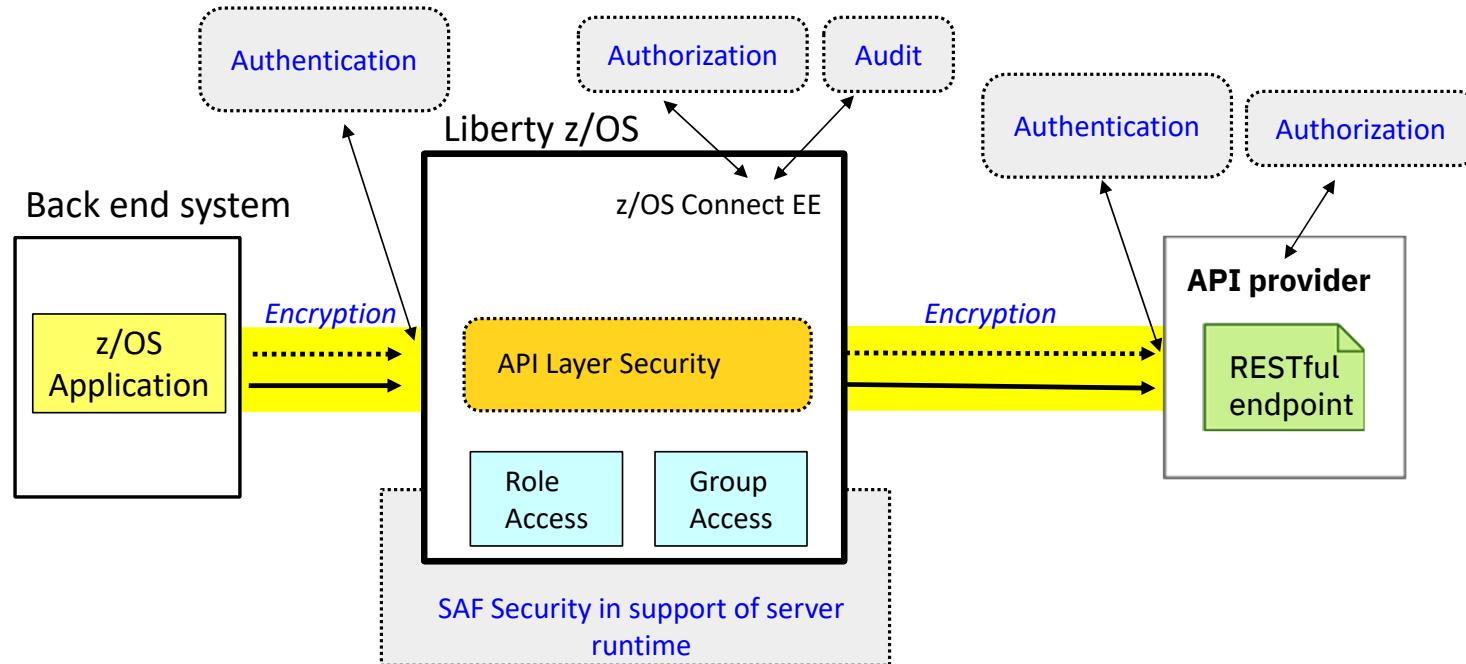
Several different ways this can be accomplished:



