



An introduction to z/OS Connect API Requesters

Mitch Johnson

mitchj@us.ibm.com

Washington System Center



Notes and Disclaimers



- The information in this presentation was derived from various product documentation web sites.
- There will be additional information on slides that will be designated as **Tech/Tips**. These contain information that will be useful to the reader.
- A z/OS Connect OpenAPI 2, or a z/OS Connect OpenAPI 3 icon will appear on slides where the information is specific to these products. If the topic applies both OpenAPI2 and OpenAPI3, both icons will appear. Don't hesitate to ask questions as to why the icon does or does not appear on certain slides.
- The examples, tips, etc. present in this material are based on firsthand experiences and are not necessarily sanctioned by Liberty or z/OS Connect product areas.

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



ZCINTRO - IBM z/OS Connect An introduction and overview

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

mitchj@us.ibm.com



z/OS Connect Open API 3

Designer and z/OS Native server
Experiences and Observations

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

mitchj@us.ibm.com



ZCADMIN – IBM z/OS Connect Administration

WebSphere Liberty Profile with
(OpenAPI 2) and/or
React (OpenAPI 3)
Registration



© 2017, 2022 IBM Corporation
Slide 1



© 2022 IBM Corporation
Slide 1

mitchj@us.ibm.com

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

© 2018, 2023 IBM Corporation

Page 3

z/OS Connect Github Site

<https://ibm.biz/zCEEWorkshopMaterial>



The screenshot displays three GitHub repository pages:

- Top Repository:** [ibm-wsc/zCONNEE-Wildfire-Workshop](https://github.com/ibm-wsc/zCONNEE-Wildfire-Workshop) (Public). The master branch shows several PDF files under the OpenAPI2 directory, all uploaded by user emitchj. A red oval highlights these files.
- Middle Repository:** [ibm-wsc/zCONNEE-Wildfire-Workshop](https://github.com/ibm-wsc/zCONNEE-Wildfire-Workshop) (Public). The master branch shows several PDF files under the APIRequesters directory, all uploaded by user emitchj. A red oval highlights these files.
- Bottom Repository:** [ibm-wsc/zCONNEE-Wildfire-Workshop](https://github.com/ibm-wsc/zCONNEE-Wildfire-Workshop) (Public). The master branch shows several PDF files under the OpenAPI2 directory, all uploaded by user emitchj. A red oval highlights these files.

Common File List:

- Developing CICS API Requester Applications.pdf
- Developing IMS API Requester Applications.pdf
- Developing MVS Batch API Requester Applications.pdf

Bottom Repository Description:

This repository contains material from the z/OS Connect EE Wildfire workshops run by the I Center. It is should be referenced frequently for updates to the presentations, exercises, sam

mitchj@us.ibm.com

- Contact your IBM representative to schedule access to these exercises

© 2018, 2023 IBM Corporation

Page 4

Agenda

- What is REST and what are REST APIs?
- Using an z/OS Connect API requester to access a REST API
- General API requester COBOL client programming considerations
- Developing API requesters for OpenAPI 2 REST APIs
- Developing API requesters for OpenAPI 3 REST APIs
- Configuration and Security considerations for API requesters



/what_is_REST_and_what_are_REST_API?



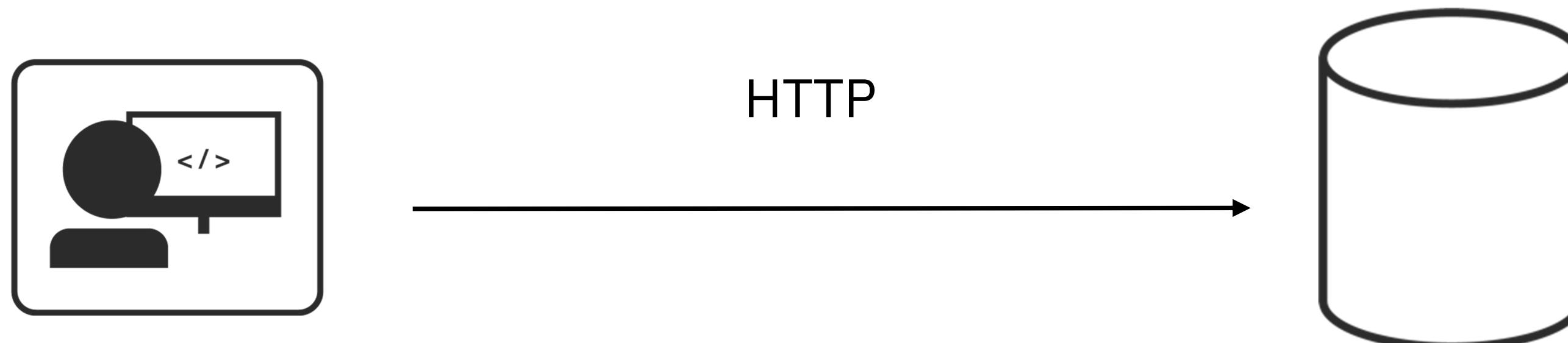
REST is architectural programming style

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

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

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

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



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



The key principles of REST

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

POST
GET
PUT
DELETE

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

Use Path and Query parameters to refine the request

URI path identifies a resource (or lists of resources)

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

Request/Response Body is used to represent the data object

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



REST vs RESTful

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

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

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

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



RESTful Examples

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

GET /account?number=1234

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

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

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

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

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

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



Not every REST API is a RESTful API

(How to know if an API is not RESTful)

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

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

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

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

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

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

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

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

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



Why are REST APIs popular?

Ubiquitous Foundation

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

Relatively Lightweight

Compared to other technologies (for example, SOAP/WSDL), the REST/JSON pattern is relatively light protocol and data model, which maps well to resource-limited devices.

Relatively Easy Development

Since the REST interface is so simple, developing the client involves very few things: an understanding of the URI requirements (path, parameters) and any JSON data schema.

Increasingly Common

REST/JSON is becoming more and more a de facto "standard" for exposing APIs and Microservices. As more adopt the integration pattern, the more others become interested.

Stateless

REST is by definition a stateless protocol, which implies greater simplicity in topology design. There's no need to maintain, replicate or route based on state.



What is the goal? To have client code that is independent of the infrastructure

```
ZceeCICSGet.java
55
56
57
58
59
60
61
62
63
64
65
66
67
68
// Invoke the REST API using a GET method
URL url = new URL("https://wg31.washington.ibm.com:9453/cscvinc/employee/" + args[1]);
System.out.println("URL: " + url);
HttpsURLConnection conn = (HttpsURLConnection) url.openConnection();
conn.setRequestMethod("GET");
conn.setRequestProperty("Content-Type", "application/json");
byte[] bytesEncoded = Base64.encodeBase64(args[0].getBytes());
conn.addRequestProperty("Authorization", "Basic " + new String(bytesEncoded));
try {
    if (conn.getResponseCode() != 200) {
        throw new RuntimeException("Failed : HTTP error code : " + conn.getResponseCode());
    }
    BufferedReader bufferReader = new BufferedReader(new InputStreamReader((conn.getInputStream())));
    while ((output = bufferReader.readLine()) != null) {
        System.out.println(output);
    }
}
```

CICS

```
ZceeDb2Get.java
52
53
54
55
56
57
58
59
60
61
62
63
64
65
// Invoke the REST API using a GET method
URL url = new URL("https://wg31.washington.ibm.com:9453/db2/employee/" + args[1]);
System.out.println("URL: " + url);
HttpsURLConnection conn = (HttpsURLConnection) url.openConnection();
conn.setRequestMethod("GET");
conn.setRequestProperty("Content-Type", "application/json");
byte[] bytesEncoded = Base64.encodeBase64(args[0].getBytes());
conn.addRequestProperty("Authorization", "Basic " + new String(bytesEncoded));
try {
    if (conn.getResponseCode() != 200) {
        throw new RuntimeException("Failed : HTTP error code : " + conn.getResponseCode());
    }
    BufferedReader bufferReader = new BufferedReader(new InputStreamReader((conn.getInputStream())));
    while ((output = bufferReader.readLine()) != null) {
        System.out.println(output);
    }
}
```

Db2

```
ZceeMqGet.java
54
55
56
57
58
59
60
61
62
63
64
65
// Invoke the REST API using a GET method
URL url = new URL("https://wg31.washington.ibm.com:9453/mqapi/");
System.out.println("URL: " + url);
HttpsURLConnection conn = (HttpsURLConnection) url.openConnection();
conn.setRequestMethod("GET");
conn.setRequestProperty("Content-Type", "application/json");
byte[] bytesEncoded = Base64.encodeBase64(args[0].getBytes());
conn.addRequestProperty("Authorization", "Basic " + new String(bytesEncoded));
try {
    if (conn.getResponseCode() != 200) {
        throw new RuntimeException("Failed : HTTP error code : " + conn.getResponseCode());
    }
    BufferedReader bufferReader = new BufferedReader(new InputStreamReader((conn.getInputStream())));
    while ((output = bufferReader.readLine()) != null) {
        System.out.println(output);
    }
}
```

MQ

```
ZceelMSGet.java
53
54
55
56
57
58
59
60
61
URL url = new URL("https://wg31.washington.ibm.com:9453/phonebook/contacts/" + args[1]);
System.out.println("URL: " + url);
HttpURLConnection conn = (HttpURLConnection) url.openConnection();
conn.setRequestMethod("GET");
conn.setRequestProperty("Content-Type", "application/json");
byte[] bytesEncoded = Base64.encodeBase64(args[0].getBytes());
conn.addRequestProperty("Authorization", "Basic " + new String(bytesEncoded));
try {
    if (conn.getResponseCode() != 200) {
        throw new RuntimeException("Failed : HTTP error code : " + conn.getResponseCode());
    }
    BufferedReader bufferReader = new BufferedReader(new InputStreamReader((conn.getInputStream())));
    while ((output = bufferReader.readLine()) != null) {
        System.out.println(output);
    }
}
```

IMS



Goal: To have client code that is unaware of the z/OS infrastructure

In these examples, these Java clients have the same application programming pattern.

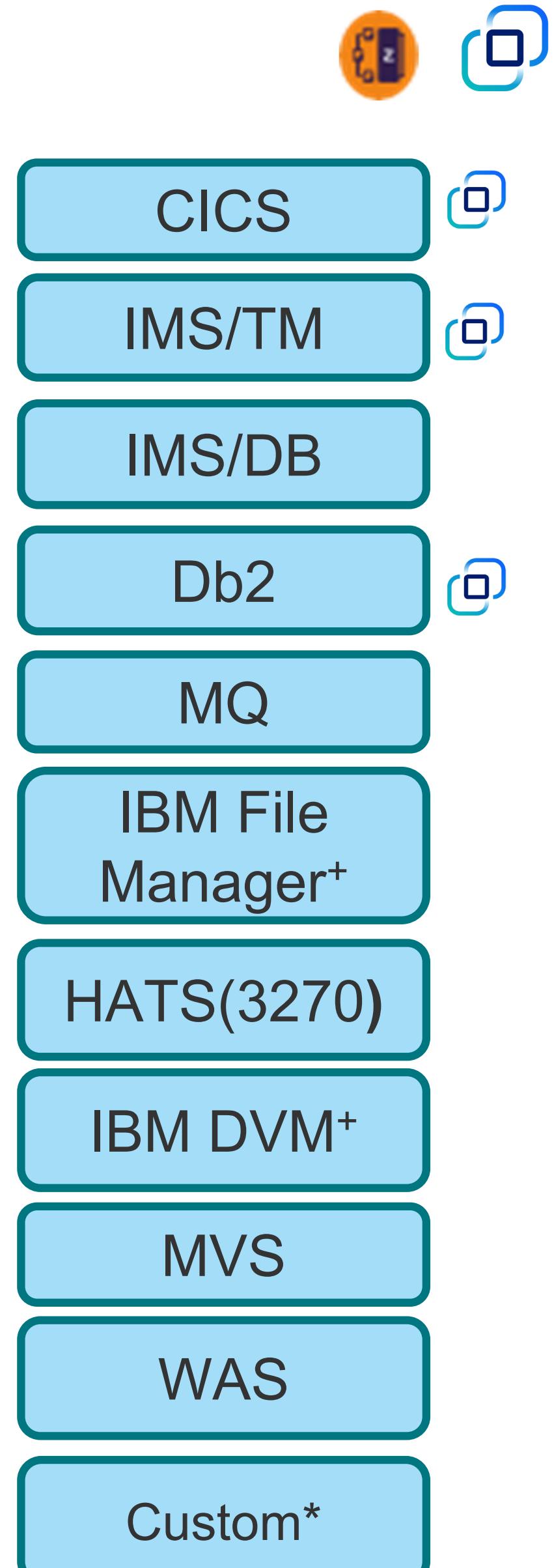
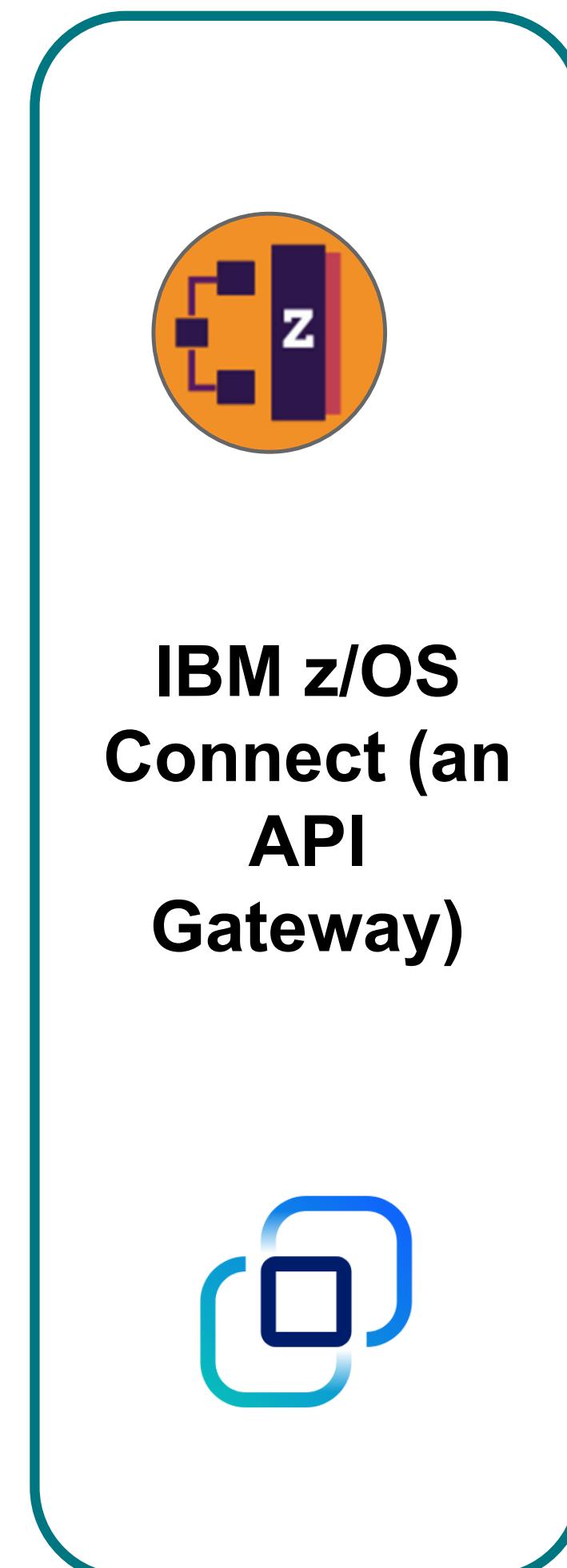
- They provide an URL, specify a method and provide a request message.
- Then use HTTP protocol to send a request to the API provider.
- A response message is returned after accessing a remote resource, e.g., a CICS program, a Db2 table, an IMS transaction or a MQ queue. The client application is unaware of the underlying infrastructure. No dependencies on coding for ECI, JDBC, OTMA, JMS, J2C, etc.

```
61 conn.addRequestProperty("Authorization", new String(bytesEncoded));
62 try {
63     if (conn.getResponseCode() != 200) {
64         throw new RuntimeException("Failed : HTTP error code : "
65             + conn.getResponseCode());
66     }
67 }
```

```
60 conn.addRequestProperty("Authorization", "Basic " + new String(bytesEncoded));
61 }
```

z/OS Connect API requester support provides the same pattern for COBOL applications running on z/OS. The fundamentals of this pattern is encapsulated with z/OS Connect therefore there is a minimum need for new COBOL code.