# API requester security – overview

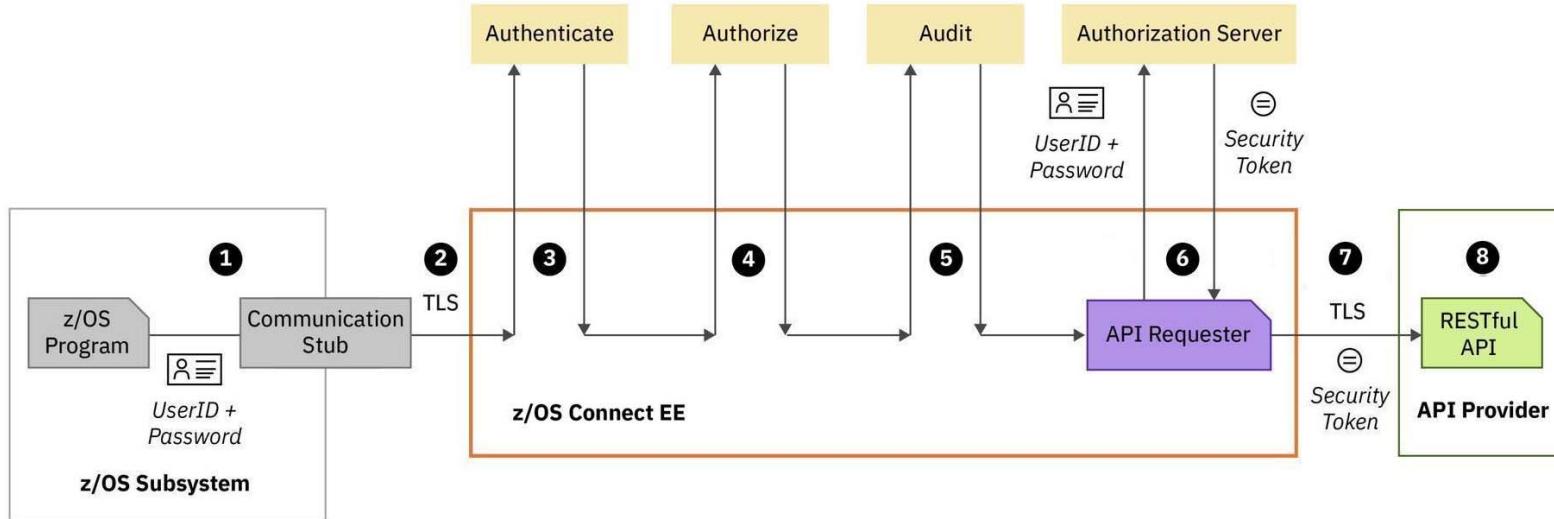


z/OS Connect EE



1. Authentication (basic, client certificate)
2. Encryption (aka "SSL" or "TLS")
3. Authorization (OAuth)
4. Audit
5. Configuring security with SAF