Initially, z/OS Connect shipped *API provider* support for exposing z/OS resources to client applications in the “cloud” via RESTful APIs

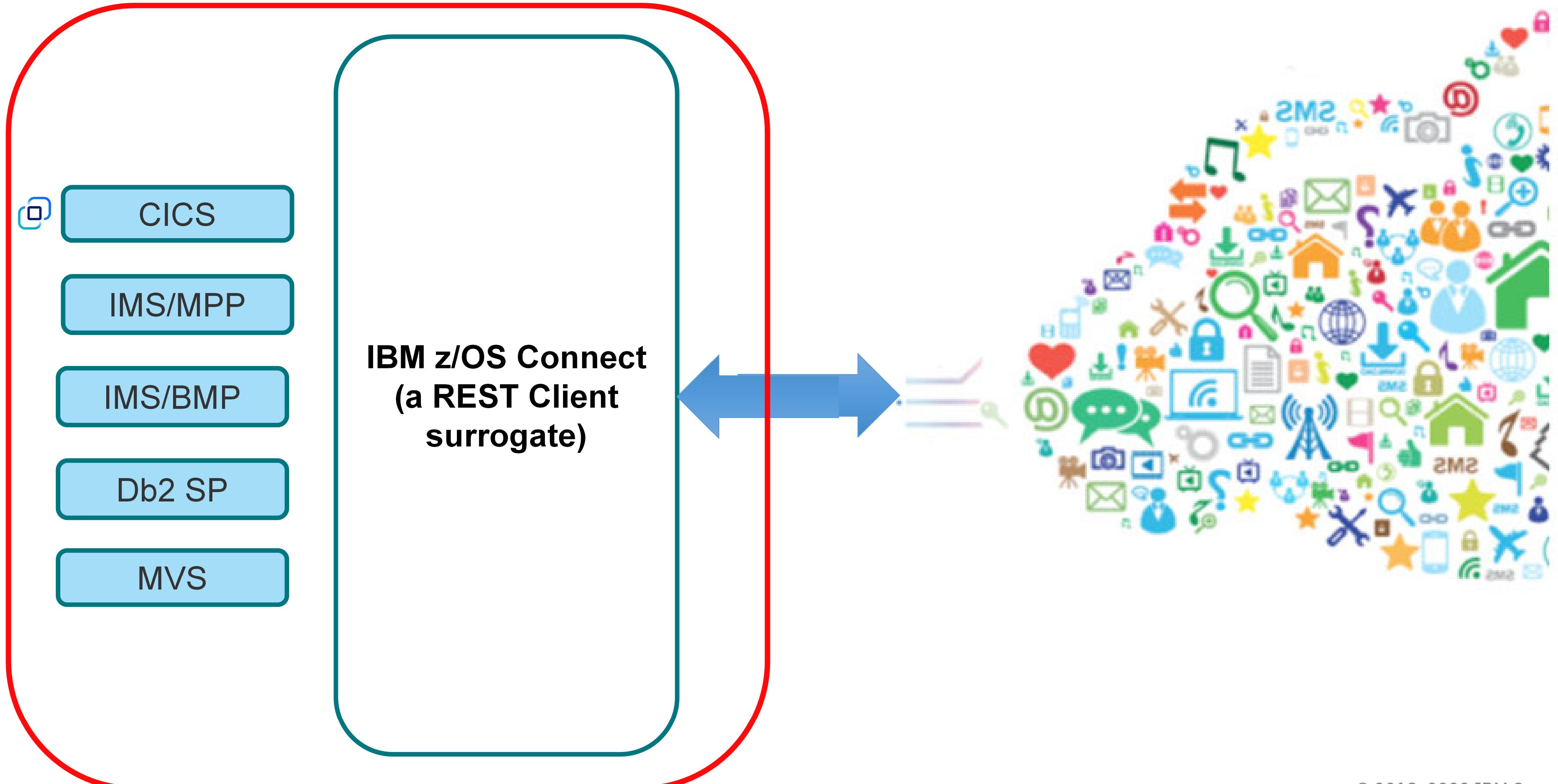


+ HCL and Rocket Software

*Other Vendors or your own implementation



Later, z/OS Connect added *API requester* support for providing access to REST APIs in the “cloud” to z/OS COBOL client applications





Let's start at the beginning, how are REST API described?

By using an OpenAPI specification document

The industry standard framework for describing REST APIs

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



Open API Initiative provides more than just an API framework

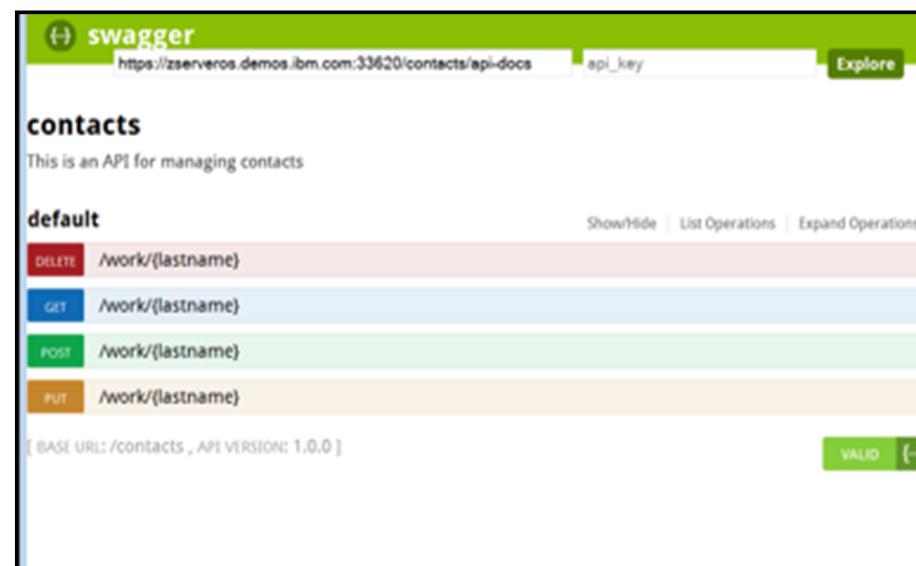


There are a number of tools available to aid consumption:

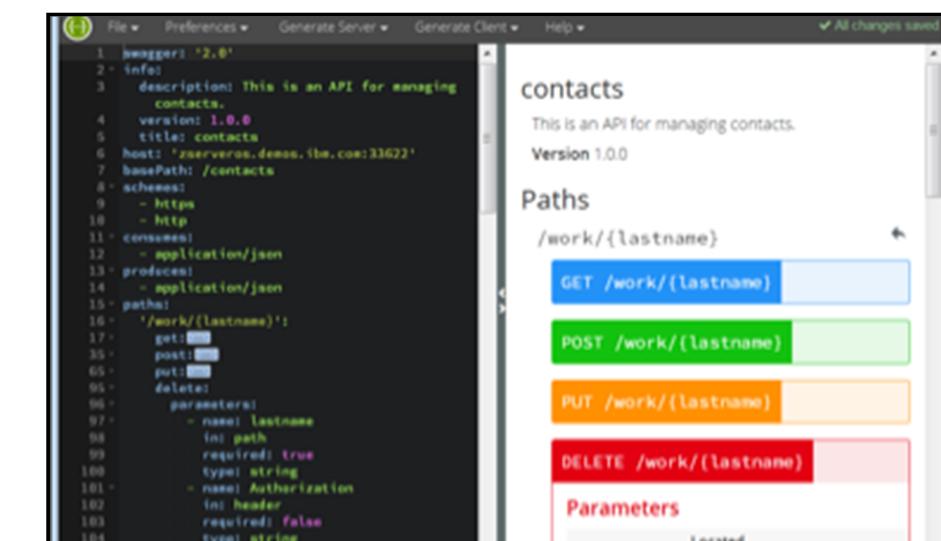
Code Generation* - create stub code to consume APIs from various languages



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



Editors - allows API developers to design their OpenAPI documents.



* z/OS Connect API Requester

+z/OS Connect, MQ REST support, Zowe

Important - You may have used or heard of the term Swagger with the use of APIs. As the use of APIs has grown this term has become in some respects misleading. To be more precise, OpenAPI refers to the API specifications (OpenAPI 2 and OpenAPI 3) where Swagger refers to the tooling used to implement the specifications.

Tech-Tip: An OpenAPI 2 specification example



The image shows two side-by-side JSON editors displaying an OpenAPI 2 specification for a miniloan service. The left editor shows the Request message, and the right editor shows the Response message layout.

Request message (Left Editor):

- Base URI Path:** Circled in red, showing the basePath: "/miniloan".
- URI Path extension/method:** Circled in red, showing the path: /loan: post.
- Request message:** Circled in red, showing the schema: "#/definitions/postMiniloanService_request".

Response message layout (Right Editor):

- Response message fields:** A large red circle highlights the properties section of the response schema, which includes fields like MINILOAN_COMMAREA, creditScore, yearlyIncome, age, amount, effectiveDate, and yearlyRepayment.
- Response message layout:** Circled in red, showing the schema: "#/definitions/postMiniloanService_response_200".

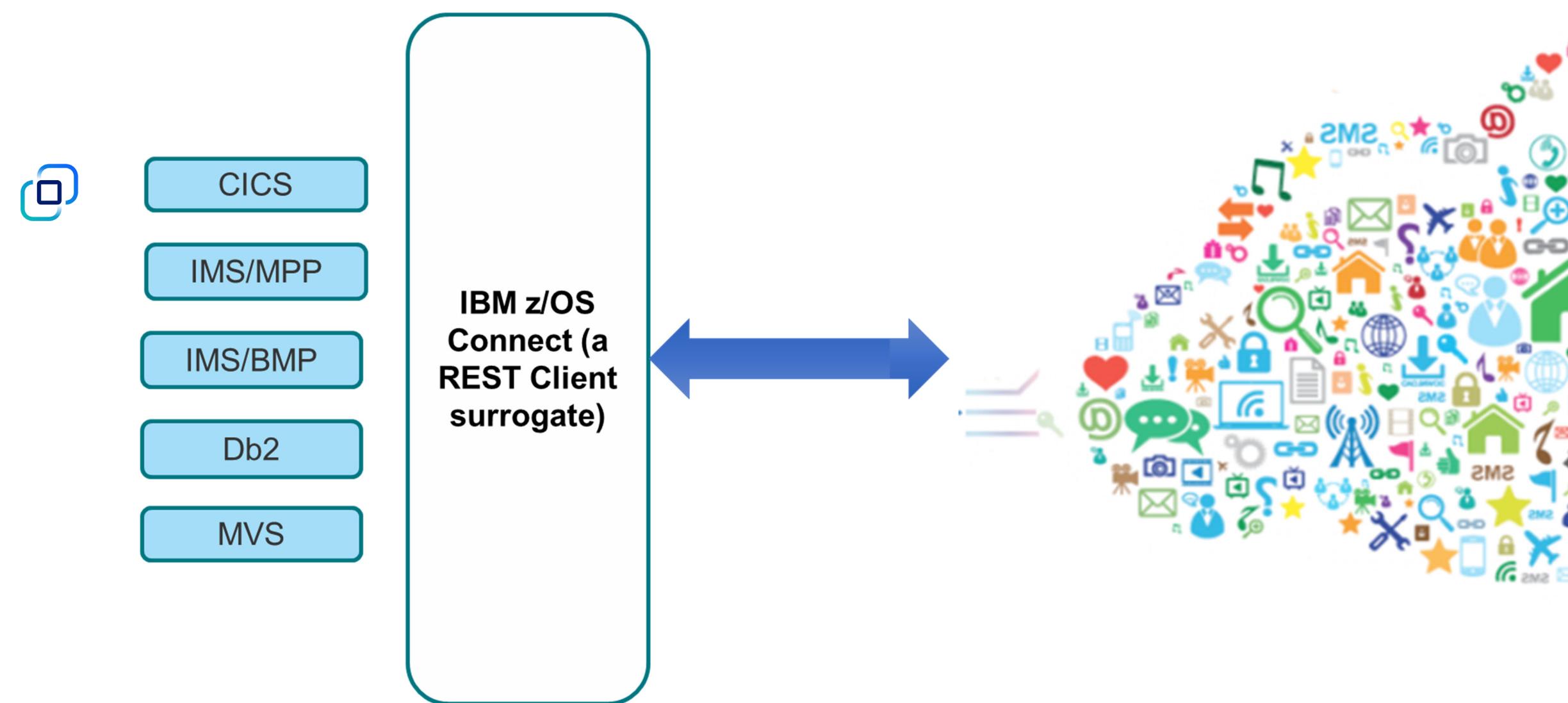
```
swagger: "2.0"
info:
  description: ""
  version: "1.0"
  title: "miniloan"
host: "localhost:8080"
basePath: "/miniloan"
schemes:
  0: "https"
  1: "http"
consumes:
  0: "application/json"
produces:
  0: "application/json"
paths:
  /loan:
    post:
      tags:
        0:
          name: "miniloan"
          operationId: "postMiniloanService"
      parameters:
        0:
          name: "Authorization"
          in: "header"
          required: false
          type: "string"
        1:
          in: "body"
          name: "postMiniloanService_request"
          description: "request body"
          required: true
          schema:
            $ref: "#/definitions/postMiniloanService_request"
      responses:
        200:
          description: "OK"
          schema:
```

```
schema:
  $ref: "#/definitions/postMiniloanService_response_200"
definitions:
  postMiniloanService_request:
    type: "object"
    properties:
      MINILOAN_COMMAREA:
        type: "object"
        properties:
          name:
            type: "string"
            maxLength: 20
          creditScore:
            type: "integer"
            minimum: 0
            maximum: 10000000000000000000
          yearlyIncome:
            type: "integer"
            minimum: 0
            maximum: 10000000000000000000
          age:
            type: "integer"
            minimum: 0
            maximum: 9999999999
          amount:
            type: "integer"
            minimum: 0
            maximum: 10000000000000000000
          effectiveDate:
            type: "string"
            maxLength: 8
          yearlyRepayment:
            type: "integer"
            minimum: 0
            maximum: 10000000000000000000
    postMiniloanService_response_200:
      type: "object"
```

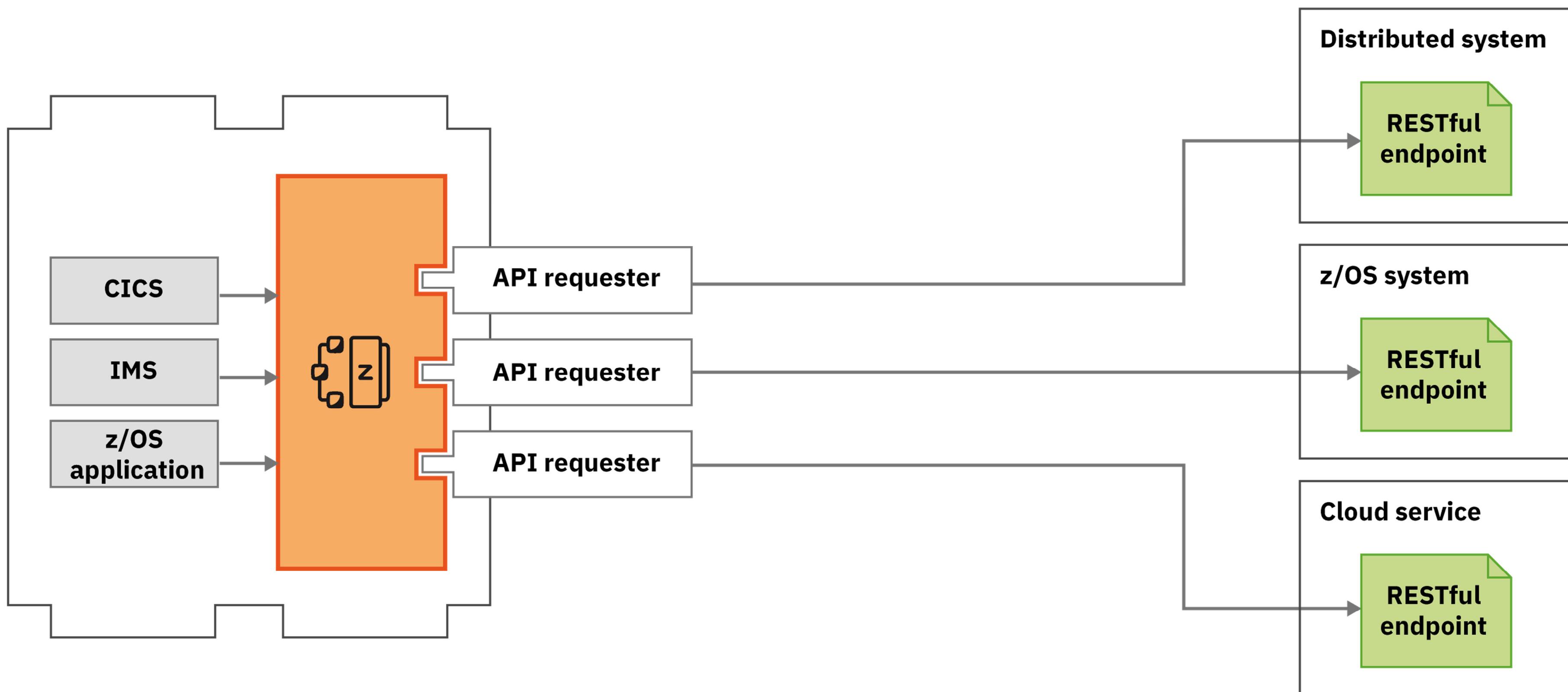


Using an z/OS Connect API requester to access a REST APIs

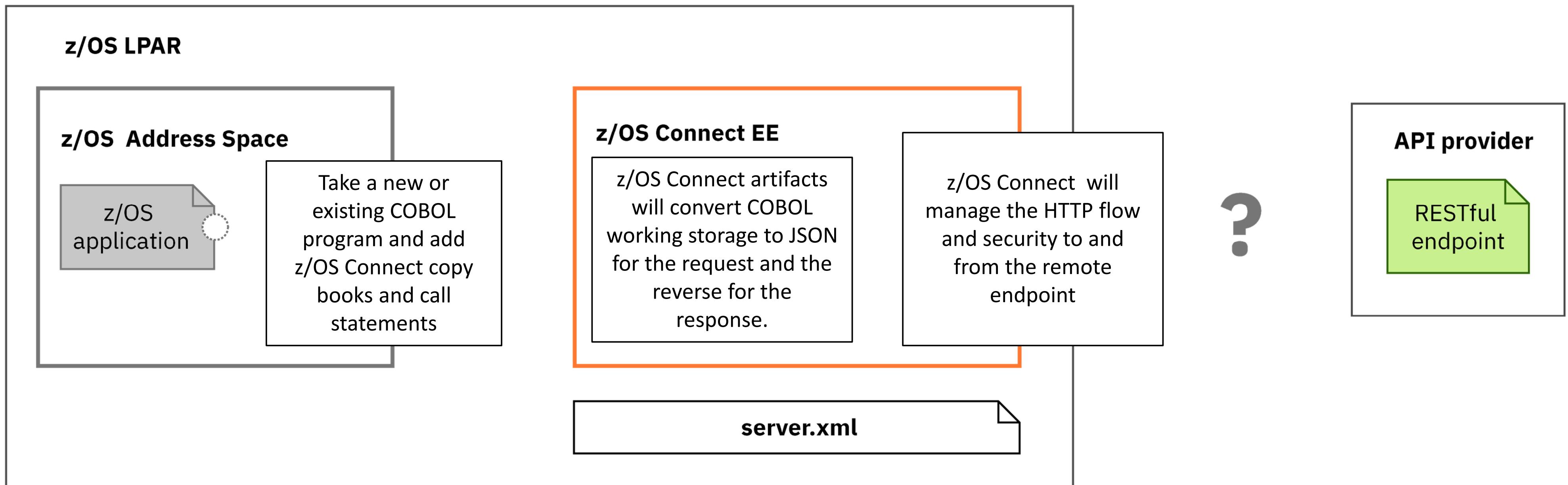
Developing COBOL API Requester applications



z/OS Connect generates API requester artifacts are used to invoke APIs from z/OS application