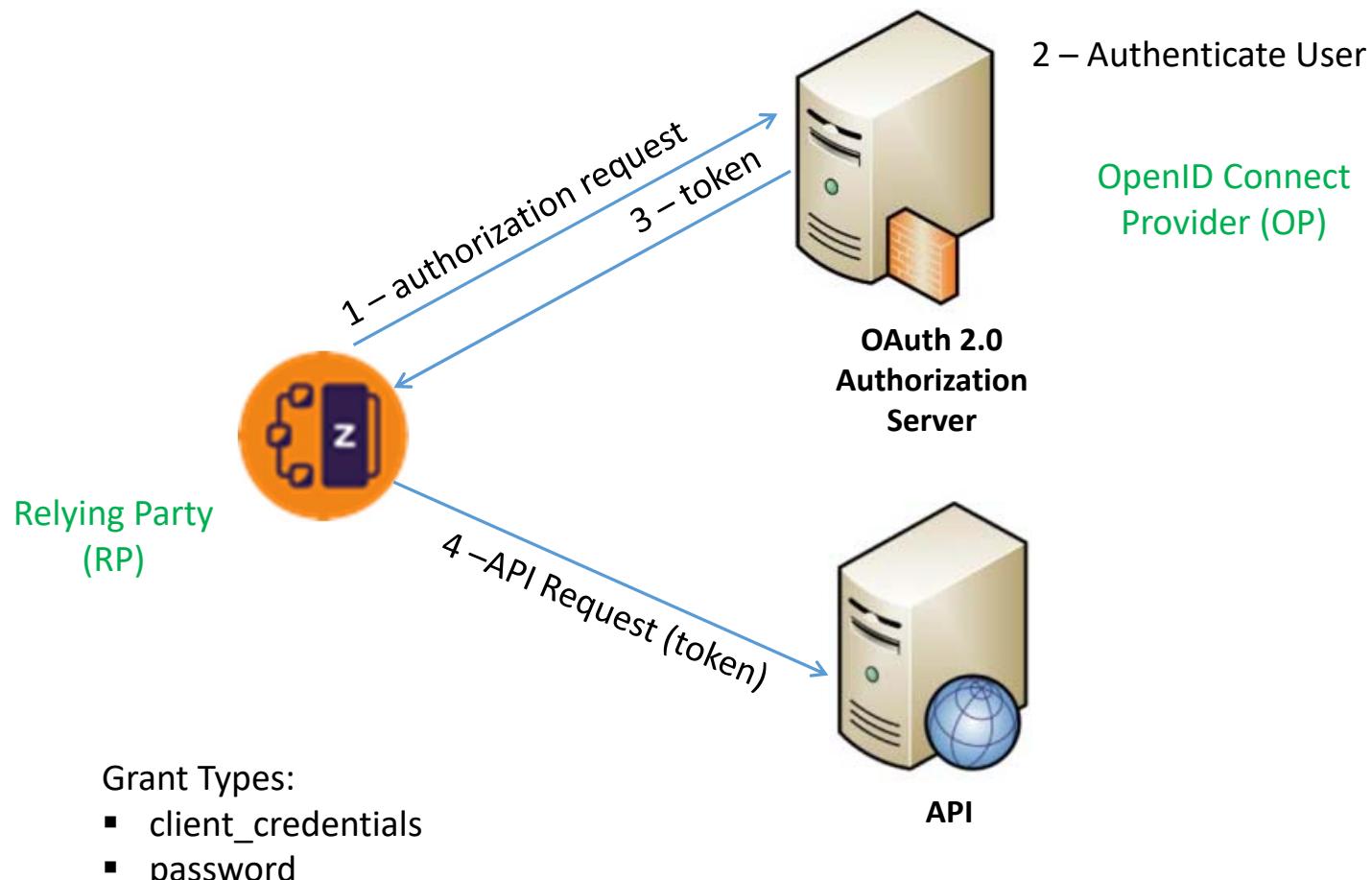
# Typical z/OS Connect EE security flow



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. Audit the API requester request
6. Pass the user ID and password credentials to an authorization server to obtain a security token.
7. 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**
8. The RESTful API runs in the external API provider



z/OS Connect EE



## Configuring OAuth support – BAQRINFO copy book



z/OS Connect EE

### 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.

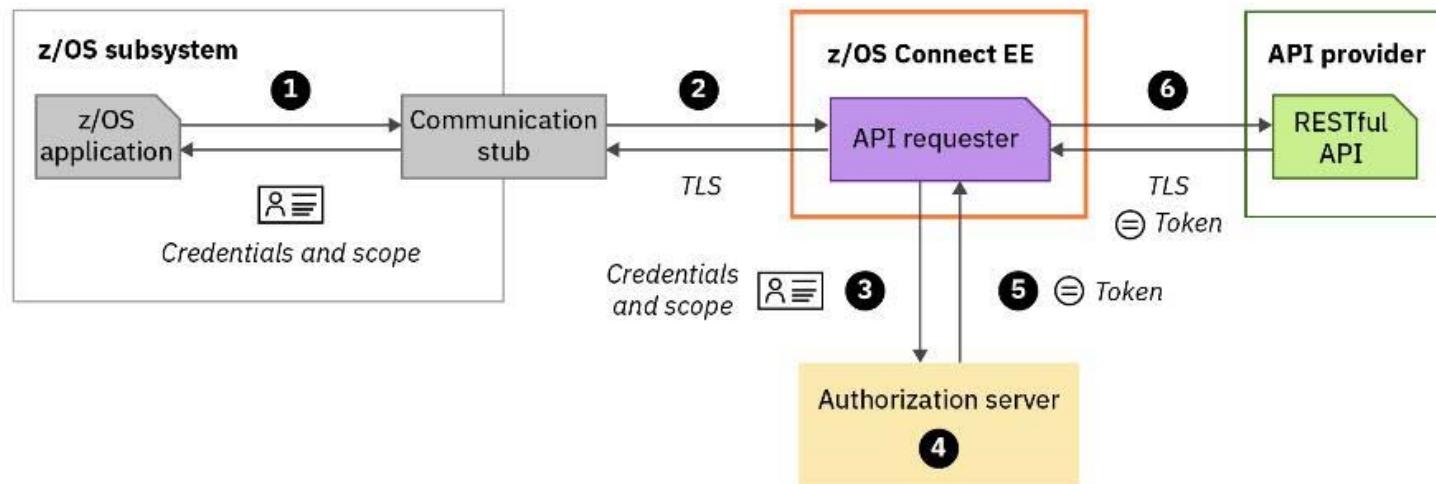
Grant Type: *client\_credentials* - the identity associated with the combination of the CICS, IMS, or z/OS application, and the z/OS Connect EE server that calls the RESTful API on behalf of the CICS, IMS, or z/OS application

Grant Type: *password* - The identity of the user of the CICS, IMS, or z/OS application, or it might be another entity.