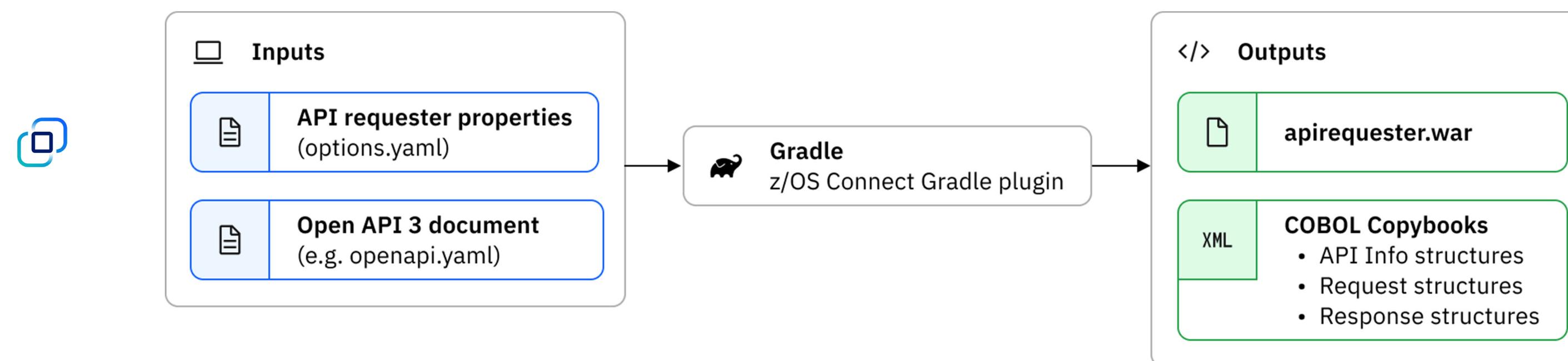
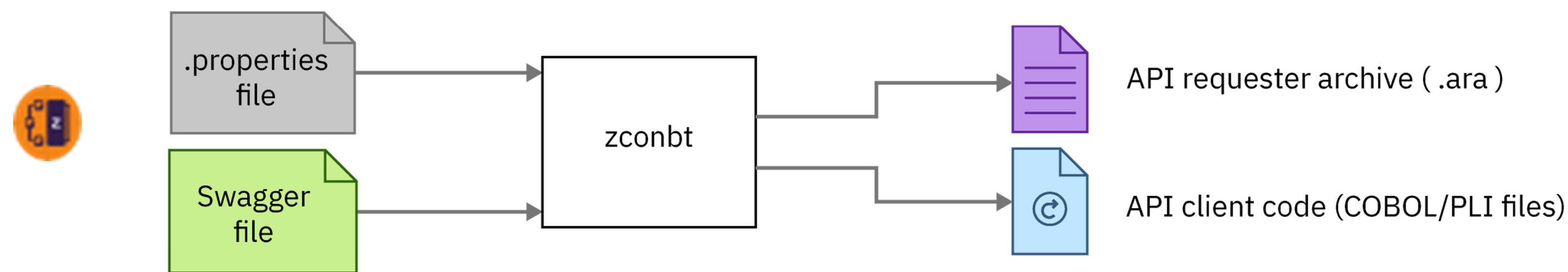
The basic steps required to invoke a remote API



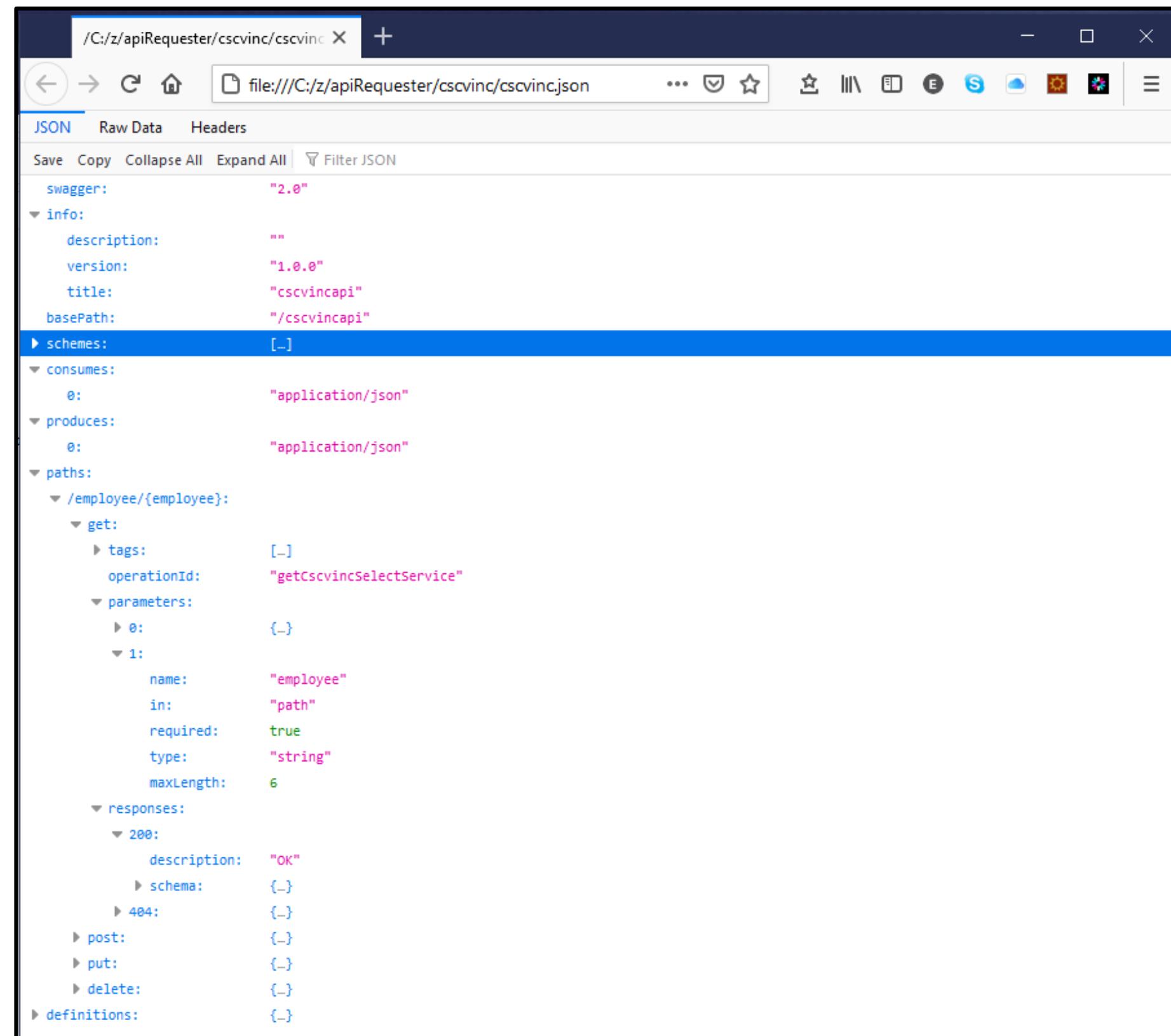
API requester development starts with the API's specification document



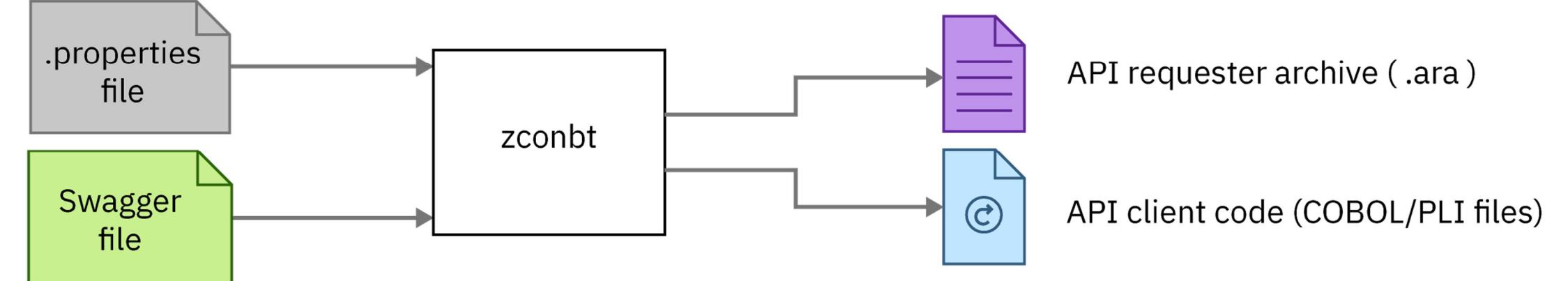
z/OS Connect tooling consuming the API's specification description to generate COBOL copy books and an API requester archive (ARA) file for an OpenAPI 2 API or a web archive (WAR) file for an OpenAPI 3 API.



For APIs described using Open API 2, use the z/OS Connect build toolkit (zconbt)



The screenshot shows a browser window displaying an Open API 2 JSON file named cscvinc.json. The JSON structure includes fields like swagger version (2.0), info (title: cscvincapi, version: 1.0.0), paths (e.g., /employee/{employee}), and responses (e.g., 200 OK). The browser interface includes tabs for JSON, Raw Data, Headers, and various save/copy options.

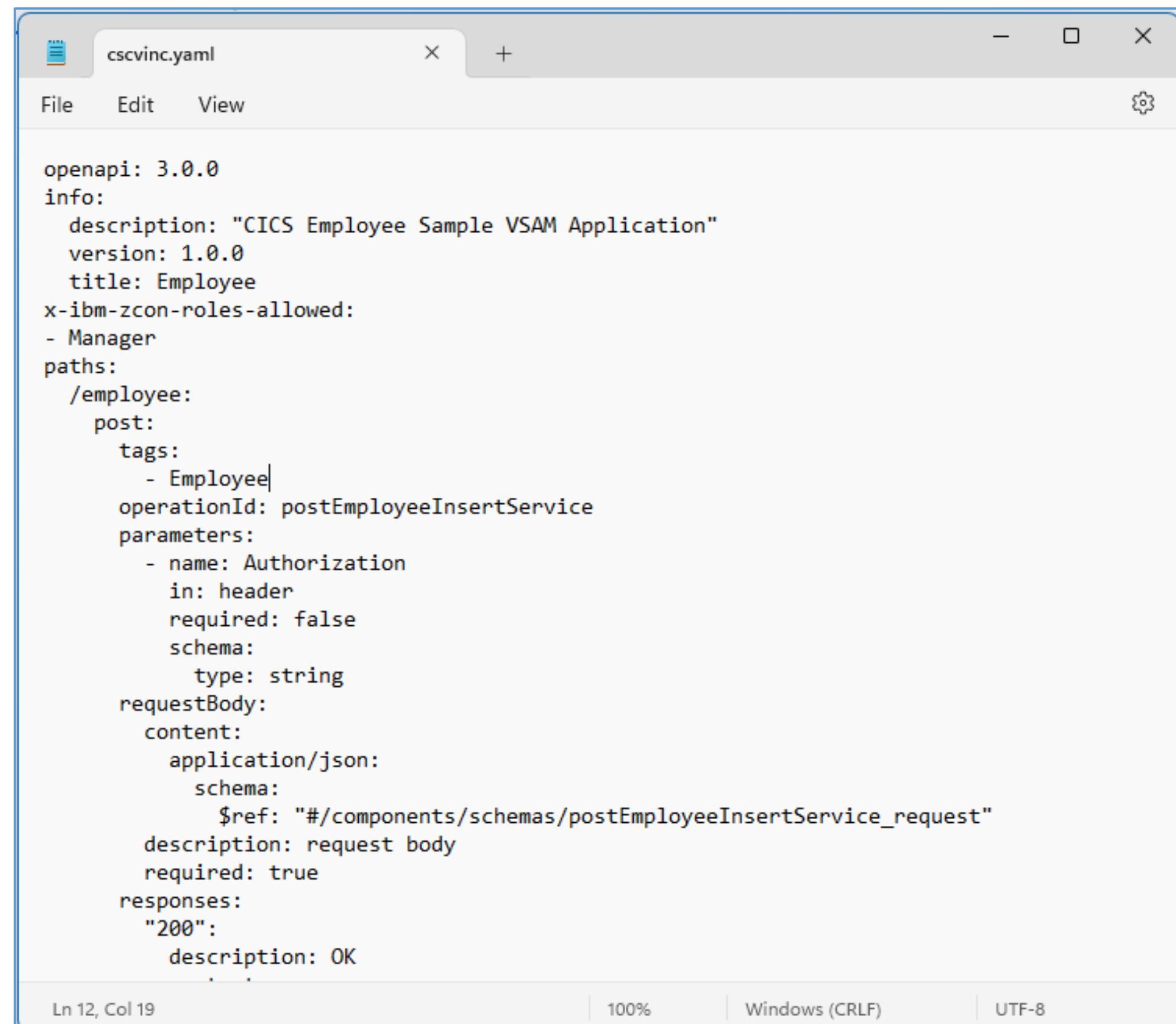


properties file#

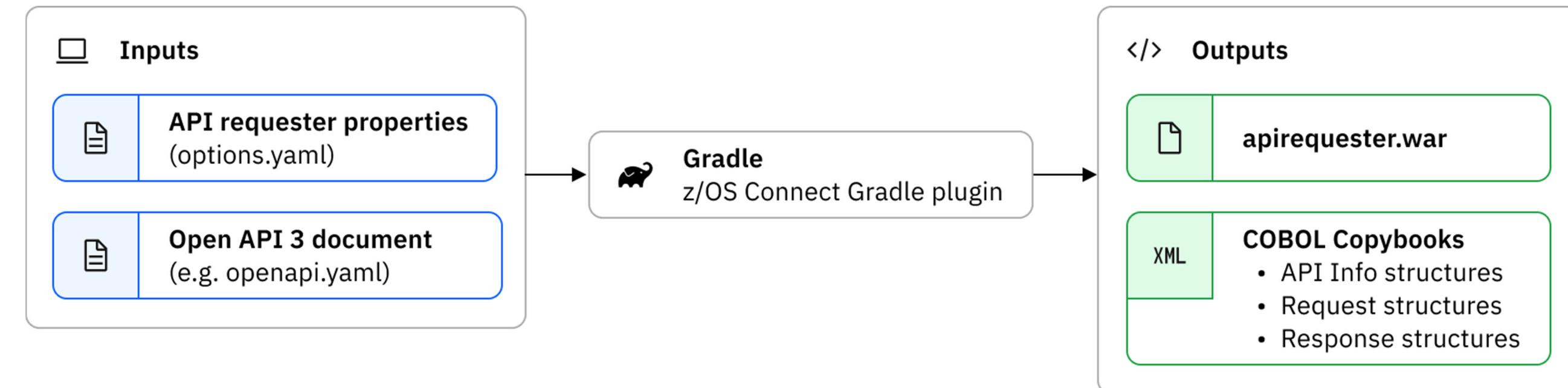
```
apiDescriptionFile=./cscvinc.json
dataStructuresLocation=./syslib
apiInfoFileLocation=./syslib
logFileDirectory=./logs
language=COBOL
connectionRef=cscvincAPI
requesterPrefix=csc
```

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

For APIs describe using Open API 3, use the z/OS Connect Gradle plug-in



```
cscvinc.yaml
File Edit View
openapi: 3.0.0
info:
  description: "CICS Employee Sample VSAM Application"
  version: 1.0.0
  title: Employee
x-ibm-zcon-roles-allowed:
- Manager
paths:
  /employee:
    post:
      tags:
        - Employee
      operationId: postEmployeeInsertService
      parameters:
        - name: Authorization
          in: header
          required: false
          schema:
            type: string
      requestBody:
        content:
          application/json:
            schema:
              $ref: "#/components/schemas/postEmployeeInsertService_request"
            description: request body
            required: true
      responses:
        "200":
          description: OK
Ln 12, Col 19 | 100% | Windows (CRLF) | UTF-8
```



Gradle plug-in properties and options#

```
apiKeyMaxLength=255
characterVarying=NO
Operations=getEmployeeSelectService
language=COBOL
connectionRef=cscvincAPI
requesterPrefix=csc
```

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



Both z/OS Connect tool generate application artifacts

- The z/OS Connect build tooling, e.g., `zconbt` or the *Gradle plugin-in*, will generate at most, 3 copy books per method found in the specification document and either an **API requester archive file (ARA)** for an OpenAPI 2 APIs or a **web archive file (WAR)** for an OpenAPI 3 APIs.

```
zconbt.bat -p=./cscvinc.properties -f=./cscvinc.ara
BAQB0000I: z/OS Connect Enterprise Edition 3.0 Build Toolkit Version 1.5 (20210816-0926).
BAQB0008I: Creating API requester archive from configuration file ./cscvinc.properties.
BAQB0040I: The generated API requester is automatically named cscvincapi_1.0.0 based on the title cscvincapi and version 1.0.0 of the API to be called.

. . . .
Total 4 operation(s) (success: 4, ignored: 0) defined in api description file: ./cscvinc.json
----- Successfully processed operation(s) -----
operationId: getCscvincSelectService, basePath: /cscvincapi, relativePath: /employee/{employee}, method: GET
- request data structure : CSC00Q01
- response data structure : CSC00P01
- api info file : CSC00I01

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

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

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

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



Tech-Tip: The copy books naming convention

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

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

Tech-Tip: BTW, the z/OS Connect Build Toolkit can be executed on z/OS

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

cscvinc.properties

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

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



COBOL client programming considerations for both OpenAPI 2 and OpenAPI 3 APIs



First, be aware of COBOL working storage implications

API specification properties are usually not constrained, this can lead to excessive working storage consumption¹

The screenshot shows a JSON API specification for 'ATSContactPreferences'. Several properties are highlighted with red circles:

- 'communicationPreferences' is an array.
- 'memberCodeableConcept' is an array.
- 'lastName' is an array.
- 'birthDate' is a string.

The screenshot shows a COBOL source code snippet from 'ATS01P01 - Notepad'. Multiple fields are defined with the same data type:
09 memberContactsResponse-num PIC S9(9) COMP-5 SYNC.
09 memberContactsResponse OCCURS 255.
12 umi-num PIC S9(9) COMP-5 SYNC.
12 umi.
15 umi2-length
15 umi2 PIC S9999 COMP-5
PIC X(255).
12 pin-num PIC S9(9) COMP-5 SYNC.
12 pin.
15 pin2-length
15 pin2 PIC S9999 COMP-5
PIC X(255).
12 firstName-num PIC S9(9) COMP-5 SYNC.
12 firstName.
15 firstName2-length
15 firstName2 PIC S9999 COMP-5
PIC X(255).
12 middleName-num PIC S9(9) COMP-5 SYNC.
12 middleName.
15 middleName2-length
15 middleName2 PIC S9999 COMP-5
PIC X(255).
12 lastName-num PIC S9(9) COMP-5 SYNC.
12 lastName.
15 lastName2-length
15 lastName2 PIC S9999 COMP-5
PIC X(255).



These are the API Requester generation properties available to help

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

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

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

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

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

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



An alternative, consider adding explicit constraints to the properties

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