## Calling an API with OAuth 2.0 support



z/OS Connect EE



```
MOVE user TO BAQ-OAUTH-USERNAME  
MOVE password TO BAQ-OAUTH-PASSWORD  
MOVE clientid TO BAQ-OAUTH-CLIENTID  
MOVE secret TO BAQ-OAUTH-CLIENT-SECRET
```



## Configuring OAuth support

For **OAuth**, two grant types are supported:

- Resource Owner Password Credential [a.k.a. password]
- Client Credentials [a.k.a. client credentials]

The access token is a way for the API provider to validate the client application rights to invoke its APIs.

```
<zosconnect_endpointConnection id="cscvincAPI"
    host="https://wg31.washington.ibm.com"
    port="26231"
    authenticationConfigRef="myOAuthConfig" />

<zosconnect_oAuthConfig id="myOAuthConfig"
    grantType= "password"
    authServerRef="myOAuthProvider" />

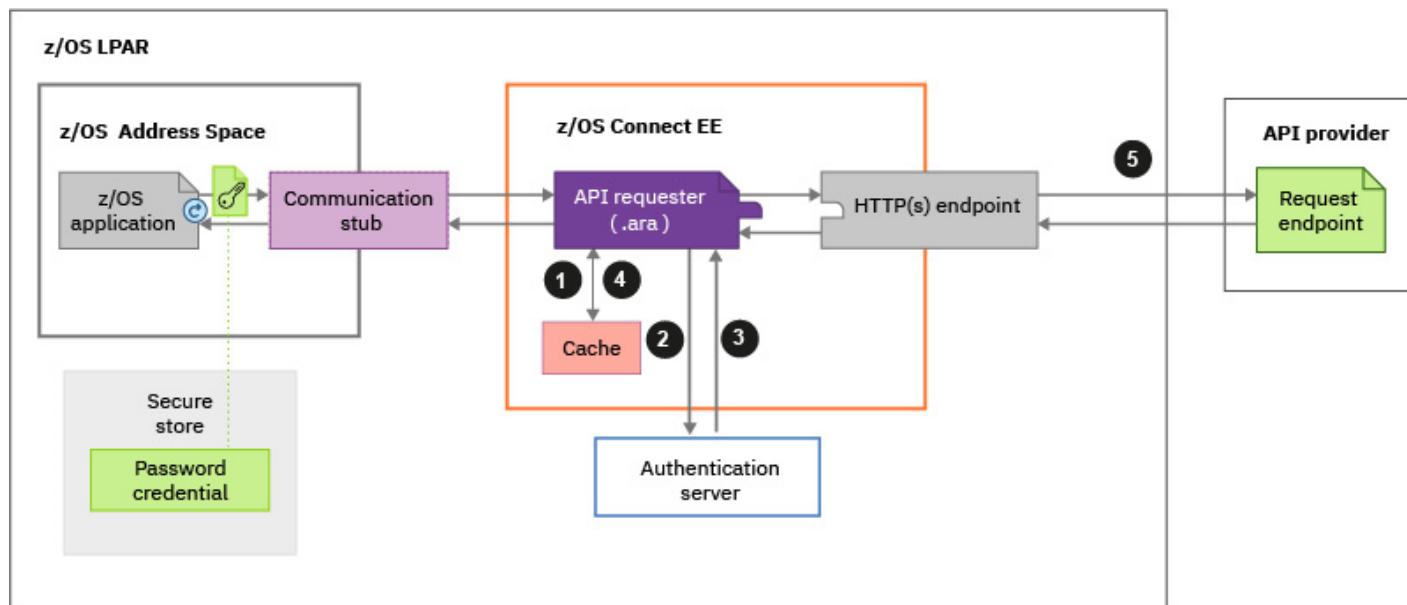
<zosconnect_authorizationServer id="myOAuthProvider"
    tokenEndpoint="https://wg31.washington.ibm.com:26213/oidc/endpoint/OP/token"
    basicAuthRef="tokenCredential"/> ← optional

<zosconnect_authData id="tokenCredential"
    user="rpSslToken" password="secret"/>
```

# Calling an API with Custom JWT support



z/OS Connect EE



MOVE user TO BAQ-TOKEN-USERNAME

MOVE password TO BAQ-TOKEN-PASSWORD

# Getting Started Guide



WP102724 - z/OS Connect EE Getting Started

**z/OS Connect Enterprise Edition V3.0**

**Getting Started Guide**  
for CICS, IMS, Db2 and MQ

Version Date: June 9, 2020



© IBM Corporation 2016, 2020

© 2016, 2020 IBM Corporation

Version Date: Tuesday, June 09, 2020

WP102724 - z/OS Connect EE V3 Getting Started.pdf - Adobe A...

File Edit View Window Help

Home Tools WP102724 - z/OS C... Sign In

F     3 / 160 50% ...

WP102724 - z/OS Connect EE Getting Started

Table of Contents

Document Overview  
Program numbers and FMIDs  
Service and Maintenance URLs  
The z/OS Connect EE Knowledge Center URL  
The z/OS Connect EE Support Center for z/OS Liberty Knowledge Center URL  
IBM developerWorks articles URL  
Additional IBM Support web pages

Installation and Initial Setup  
SMPIE Install of z/OS Connect EE  
Essential prerequisites  
OMVS Ownership/Permissions Bits Considerations  
User IDs and profiles  
Permit access to Unix Privileges  
Confirm that a Java runtime can be created  
Liberty angle Considerations  
Named angels  
Post-SMPIE configuration steps  
SAF Resources  
SAF Groups and Server IDs  
SAF STARTED profiles  
SAF SERVER and FACILITY profiles  
z/OS Connect EE server creation  
TCP ports and host name  
Start a z/OS Connect EE server  
Setup of basic security  
Installing the z/OS Connect EE V3.0 tooling  
Installing the z/OS Connect EE V3.0 API Toolkit in an Eclipse environment  
Installing the z/OS Connect EE V3.0 API Toolkit in an IBM Installation Manager environment  
Checkpoint: status at this point  
Open IBM z/OS Explorer for z/OS and connect to the z/OS Connect EE server

CICS RESTful APIs  
Adding IPIC support to a z/OS Connect server  
Setup of IPIC support in a CICS region  
Developing RESTful Services for CICS  
Test the Services  
Security and CICS

IMS TM RESTful APIs  
Adding IMS Connect support to a z/OS Connect server  
Install the IMS Phone Sample in the IMS control region  
Verify the IMS Service Provider  
Using Postman  
Using curl  
IMS connections (connections and interaction)  
Developing RESTful Services for an IMS transaction  
Test the Services  
Security and IMS TM

IMS DB RESTful APIs  
Adding IMS Database support to a z/OS Connect server  
A Review of the IMS artifacts

© 2016, 2020 IBM Corporation - 3 - Version Date: Monday, August 17, 2020

8.50 x 11.00 in < >

WP102724 - z/OS Connect EE V3 Getting Started.pdf - Adobe A...

File Edit View Window Help

Home Tools WP102724 - z/OS C... Sign In

H     4 / 160 50% ...

WP102724 - z/OS Connect EE Getting Started

Developing RESTful Services for an IMS database  
Test the Services  
Security and IMS DB

Db2 RESTful APIs  
Creating Db2 REST Services  
Adding Db2 REST support to a z/OS Connect server  
Developing RESTful Services for Db2 Native REST Services  
Db2 Native Procedure Considerations  
Test the Services  
Security and Db2

IBM MQ RESTful APIs  
Adding the IBM MQ Service provider support to a z/OS Connect server  
Adding JMS resources to the z/OS Connect EE configuration  
Developing RESTful Services for MQ  
Test the Services  
Security and MQ