```
/C:/z/apiRequester/ATS/MemberCo X +  
file:///C:/z/apiRequester/ATS/MemberContactPrefe  
JSON Raw Data Headers  
Save Copy Collapse All Expand All Filter JSON  
type: "array"  
communicationPreferences:  
items:  
$ref: "#/definitions/member-communication-preferences"  
type: "array"  
maxItems: 10  
memberCodeableConcept:  
description: "Multiple member codes"  
items:  
$ref: "#/definitions/member-codeable-concept"  
type: "array"  
object:  
type:  
member-contacts-request:  
title: "Member Contacts Request"  
description: "Read-only request data to search for member contact information."  
properties:  
umi:  
description: "Unique Member Id. This value is at a contract level. All members under one contract have the same UMI."  
example: "122222444001"  
type: "string"  
maxLength: 12  
firstName:  
description: "Member first name or given name."  
example: "Arthur"  
type: "string"  
maxLength: 30  
lastName:  
description: "Member last name or family name."  
example: "smith"  
10  
Highlight All Match Case Match Diacritics Whole V X
```

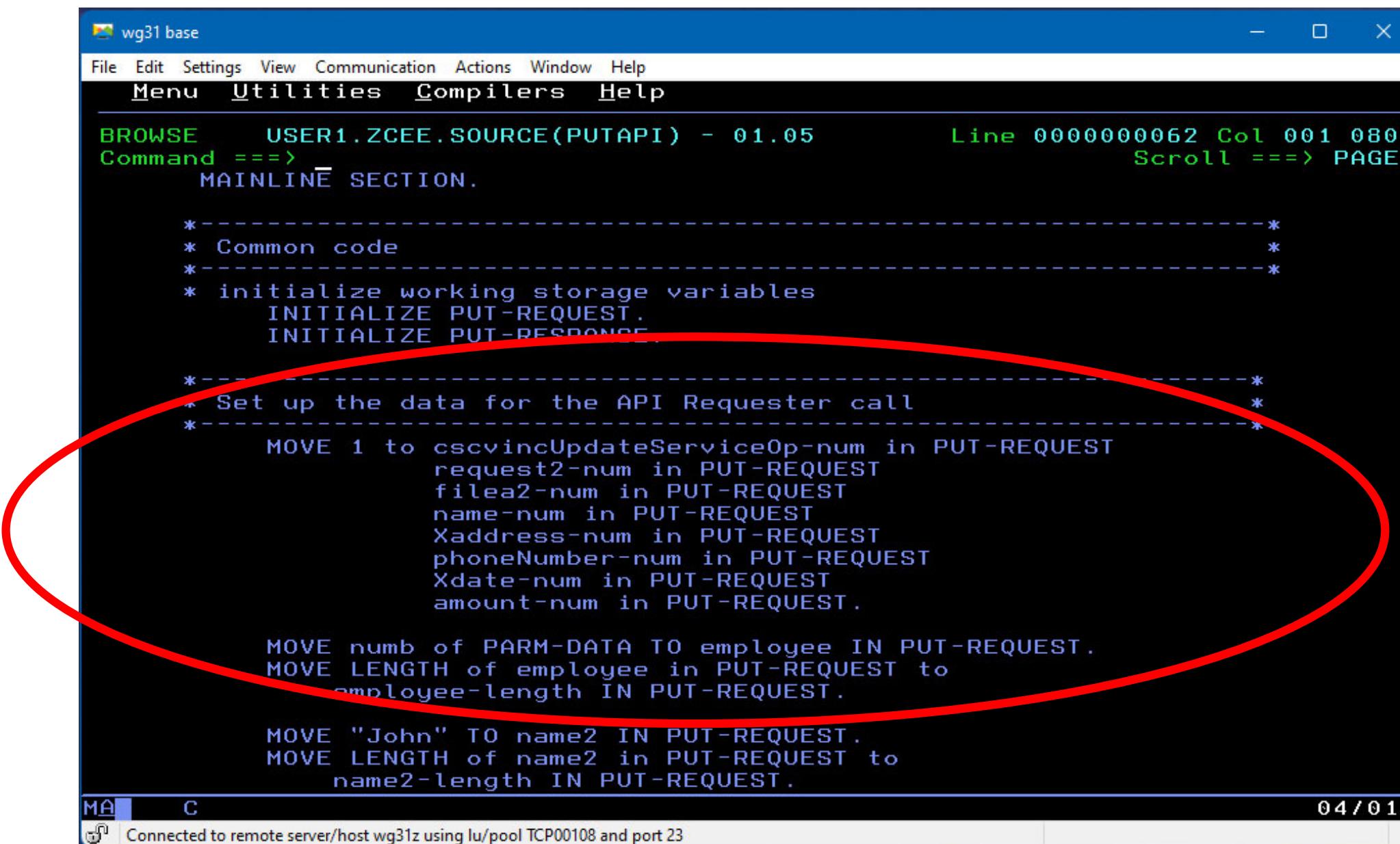
```
ATS01P01 - Notepad  
File Edit Format View Help  
* Comments for field 'filler':  
* This is a filler entry to ensure the correct padding for a  
* structure. These slack bytes do not contain any application  
* data.  
* 15 filler PIC X(3).  
*  
*  
* ++++++  
06 RespBody.  
09 memberContactsResponse-num PIC S9(9) COMP-5 SYNC.  
09 memberContactsResponse OCCURS 10.  
12 umi-num PIC S9(9) COMP-5 SYNC.  
12 umi.  
15 umi2-length  
15 umi2  
PIC S9999 COMP-5  
PIC X(12).  
12 pin-num PIC S9(9) COMP-5 SYNC.  
12 pin.  
15 pin2-length  
15 pin2  
PIC S9999 COMP-5  
PIC X(255).  
12 firstName-num PIC S9(9) COMP-5 SYNC.  
12 firstName.  
15 firstName2-length  
15 firstName2  
PIC S9999 COMP-5  
PIC X(30).  
12 middleName-num PIC S9(9) COMP-5 SYNC.  
12 middleName.  
15 middleName2-length  
15 middleName2  
PIC S9999 COMP-5  
PIC X(30).  
Ln 783, Col 70 100% Windows (CRLF) UTF-8
```



The number of occurrences of an entry can also be ambiguous

In this case the COBOL copy book will include a counter variable (**-num**) for each variable whose number of occurrences is ambiguously. Review the generated COBOL and provide the number of occurrences of these variables in a request message or use the value of this variable to know how many instances were returned.

If the JSON property is an *array*, then the variable name is appended with **-num** and the value of this variable provides the number of occurrences or array entries of this array, including 0. If the JSON variable was an *Object* type, then the variable name is appended with **-existence** and this variable contains either a 0 or 1 to specify whether an object was returned in the response.



```
wg31 base
File Edit Settings View Communication Actions Window Help
Menu Utilities Compilers Help
BROWSE USER1.ZCEE.SOURCE(PUTAPI) - 01.05      Line 0000000062 Col 001 080
Command ==>                                         Scroll ==> PAGE
MAINLINE SECTION.

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

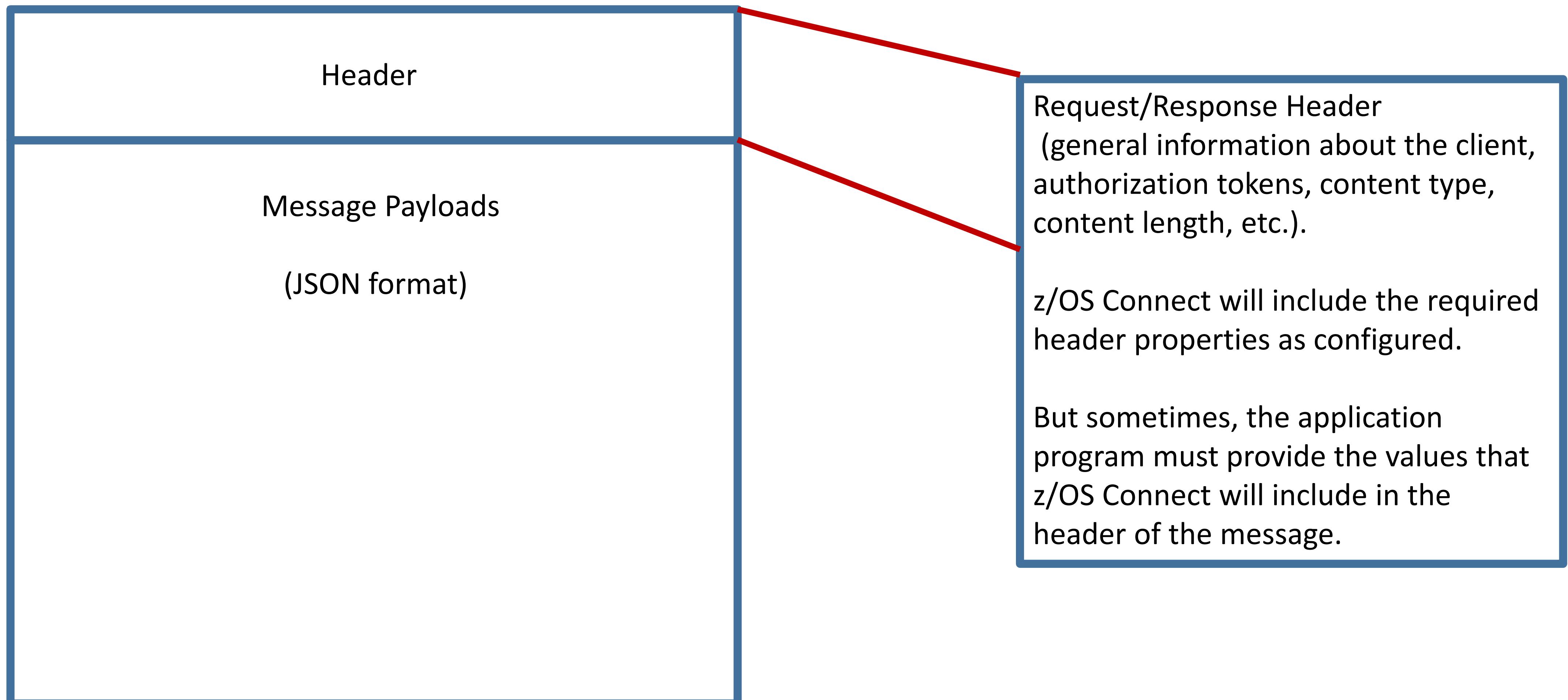
*-----*
* Set up the data for the API Requester call
*-----*
  MOVE 1 to cscvincUpdateServiceOp-num in PUT-REQUEST
  request2-num in PUT-REQUEST
  filea2-num in PUT-REQUEST
  name-num in PUT-REQUEST
  Xaddress-num in PUT-REQUEST
  phoneNumber-num in PUT-REQUEST
  Xdate-num in PUT-REQUEST
  amount-num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

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

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

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

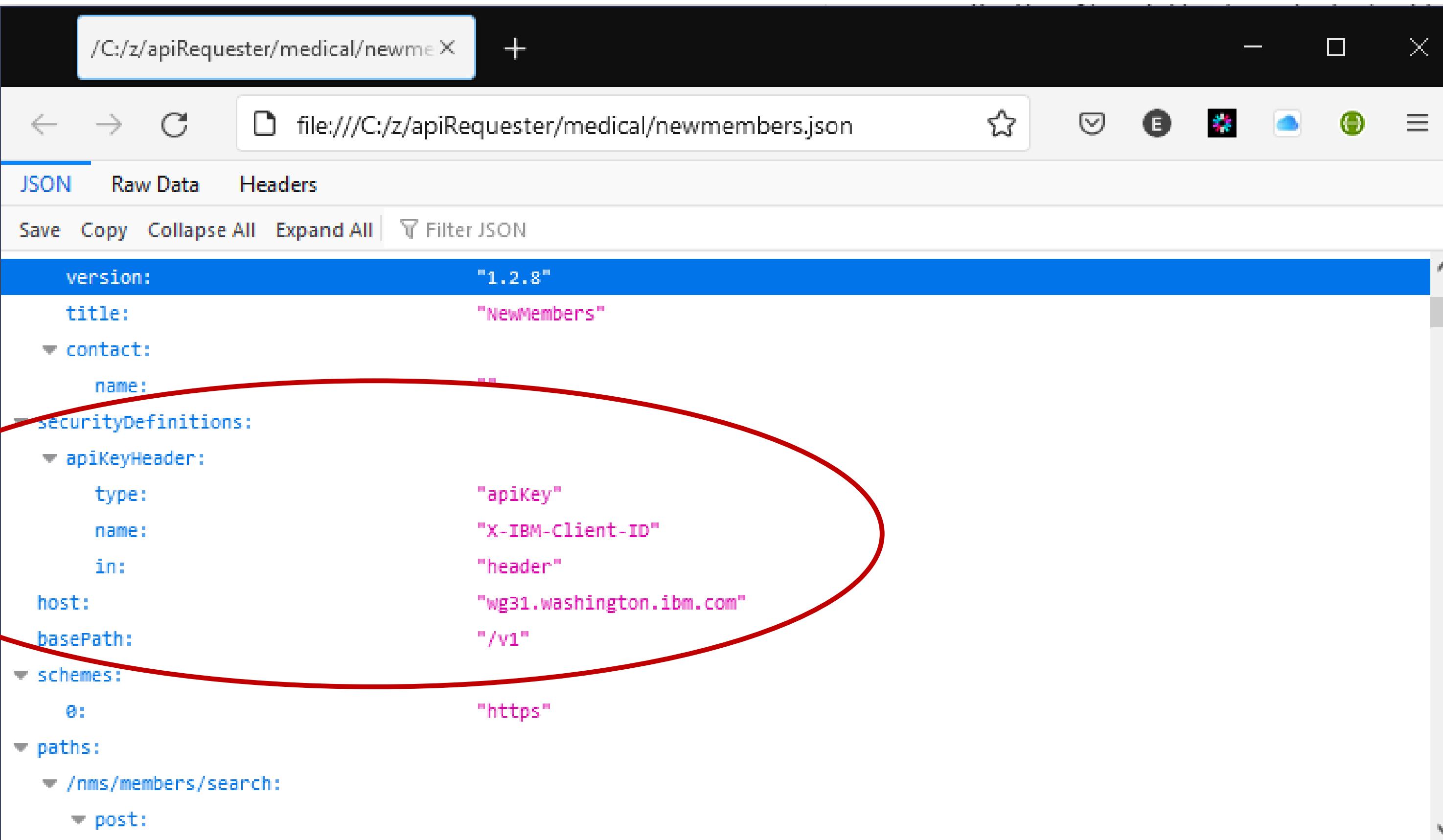
Let's look at the anatomy of Request and Response Messages





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

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



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

Via a HTTP header

GET /something HTTP/1.1

X-API-Key: abcdef12345

Or via a query parameter

GET /something?api_key=abcdef12345



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

Or if required and not in the specification, then generation properties must be used to have the required supported added to the request copybook ...

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

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

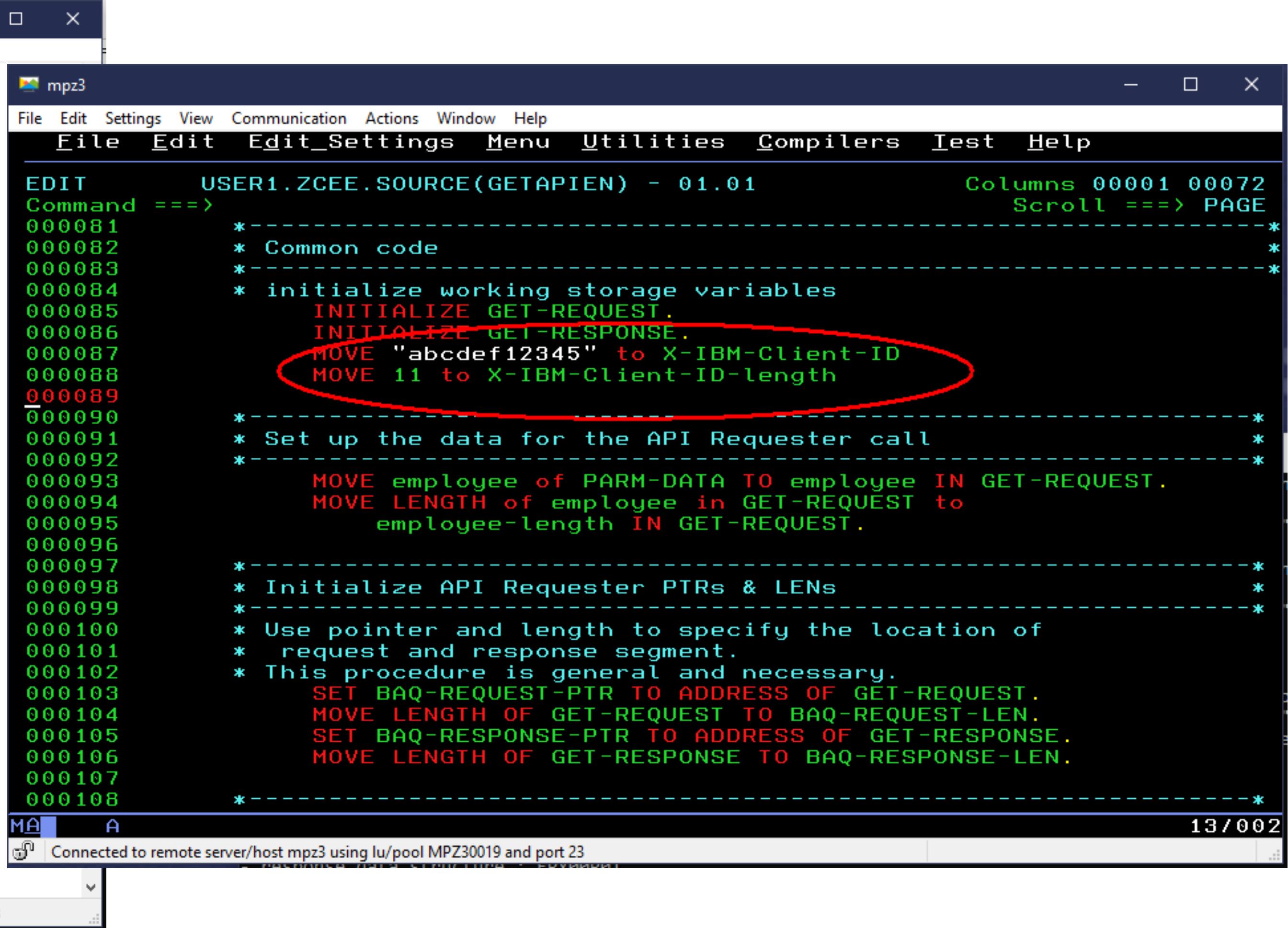
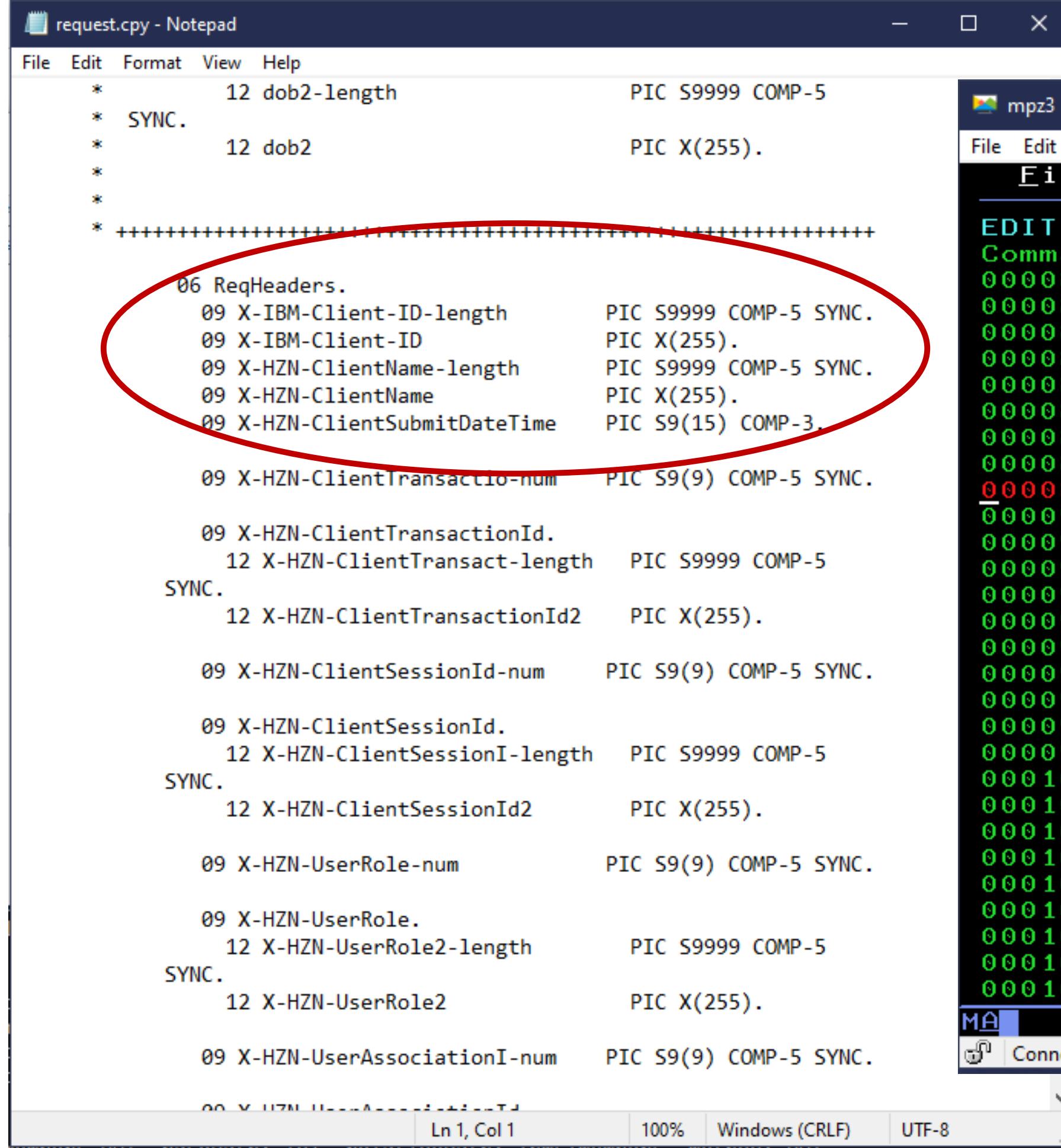
apiKeyParmNameInQuery - Specify the name of an API key that is sent in a query string. The value of this parameter can be set in a comma separated list of a combination of client ID and client secret. For example, you can set **apiKeyParmNameInQuery**=query-IBM-Client-ID, query-IBM-Client-secret when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

apiKeyParmNameInCookie - Specify the name of an API key that is sent as a cookie. The value of this parameter can be set in a comma separated list of a combination of client ID and client secret cookie names. The actual client ID and client secret values are set in the z/OS application. For example, you can set **apiKeyParmNameInCookie**=query-IBM-Client-ID,client-secret when a client ID and a client secret are used to protect an API.



Either way, the application provides the value of the API key

The generated request copy book includes a ReqHeaders structure which can be used to provide values for the API key



```
request.cpy - Notepad
File Edit Format View Help
*      12 dob2-length          PIC S9999 COMP-5
* SYNC.
*      12 dob2                PIC X(255).
*
*
* ++++++
06 ReqHeaders.
09 X-IBM-Client-ID-length  PIC S9999 COMP-5 SYNC.
09 X-IBM-Client-ID        PIC X(255).
09 X-HZN-ClientName-length PIC S9999 COMP-5 SYNC.
09 X-HZN-ClientName       PIC X(255).
09 X-HZN-ClientSubmitDateTime PIC S9(15) COMP-3.
09 X-HZN-ClientTransaction-num PIC S9(9) COMP-5 SYNC.

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

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

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

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

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

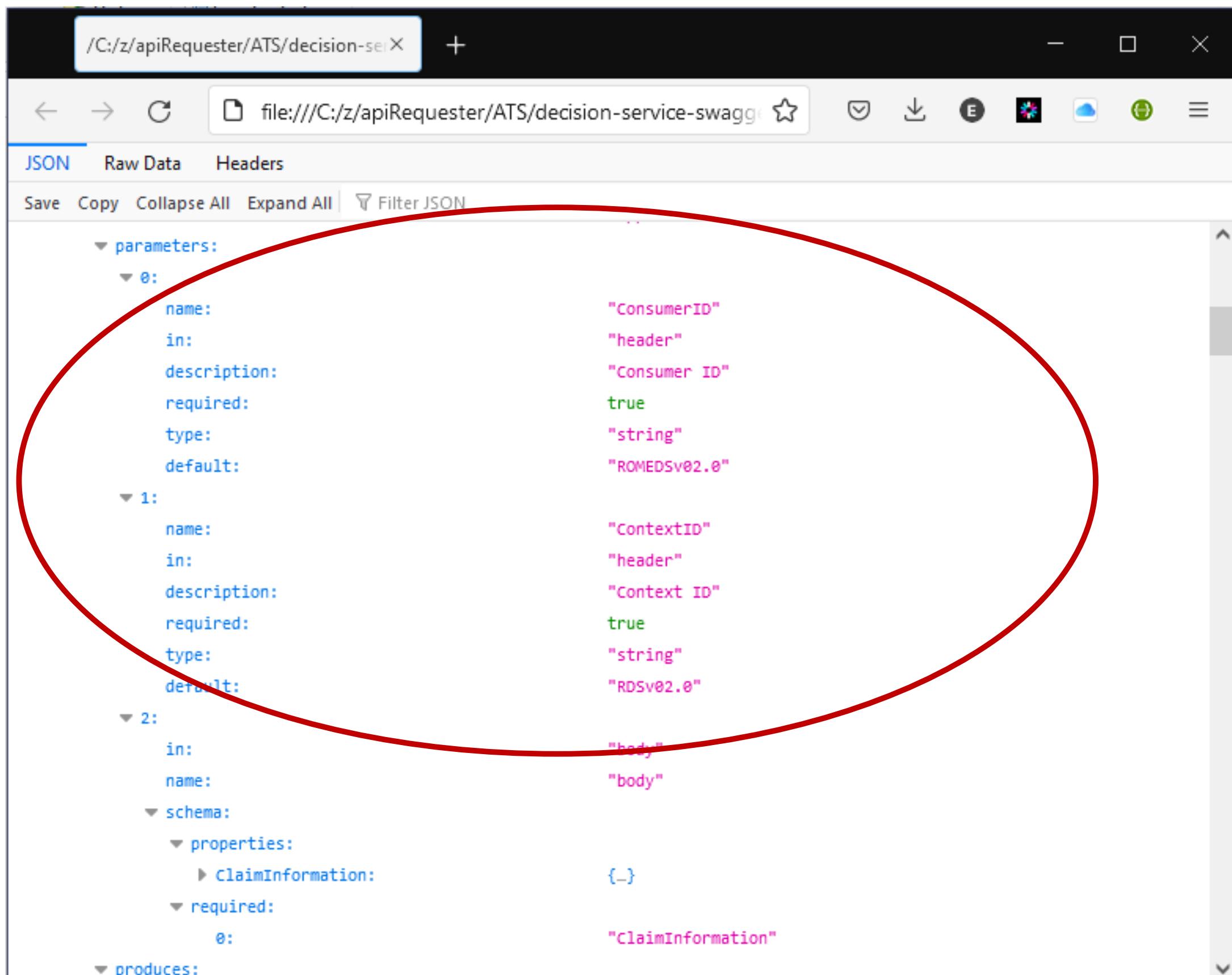
09 X-HZN-UserAssociationI-num PIC S9(9) COMP-5 SYNC.

mpz3
File Edit Settings View Communication Actions Window Help
File Edit Edit_Settings Menu Utilities Compilers Test Help
EDIT          USER1.ZCEE.SOURCE(GETAPIEN) - 01.01
Command ===> Columns 00001 00072
000081 *-----*
000082 * Common code
000083 *-----*
000084 * initialize working storage variables
000085   INITIALIZE GET-REQUEST.
000086   INITIALIZE GET-RESPONSE.
000087   MOVE "abcdef12345" to X-IBM-Client-ID
000088   MOVE 11 to X-IBM-Client-ID-length
000089
000090 *-----*
000091 * Set up the data for the API Requester call
000092 *-----*
000093   MOVE employee of PARM-DATA TO employee IN GET-REQUEST.
000094   MOVE LENGTH of employee in GET-REQUEST to
000095     employee-length IN GET-REQUEST.
000096 *-----*
000097 * Initialize API Requester PTRs & LENs
000098 *-----*
000099 * Use pointer and length to specify the location of
000100 * request and response segment.
000101 * This procedure is general and necessary.
000102   SET BAQ-REQUEST-PTR TO ADDRESS OF GET-REQUEST.
000103   MOVE LENGTH OF GET-REQUEST TO BAQ-REQUEST-LEN.
000104   SET BAQ-RESPONSE-PTR TO ADDRESS OF GET-RESPONSE.
000105   MOVE LENGTH OF GET-RESPONSE TO BAQ-RESPONSE-LEN.
000106
000107
000108 *-----*
```

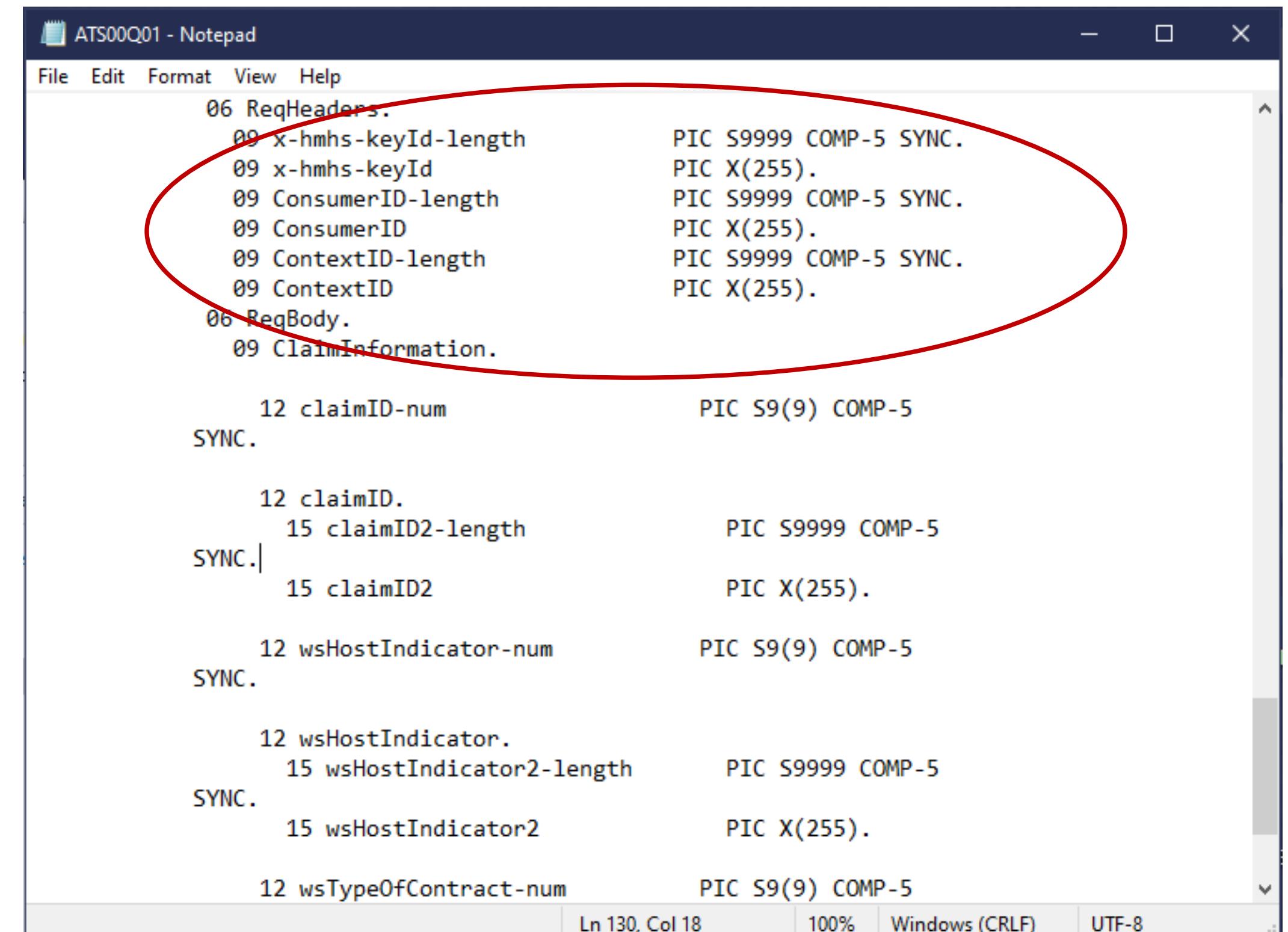


And there may be other required custom header properties

The application may also need to set the values for other header properties that may be required by the API and must be set by the application. These properties are defined in the specification document.



```
/C:/z/apiRequester/ATS/decision-service-swagg  
+  
file:///C:/z/apiRequester/ATS/decision-service-swagg  
JSON Raw Data Headers  
Save Copy Collapse All Expand All Filter JSON  
parameters:  
0:  
name: "ConsumerID"  
in: "header"  
description: "Consumer ID"  
required: true  
type: "string"  
default: "ROMEDSv02.0"  
1:  
name: "ContextID"  
in: "header"  
description: "Context ID"  
required: true  
type: "string"  
default: "RDSv02.0"  
2:  
in: "body"  
name: "body"  
schema:  
properties:  
claimInformation: {}  
required:  
0:  
ClaimInformation  
produces:
```