Security Topics  
Encrypting the connection, security elements  
Turning off SSL and/or Authentication  
Using SAF for controlling z/OS Connect EE access  
Using RACF for TLS and trust/key store management  
Using client certificates for authentication  
RACF security filtering and Filtering  
CICS Identity Propagation  
IMS TM PassTickets  
IMS DB PassTickets  
Db2 PassTickets  
Db2 REST services security  
MQ services security  
MQ TLS security  
z/OS Connect and AT-TLS  
Troubleshooting RACF issues with Liberty and z/OS Connect servers  
Liberty Server Startup Errors  
Messages related to enabling RACF security  
Message about enabling digital certificates (TLS)  
WebSphere Optimized Local Adapter  
WOLA Security  
WOLA Error Messages

Miscellaneous Topics  
Testing z/OS Connect Services Using Postman  
Testing z/OS Connect Services Using curl  
Importing the z/OS Connect EE Policies  
Managing CORS updates  
Liberty Environment Variables  
Managing a z/OS Connect EE server with the Admin Center  
Alternatives to using CEEOPTS DD Input for API Requesters  
Controlling dynamic updates  
z/OS Connect and Data Virtualization Manager  
Sample JCL  
Base64 Encoding and Swagger UI

© 2016, 2020 IBM Corporation - 4 - Version Date: Monday, August 17, 2020

8.50 x 11.00 in < >

# Github Site



Screenshot of a GitHub repository page for "ibm-wsc/zCONNEE-Wildfire-Workshop".

The repository has 162 commits, 2 branches, 0 packages, 0 releases, and 1 contributor.

Branch: master

Recent activity:

- emitchj Add files via upload (Latest commit dd35210 2 days ago)
- emitchj Add files via upload (Latest commit 4b21589 7 days ago)
- emitchj Add files via upload (7 days ago)
- emitchj Add files via upload (6 months ago)
- emitchj Add files via upload (2 months ago)
- emitchj Add files via upload (2 months ago)
- emitchj Add files via upload (2 months ago)
- emitchj Add files via upload (2 months ago)
- emitchj Add files via upload (2 months ago)
- emitchj Add files via upload (2 months ago)
- emitchj Add files via upload (2 months ago)
- emitchj Add files via upload (last month)
- emitchj Add files via upload (7 days ago)
- emitchj Add files via upload (2 months ago)
- emitchj Add files via upload (last month)

File list:

- Misc Presentations
- cobol
- exercises
- security
- Introduction to zOS Connect EE .pdf
- README.md
- WP102724 - zOS Connect EE V3 Getting Started.pdf
- WSC Wildfire zOS Primer.pdf
- README.md

This repository contains material from the z/OS Connect EE Wildfire workshops run by the IBM Washington Systems Center.

Collateral related to the Washington System Center z/OS Connect Wildfire Workshop

Manage topics

Code Issues Pull requests Actions Projects Wiki Security Insights Settings

Create new file Upload files Find file Clone or download

Branch: master

zCONNEE-Wildfire-Workshop / exercises /

emitchj Add files via upload (Latest commit 4b21589 7 days ago)

Developing Outbound APIs Requesters Applications.pdf  
Developing RESTful APIs for a CICS COMMAREA program.pdf  
Developing RESTful APIs for DVM 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.pdf  
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 program.pdf  
zCEE Customization Basic Configuration.pdf  
zCEE Customization Basic Security.pdf

ibm-wsc / zCONNEE-Wildfire-Workshop

Code Issues Pull requests Actions Projects Wiki Security Insights Settings

Branch: master

zCONNEE-Wildfire-Workshop / security /

emitchj Add files via upload (Latest commit dd35210 2 days ago)

zCEE Customization Basic Configuration.pdf  
zCEE Customization Basic Security.pdf  
zCEE Customization Security and CICS.pdf  
zCEE Customization Security and DB2.pdf  
zCEE Customization Security when accessing an IMS Database.pdf  
zCEE Customization Security when accessing an IMS Transaction.pdf  
zCEE Customization Security with MVS Batch.pdf

- <http://tinyurl.com/y28fsezs>



## **/questions?thanks=true**

Thank you for listening.

- z/OS Connect EE Users Group: <https://www.linkedin.com/groups/8731382/>