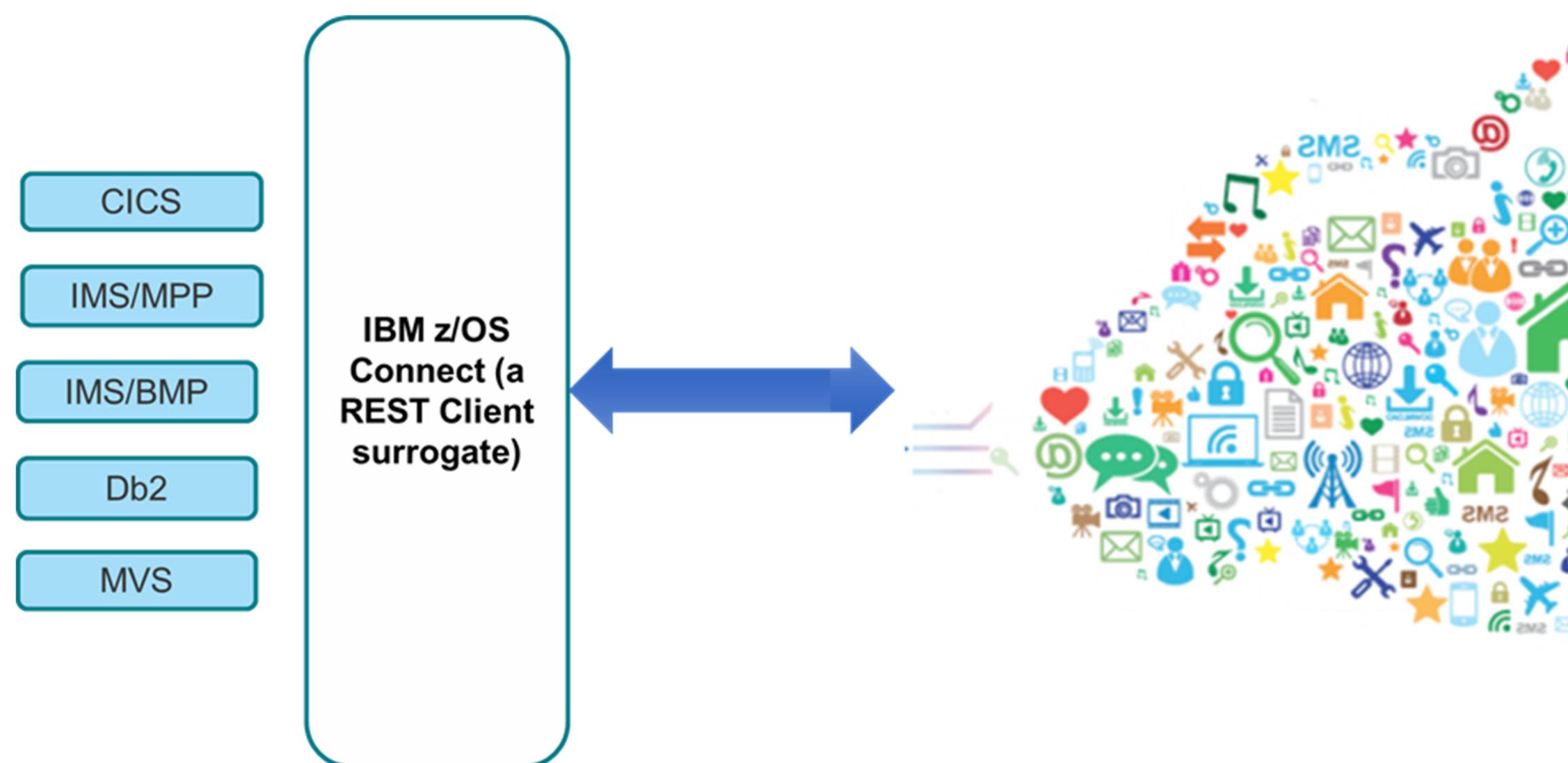


```
ATS00Q01 - Notepad  
File Edit Format View Help  
06 ReqHeaders.  
09 x-hmhs-keyId-length  
09 x-hmhs-keyId  
09 ConsumerID-length  
09 ConsumerID  
09 ContextID-length  
09 ContextID  
06 ReqBody.  
09 ClaimInformation.  
12 claimID-num SYNC. PIC S9(9) COMP-5  
12 claimID SYNC.| 15 claimID2-length PIC S9999 COMP-5  
15 claimID2 PIC X(255).  
12 wsHostIndicator-num SYNC. PIC S9(9) COMP-5  
12 wsHostIndicator SYNC. 15 wsHostIndicator2-length PIC S9999 COMP-5  
15 wsHostIndicator2 PIC X(255).  
12 wsTypeOfContract-num PIC S9(9) COMP-5  
Ln 130, Col 18 100% Windows (CRLF) UTF-8
```



Developing API Requesters

For APIs defined using an OpenAPI 2 specification



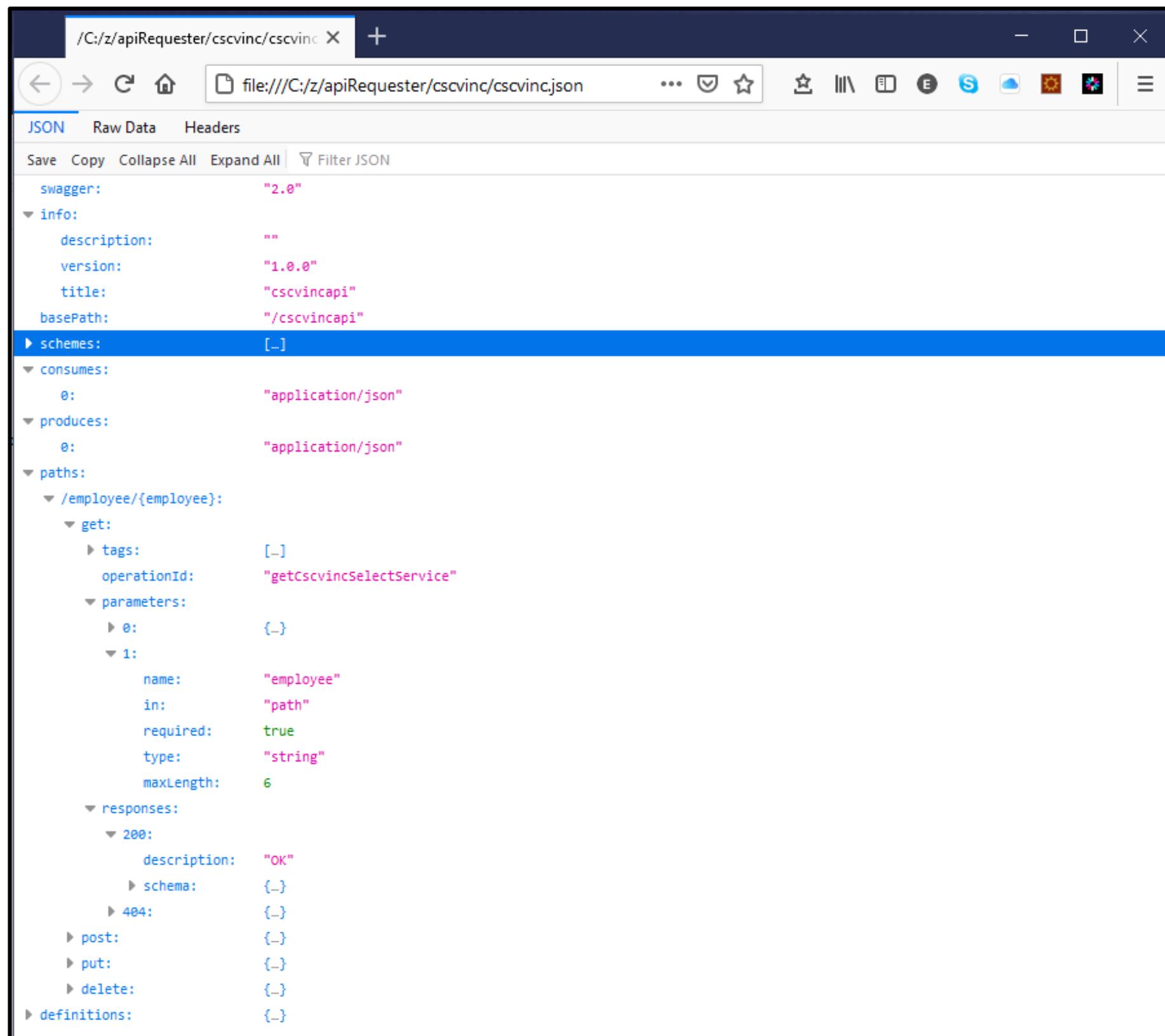


Installing the IBM z/OS Connect build toolkit -

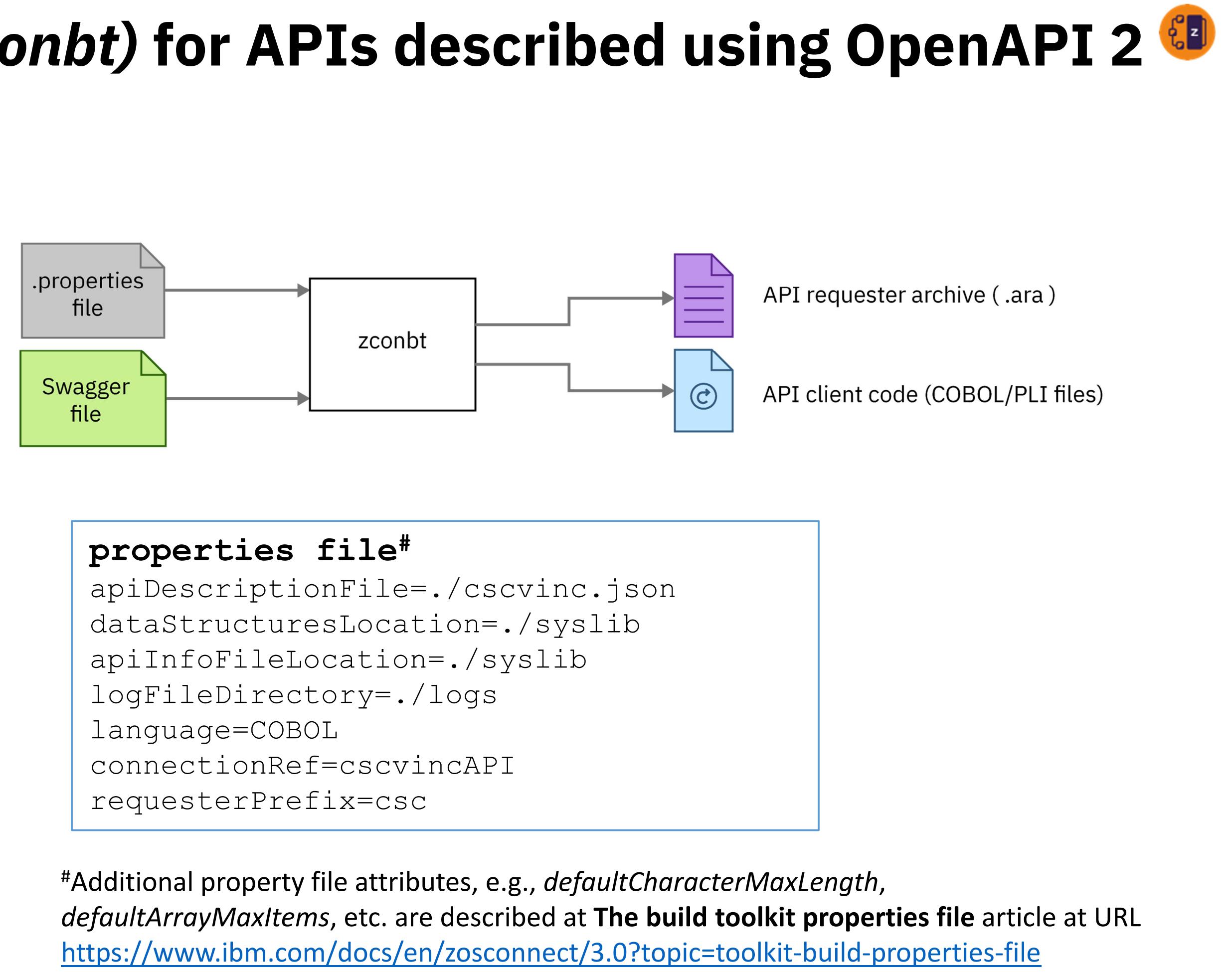
<https://www.ibm.com/docs/en/zos-connect/zosconnect/3.0?topic=installing-build-toolkit>

The screenshot shows a web browser displaying the IBM z/OS Connect documentation. The URL in the address bar is <https://www.ibm.com/docs/en/zos-connect/zosconnect/3.0?topic=installing-build-toolkit>. The page title is "Installing the IBM z/OS Connect build toolkit". The left sidebar lists several topics under "IBM z/OS Connect": "Change version" (set to 3.0), "Show full table of contents" (checked), "Filter on titles", "Installing the build toolkit" (selected), "Updating z/OS Connect (OpenAPI 2)", "Converting to z/OS Connect (OpenAPI 2) Unlimited", "Installing z/OS Explorer and the z/OS Connect API toolkit", "Updating z/OS Explorer and the z/OS Connect API toolkit", "Migrating from z/OS Connect V1", and "Upgrading from z/OS Connect EE V2". The main content area starts with a breadcrumb trail: "All products / IBM z/OS Connect / IBM z/OS Connect (OpenAPI 2) / 3.0 /". Below the breadcrumb is a feedback section with "Was this topic helpful?" buttons for "Yes" and "No". The main heading is "Installing the IBM z/OS Connect build toolkit". Below the heading is the last updated date: "Last Updated: 2023-05-26". The text describes the purpose of the toolkit: "Install the IBM® z/OS® Connect build toolkit to create service archive (.sar) files or artifacts for an API requester." A "About this task" section follows, stating: "The build toolkit is available as a command-line tool or a Software Development Kit (SDK) for inclusion in other products. For more information, read the Javadoc included in the zconbt.zip file or the Build Toolkit SPI in the Reference section. You can also find examples in GitHub." A "Procedure" section contains two numbered steps: 1. Copy the zconbt.zip file, in binary mode, from the product installation directory to a directory on your local workstation or to a UNIX System Services directory on z/OS. 2. Extract the contents of the zconbt.zip file into the current working directory.

Use the z/OS Connect build toolkit (zconbt) for APIs described using OpenAPI 2



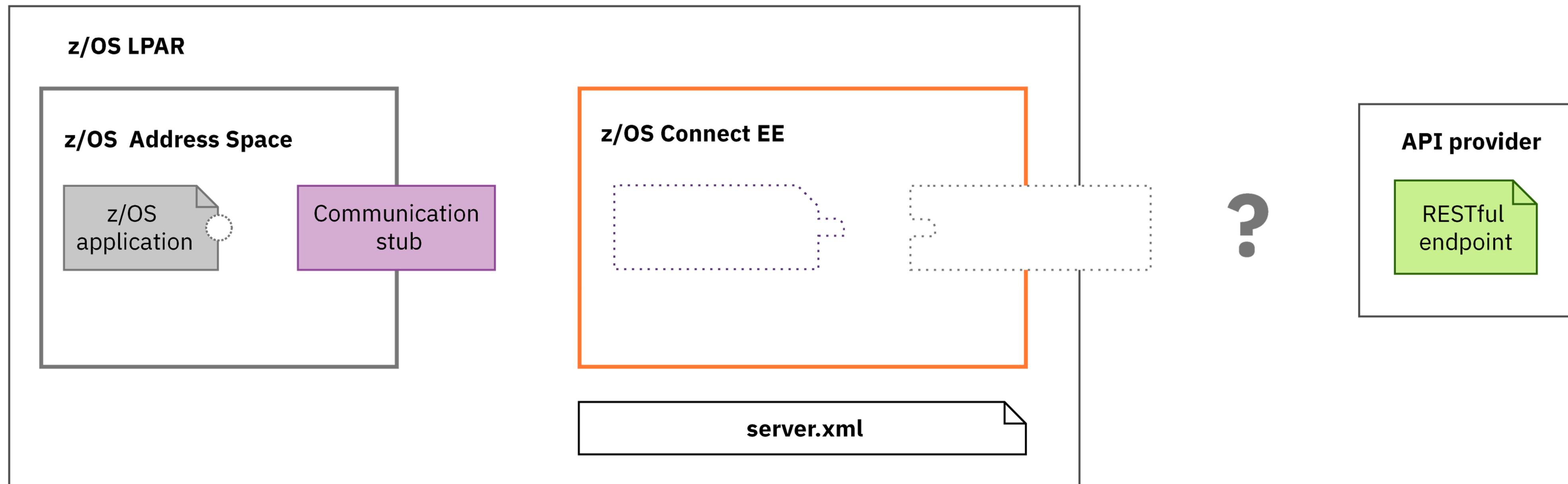
The screenshot shows a browser window displaying an OpenAPI 2 JSON schema. The URL is /C:/z/apiRequester/cscvinc/cscvinc.json. The schema includes sections for info, schemes, consumes, produces, paths, and definitions. A specific path for employees is shown with its get method, which includes tags, operationId, parameters (with one named 'employee'), and responses (including 200 OK and 404). The swagger version is 2.0.





How to access an API from a COBOL program (OpenAPI 2)

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



Configure a communication stub.

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

```
*-----  
* Call the communication stub  
*-----  
* Call the subsystem-supplied stub code to send  
* API request to zCEE  
CALL COMM-STUB-PGM-NAME USING  
      BY REFERENCE  GET-INFO-OPER1  
      BY REFERENCE  BAQ-REQUEST-INFO  
      BY REFERENCE  BAQ-REQUEST-PTR  
      BY REFERENCE  BAQ-REQUEST-LEN  
      BY REFERENCE  BAQ-RESPONSE-INFO  
      BY REFERENCE  BAQ-RESPONSE-PTR  
      BY REFERENCE  BAQ-RESPONSE-LEN.  
      END-OF-CODE-SEGMENT-1  
      END-OF-CODE-SEGMENT-2
```



Steps to calling an remote API

Include the generated copy books in a COBOL program

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

  * Copy API Requester required copybook
  COPY BAQRINFO.

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

  01 API_RESPONSE.
    COPY CSC02P01.

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

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

API-REQUEST

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

API-RESPONSE

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

API-INFO-OPER1

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



Steps to calling a remote API

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

The screenshot shows the following components:

- GETAPI**: The main editor window containing assembly language code. It highlights several sections:
 - * Set up the data for the API Requester call
 - * Initialize API Requester PTRs & LENs
 - * Use pointer and length to specify the location of request and response segment.
 - * 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.
- CSC00I01**: A copy book definition window showing fields for the API REQUEST structure.
- CSC00Q01**: A copy book definition window showing fields for the API RESPONSE structure.
- CSC00P01**: A copy book definition window showing fields for the communication stub parameters.



Steps to calling a remote API

Access the results that have been placed in working storage

The screenshot displays two IBM i environments. On the left, the GETAPI application shows AS/400 RPG code. A red box highlights the successful response logic, and another red box highlights the error handling logic. Red arrows point from these boxes to specific fields in the mpz3 browser on the right. The mpz3 browser shows the ZCEE30.SBAQC0B(BAQRINFO) copybook, with a red arrow pointing to the BAQ-RETURN-CODE field which is set to BAQ-SUCCESS.

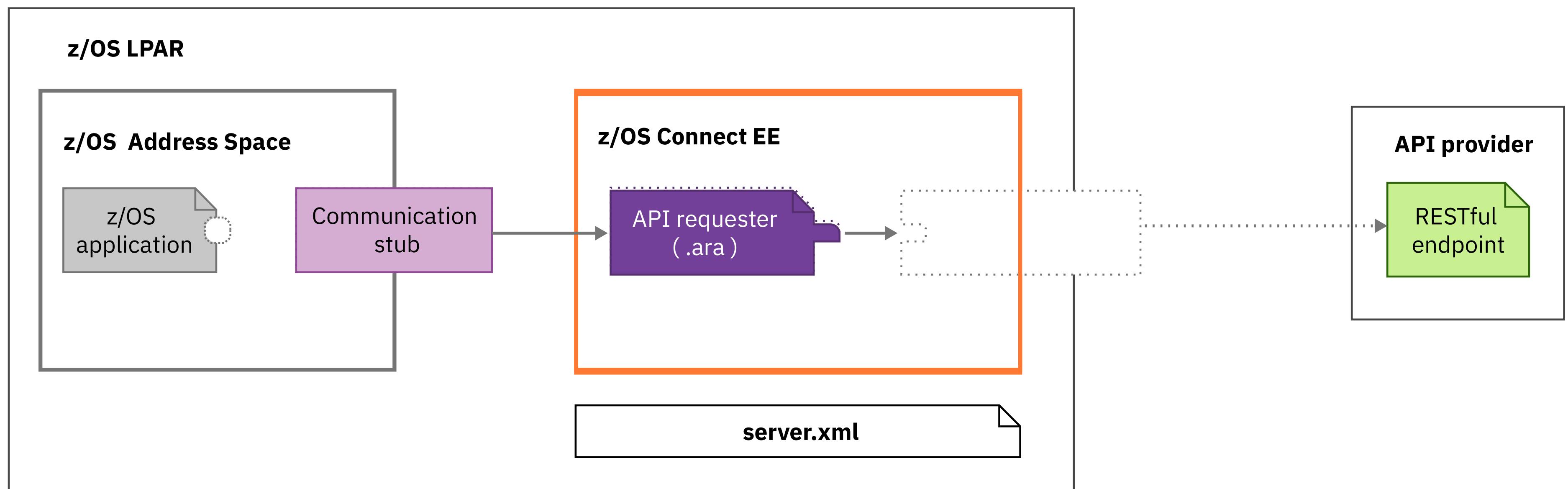
```
BY REFERENCE BAQ-RESPONSE-LEN.  
* The BAQ-RETURN-CODE field in 'BAQRINFO' indicates whether this  
* API call is successful.  
  
* When BAQ-RETURN-CODE is 'BAQ-SUCCESS', response is  
* successfully returned and fields in RESPONSE copybook  
* can be obtained. Display the translation result.  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

mpz3
File Edit Settings View Communication Actions Window Help
Menu Utilities Compilers Help
BROWSE ZCEE30.SBAQC0B(BAQRINFO)
Command ==>
Line 0000000066 Col 001 080
Scroll ==> PAGE
01 BAQ-RESPONSE-INFO.
03 BAQ-RESPONSE-INFO-COMP-LEVEL PIC S9(9) COMP-5 SYNC VALUE 0.
03 BAQ-STUB-NAME PIC X(8).
03 BAQ-RETURN-CODE PIC S9(9) COMP-5 SYNC.
 88 BAQ-SUCCESS VALUE 0.
 88 BAQ-ERROR-IN-API VALUE 1.
 88 BAQ-ERROR-IN-ZCEE VALUE 2.
 88 BAQ-ERROR-IN-STUB VALUE 3.
 88 BAQ-ERROR-NO-RESPONSE VALUE 4.
03 BAQ-STATUS-CODE PIC S9(9) COMP-5 SYNC.
03 BAQ-STATUS-MESSAGE PIC X(1024).
03 BAQ-STATUS-MESSAGE-LEN PIC S9(9) COMP-5 SYNC.
***** Bottom of Data *****
MA B
Connected to remote server/host mpz3 using lu/pool MPZ30021 and port 23
18/058

Note that the return code only addresses z/OS Connect related status, not the underlying HTTP status code.



Deploy the API requester (.ara) archive



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



Tech-Tip: Deploying API requester archive files

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

The screenshot shows the Postman application interface. A POST request is being made to the URL <https://wg31.washington.ibm.com:9483/zosConnect/apiRequesters>. The request body is set to 'binary' type and contains the file 'filea.ara'. The response status is 201 Created, and the JSON response body is displayed as:

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

Command:

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

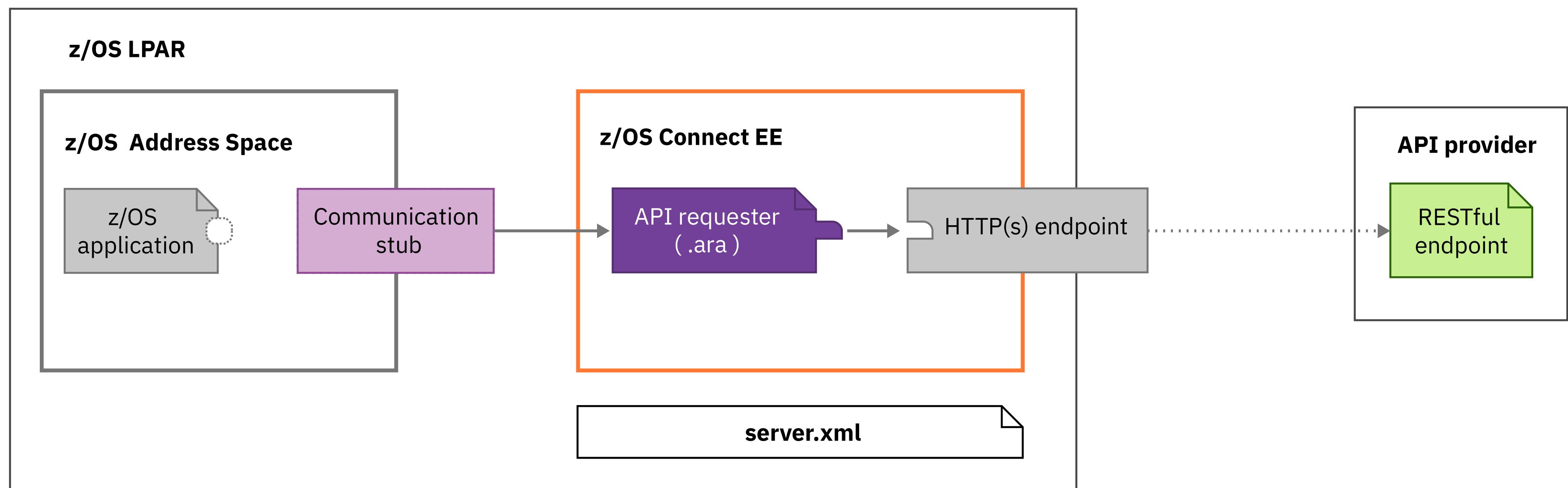
Results:

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



Steps to calling an remote API

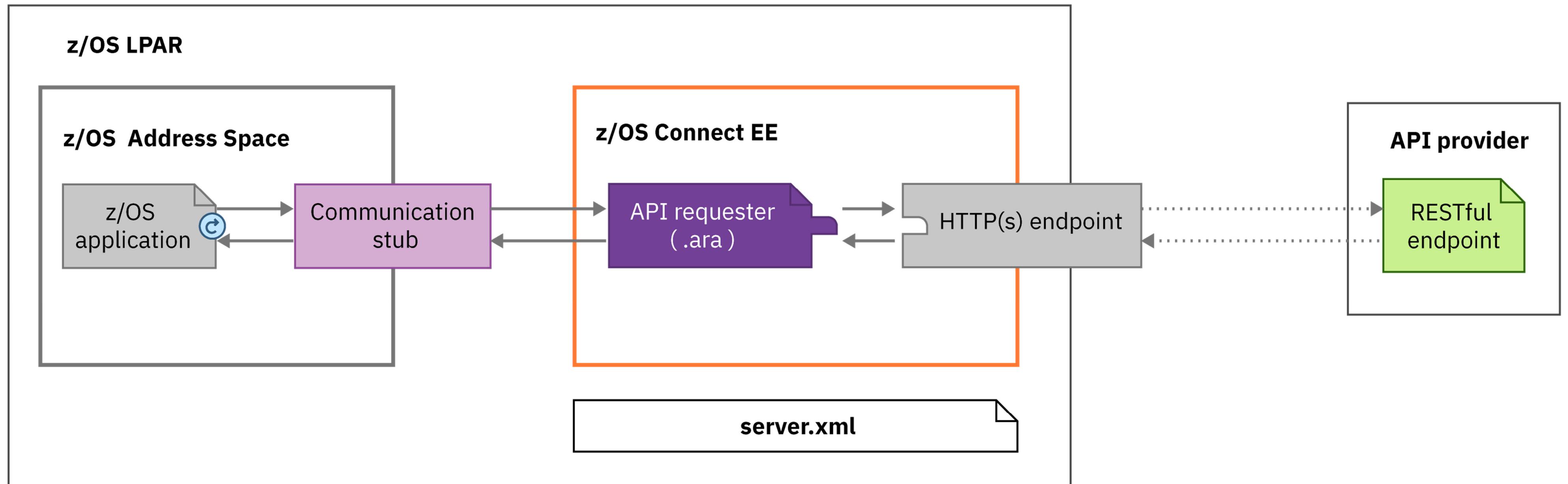
Configure HTTP(S) endpoint configuration element



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

ibm.biz/zosconnect-configure-endpoint-connection

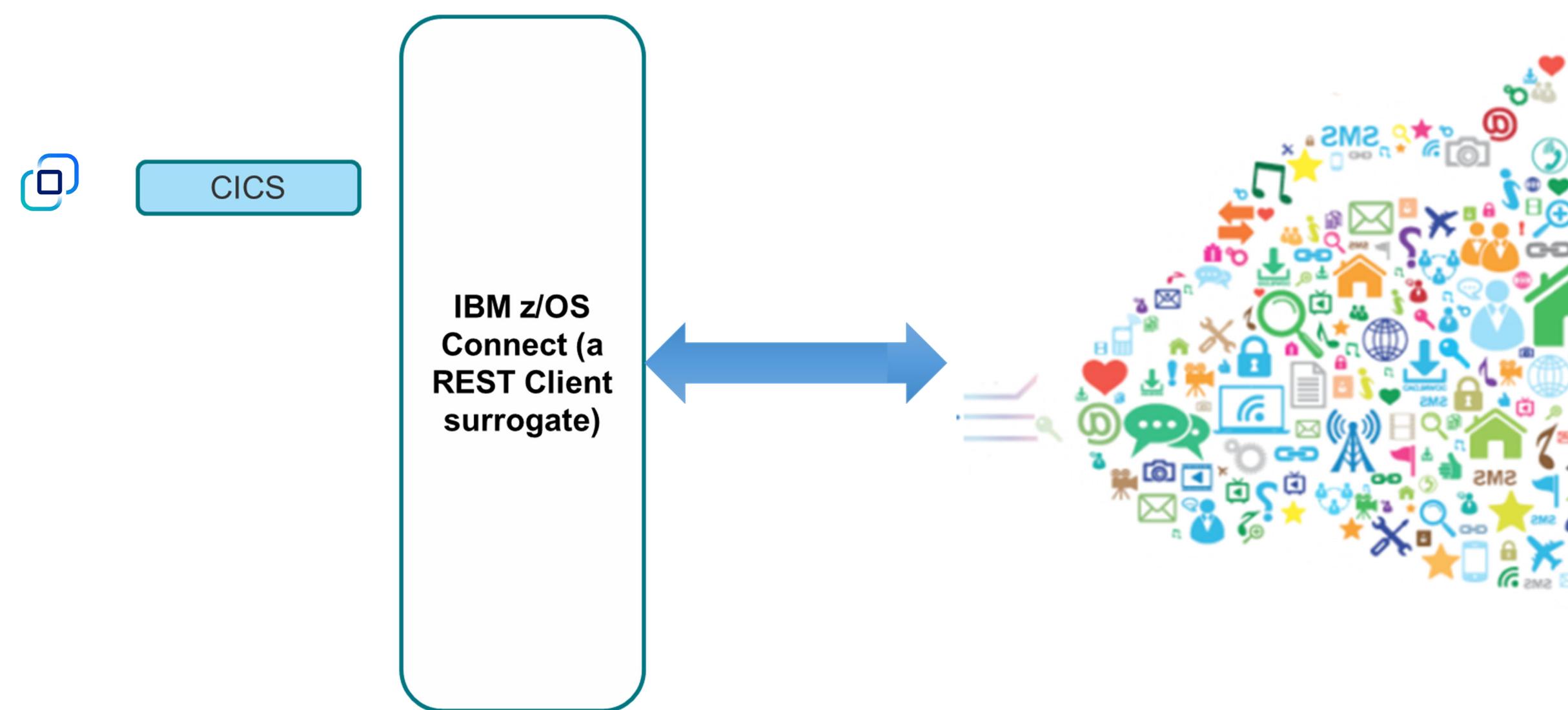
Results - A complete solution for calling a remote OpenAPI 2 API





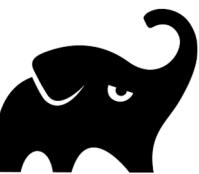
Developing z/OS Connect API Requesters

For APIs defined using an OpenAPI 3 specification

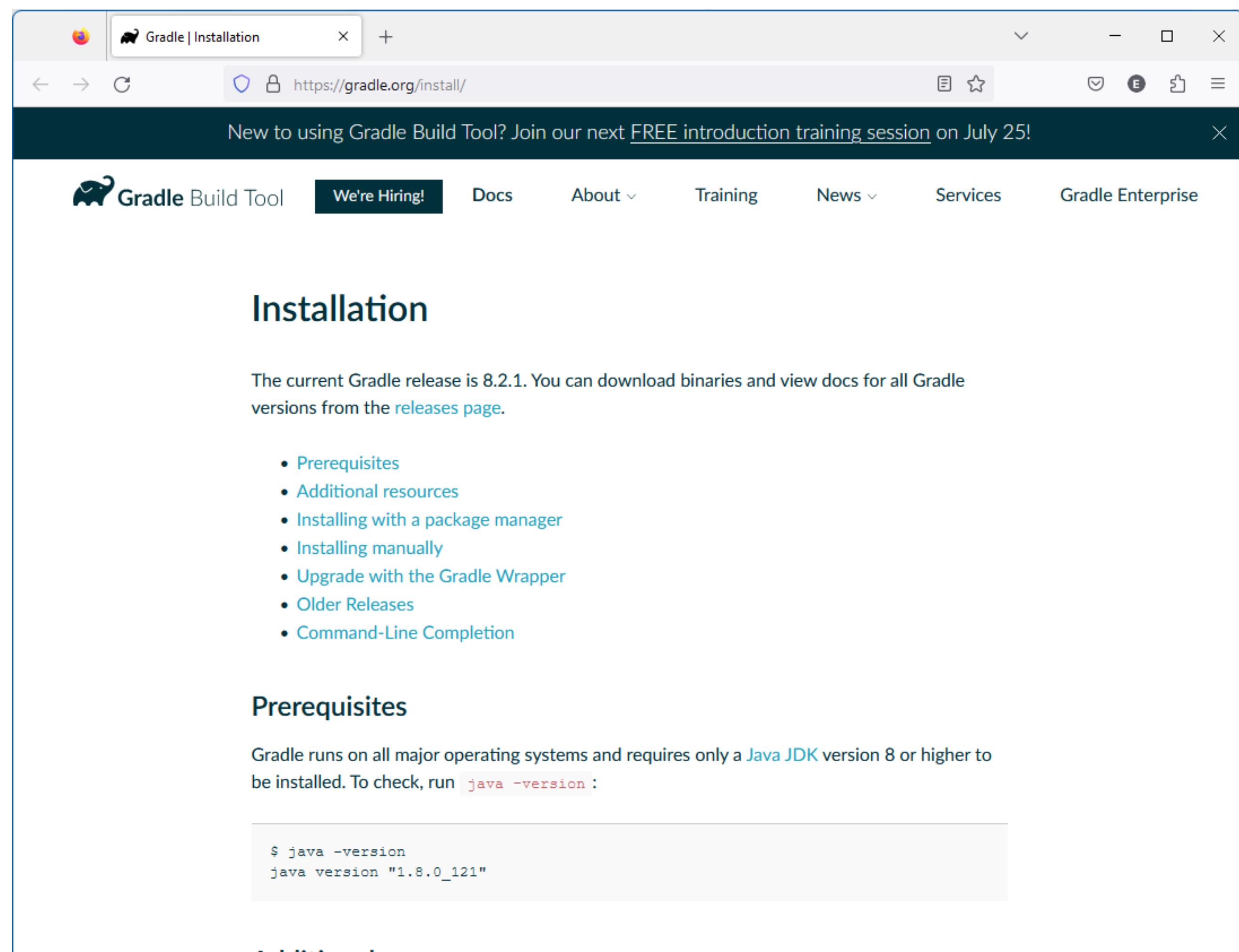


Gradle Build Tool – Installation -

<https://gradle.org/install/>



- Command line tool that builds the API requester deployable runtime artifact and the COBOL copybooks to invoke it
- It is the OpenAPI 3 equivalent of the z/OS Connect build toolkit for OpenAPI 2
- Not shipped as part of the SMP/E product
 - Hosted on Maven Central and Gradle Plugin Portal
 - Anyone can use it without purchasing z/OS Connect Unlimited
 - Uses “IBM International License Agreement for Non-Warranted Programs” [IBM Terms](#)



The screenshot shows a web browser displaying the Gradle Installation page at <https://gradle.org/install/>. The page features a dark header with the Gradle logo and navigation links for We're Hiring!, Docs, About, Training, News, Services, and Gradle Enterprise. A banner at the top encourages users to join a free introduction training session on July 25. The main content area is titled "Installation" and notes that the current release is 8.2.1. It provides links to Prerequisites, Additional resources, Installing with a package manager, Installing manually, Upgrade with the Gradle Wrapper, Older Releases, and Command-Line Completion. Below this, a "Prerequisites" section states that Gradle runs on all major operating systems and requires Java JDK version 8 or higher, with a command-line example showing "java -version" outputting "java version "1.8.0_121"".



Building z/OS Connect APIs with Gradle -

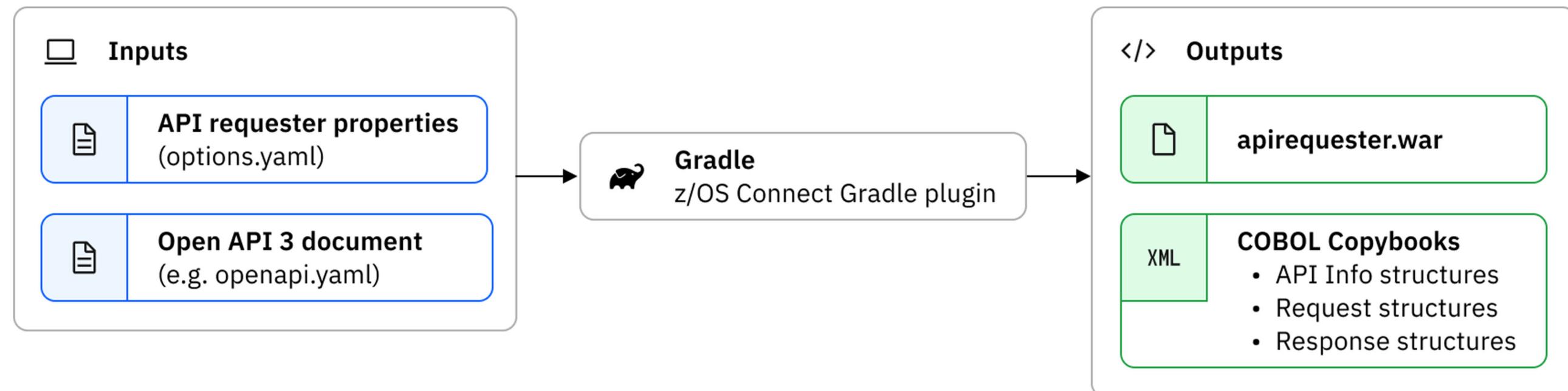
<https://www.ibm.com/docs/en/zos-connect/zos-connect/3.0?topic=building-zos-connect-apis-gradle>

The screenshot shows a web browser displaying the IBM z/OS Connect documentation. The URL in the address bar is <https://www.ibm.com/docs/en/zos-connect/zos-connect/3.0?topic=building-zos-connect-apis-gradle>. The page title is "Building z/OS Connect APIs with Gradle". The left sidebar shows navigation links for "IBM z/OS Connect" version 3.0, including "Building z/OS Connect APIs with Gradle", "Calling APIs from z/OS applications", "Installing z/OS Connect server", "Configuring IBM z/OS Connect server", "Securing IBM z/OS Connect resources", "IBM z/OS Connect DevOps", "IBM z/OS Connect high availability", "z/OS Connect monitoring", and "Troubleshooting". The main content area includes a breadcrumb trail: All products / IBM z/OS Connect / IBM z/OS Connect (OpenAPI 3) / 3.0 / Building z/OS Connect APIs with Gradle. It also features a "Was this topic helpful?" poll with "Yes" and "No" buttons, a search bar, and a copyright notice at the bottom.



Use the Gradle plug-in to generate the artifacts

```
cscvinc.yaml
openapi: 3.0.0
info:
  description: "CICS Employee Sample VSAM Application"
  version: 1.0.0
  title: Employee
x-ibm-zcon-roles-allowed:
- Manager
paths:
  /employee:
    post:
      tags:
        - Employee
      operationId: postEmployeeInsertService
      parameters:
        - name: Authorization
          in: header
          required: false
          schema:
            type: string
      requestBody:
        content:
          application/json:
            schema:
              $ref: "#/components/schemas/postEmployeeInsertService_request"
            description: request body
            required: true
      responses:
        "200":
          description: OK
Ln 12, Col 19 100% Windows (CRLF) UTF-8
```



Gradle plug-in properties and options[#]

```
apiKeyMaxLength=255
characterVarying=NO
Operations=getEmployeeSelectService
language=COBOL
connectionRef=cscvincAPI
requesterPrefix=csc
```

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

Gradle plug-in options

options.yaml contains preferences for how

- The copybooks will be generated
- The API requester runtime will behave

language: cobol

- Only required property
- **apiRequesterLayout** task adds by default

Override context root and war filename

dgglwlrqdoSurshuwlv1}h
dslQdph

dslNh|Pd{Ohqjwk

dslNh|SdupQdphLqFrrn1h

dslNh|SdupQdphLqKhdghu

dslNh|SdupQdphLqTxhu|

fkdudfwhuPxowlsolhu

fkdudfwhuYdu|lqj

fkdudfwhuYdu|lqjOlplw

fkdudfwhuZklwhVsdfh

frqghfwlrqUhi

gdwdWuxqfdwlrq

gdwhWlphIrupdw

ghidxowFkdudfwhuPd{Ohqjwk

ghidxowUhtxhvPhgldW|sh

ghidxowUhvsrqvhPhgldW|sh

ghidxowiudfwlrqG1jlwv

jhghudwhgFrghSdjh

lqolqhPd{RffxuvOlplw

odqjxdjh

qdphWuxqfdwlrq

rshudwlrv

uhtxhvPhgldW|sh

uhtxhvwhuSuhil{ }

uhvsrqvhPhgldW|sh

uxqwlphFrghSdjh

zlgFrps6

New for OpenAPI 3

API endpoint connection

Only **required** property

At what point to use data areas

Filter out the desired operations from the OpenAPI 3 document

When there is >1 media type, specify which to use

[The API requester Gradle plug-in properties and options - IBM Documentation](#)



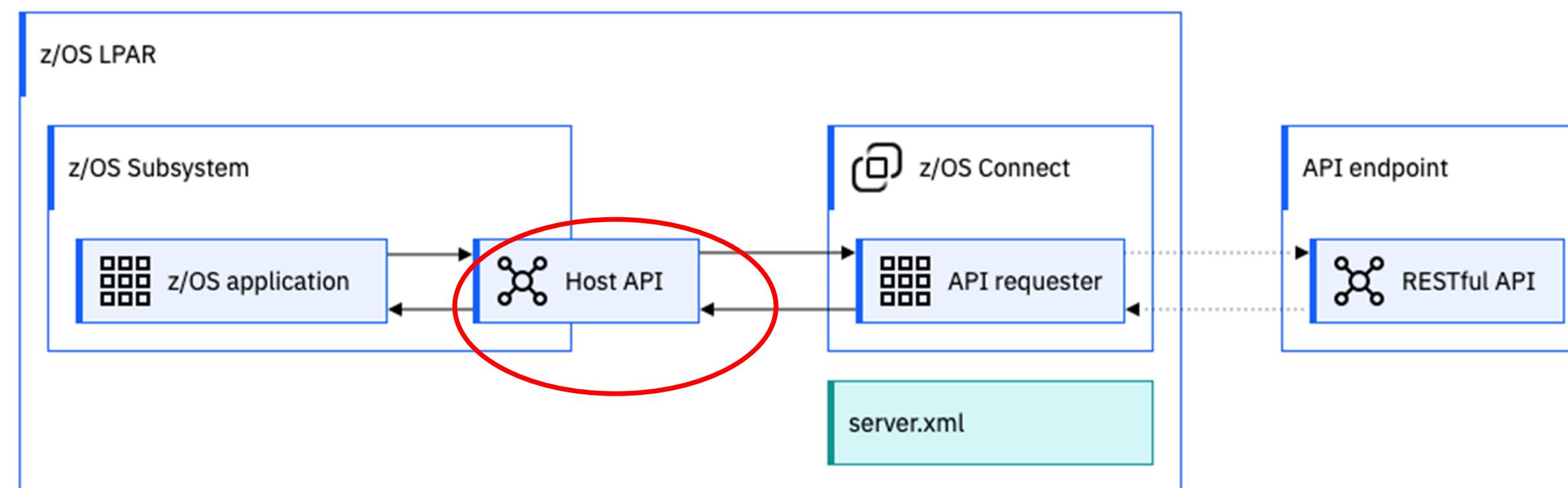
Summary of API requester enhancements for OpenAPI 3

- **Data Areas** - When the OpenAPI 3 document has large unbounded arrays, Data Areas can be used to dynamically allocate the working storage that is required to access them. This reduces the memory requirements for the COBOL program and prevents large amounts of null data being transferred between the COBOL program and z/OS Connect.
- **Multiple Response Codes** - Operations defined in the OpenAPI 3 document that return multiple response codes are now supported. At runtime, the COBOL program can detect which HTTP response code was returned by the API endpoint and then access the Data Area that is associated with the response code.
- **SAF-based security** - API requester authorization is configured by using SAF EJBROLE profiles to define the SAF users and groups that have the authority to invoke the API requester.
- **Operation Filtering** - The API requester Gradle plug-in allows operation filtering so that only a subset of the operations in an OpenAPI 3 document is exposed to the calling COBOL program.
- **API endpoint response message change toleration** - An API requester WAR file does not need to be rebuilt if the remote API is updated to return more data fields. If fields are removed, the API requester WAR file must be rebuilt.
- **Multiple API requesters for an endpoint** - Multiple API requesters can be deployed to a single z/OS Connect server for a single API endpoint.



What are the major differences with OpenAPI 3 API requester support?

The major changes is that the applications uses multiple calls to a z/OS Connect server rather than a single call to a OpenAPI 2 communication stub. This change was implemented to address storage issues encountered with the OpenAPI 2 solution, to add the ability to handle multiple response messages and to add extended support for new OpenAP3 features. The APIs added for OpenAPI 3 support are collectively known as the **Host API** callable interface.



The Host API has an extended set of callable verbs to support the richer set of features available in the OpenAPI 3.0 specification. For example, the Host API supports receiving **multiple mapped HTTP response codes** and **dynamically sized arrays**.

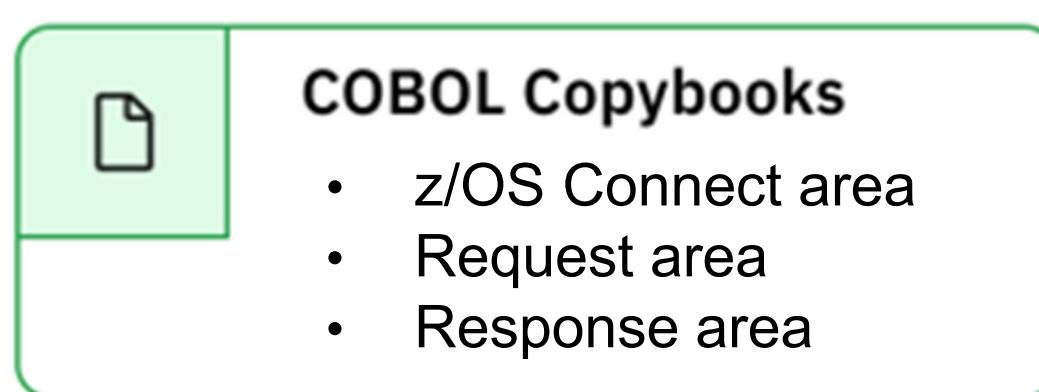
- New function added support for an application to program to be able to handle different response message payloads depending on whether the HTTP status code is 200 (OK), 201 (CREATED), or 404 (NOTFOUND) for example. Each of the response payloads documented in the API's OpenAPI 3.0 definition are mapped to a different COBOL language structure.
- Also, the storage required for arrays in a response messages is no longer is contained in the application's program working storage section. Rather than reserving static storage to hold the entire contents of all arrays, the storage for a single entry of an array is defined in the application program's LINKAGE SECTION. Now the program processes individual array entries sequentially one at a time.
- To support these new features, a **data area** is used. Each dynamic array and each return response code message has its own data area. The data areas in the LINKAGE SECTION of program are updated by the HOST APIs and this provides access to the contents of response message and individual elements of an array.



COBOL copybooks

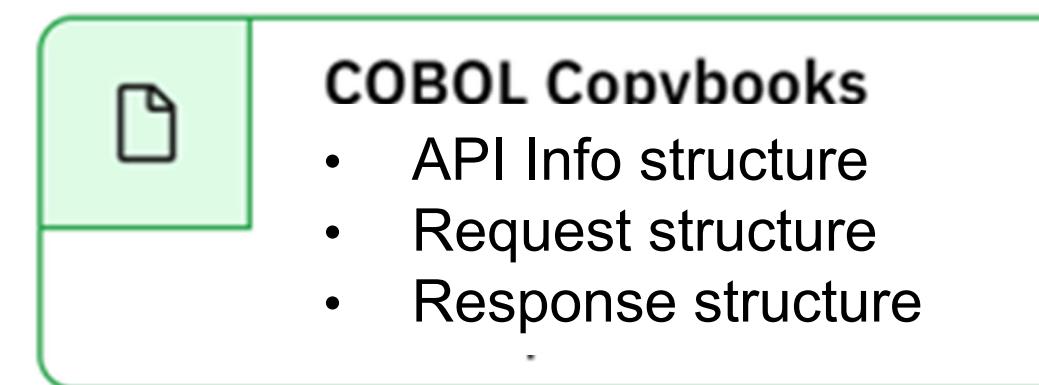
z/OS Connect provides two copybooks that are required to interface with the Host API and tooling to generate application specific copybooks

1. z/OS Connect provided static structures



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

2. Generated dynamic structures



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



BAQHCONC copy book

Product provided constants for convenience

- Host API entry point names for dynamic calling
- Host API Request specific parameters – are relevant to invoking the API
- Host API z/OS Connect parameters – are relevant to the connection to the z/OS API requester server.

```
WG31 - 3270
File Edit Settings View Communication Actions Window Help
File Edit Edit_Settings Menu Utilities Compilers Test Help
EDIT      ZCEE30.SBAQCDB(BAQHCONC) - 01.00          Columns 00001 00072
Command ==>                                         Scroll ==> PAGE
***** **** * ***** Top of Data ***** **** * ***** 11 Line(s) not Displayed
-----+
000012    * Host API entry point names
000013    77 BAQ-INIT-NAME          PIC X(8) VALUE 'BAQINIT'.
000014    77 BAQ-EXEC-NAME          PIC X(8) VALUE 'BAQEXEC'.
000015    77 BAQ-GETN-NAME          PIC X(8) VALUE 'BAQGETN'.
000016    77 BAQ-PUTN-NAME          PIC X(8) VALUE 'BAQPUTN'.
000017    77 BAQ-FREE-NAME          PIC X(8) VALUE 'BAQFREE'.
000018    77 BAQ-TERM-NAME          PIC X(8) VALUE 'BAQTERM'.
000019
000020
000021    * Host API Request parameter names
000022    77 BAQR-OAUTH-USERNAME    PIC X(22)
000023    VALUE 'BAQHAPI-oAuth-Username'.
000024    77 BAQR-OAUTH-PASSWORD    PIC X(22)
000025    VALUE 'BAQHAPI-oAuth-Password'.
000026    77 BAQR-OAUTH-SCOPE      PIC X(19)
000027    VALUE 'BAQHAPI-oAuth-Scope'.
000028    77 BAQR-OAUTH-CLIENT-ID   PIC X(22)
000029    VALUE 'BAQHAPI-oAuth-ClientId'.
000030    77 BAQR-OAUTH-CLIENT-SECRET PIC X(26)
000031    VALUE 'BAQHAPI-oAuth-ClientSecret'.
000032
000033
000043    * Host API ZCON parameter names
000044    77 BAQZ-TRACE-VERBOSE    PIC X(21)
000045    VALUE 'BAQHAPI-Trace-Verbose'.
000046    77 BAQZ-SERVER-URIMAP    PIC X(21)
000047    VALUE 'BAQHAPI-Server-URIMAP'.
000048    77 BAQZ-SERVER-HOST      PIC X(19)
000049    VALUE 'BAQHAPI-Server-Host'.
000050    77 BAQZ-SERVER-PORT      PIC X(19)
000051    VALUE 'BAQHAPI-Server-Port'.
000052    77 BAQZ-SERVER-TIMEOUT   PIC X(22)
000053    VALUE 'BAQHAPI-Server-Timeout'.
000054    77 BAQZ-SERVER-USERNAME   PIC X(23)
000055    VALUE 'BAQHAPI-Server-Username'.
000056    77 BAQZ-SERVER-PASSWORD   PIC X(23)
000057    VALUE 'BAQHAPI-Server-Password'.
***** **** * ***** Bottom of Data ***** **** * ***** 11 Line(s) not Displayed
-----+
MA A
Connected to remote server/host wg31 using lu/pool TCP00136 Adobe PDF on Documents\*.pdf
06/002
```



BAQHAREC copy book

BAQ-ZCONNECT-AREA

Input & Output interface for all Host API verbs

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

```
WG31 - 3270
File Edit Settings View Communication Actions Window Help
File Edit Edit_Settings Menu Utilities Compilers Test Help
EDIT      ZCEE30.SBAQCOB(BAQHAREC) - 01.00          Columns 00001 00072
Command ==> -                                     Scroll ==> PAGE
***** **** * ***** Top of Data ***** **** ****
000001   ****
000002   *
000003   * PID 5655-CES
000004   *
000005   * Copyright IBM Corp. 2023
000006   *
000007   ****
000008   * This file contains the language structures required by COBOL
000009   * programs to work with the API requester Host API.
000010   ****
000011 01 BAQ-ZCONNECT-AREA.
000012 03 BAQ-ZCON-AREA-EYE          PIC X(4) VALUE 'BAQZ'.
000013 03 BAQ-ZCON-AREA-LENGTH       PIC 9(8) COMP-5 VALUE 4648.
000014 03 BAQ-ZCON-AREA-VERSION      PIC 9(8) COMP-5 VALUE 1.
000015 03 BAQ-ZCON-RESERVED-01      PIC 9(8) COMP-5 VALUE 0.
000016 03 BAQ-ZCON-PARM-INIT        VALUE LOW-VALUES.
000017 05 BAQ-ZCON-RESERVED-02        PIC X(3584).
000018 03 BAQ-ZCON-PARAMETERS       REDEFINES BAQ-ZCON-PARM-INIT.
000019 05 BAQ-ZCON-PARMS           OCCURS 64.
000020 07 BAQ-ZCON-PARM-NAME         PIC X(48).
000021 07 BAQ-ZCON-PARM-ADDRESS       USAGE POINTER.
000022 07 BAQ-ZCON-PARM-LENGTH       PIC 9(9) BINARY.
000023 03 BAQ-ZCON-RETURN-CODES.
000024 05 BAQ-ZCON-COMPLETION-CODE  PIC 9(8) COMP-5 VALUE 0.
000025 88 BAQ-SUCCESS             VALUE 0.
000026 88 BAQ-WARNING             VALUE 4.
000027 88 BAQ-ERROR               VALUE 8.
000028 88 BAQ-SEVERE              VALUE 12.
000029 88 BAQ-CRITICAL            VALUE 16.
000030 05 BAQ-ZCON-REASON-CODE     PIC 9(8) COMP-5 VALUE 0.
000031 05 BAQ-ZCON-SERVICE-ID      PIC 9(8) COMP-5 VALUE 0.
000032 05 BAQ-ZCON-SERVICE-CODE    PIC 9(8) COMP-5 VALUE 0.
000033 05 BAQ-ZCON-RESERVED-03      PIC 9(8) COMP-5 VALUE 0.
000034 03 BAQ-ZCON-RETURN-MESSAGE-LEN PIC 9(8) COMP-5 VALUE 0.
000035 03 BAQ-ZCON-RETURN-MESSAGE  PIC X(1024).
000036
000037 01 BAQ-REQUEST-AREA.
000038 03 BAQ-REQ-AREA-EYE          PIC X(4) VALUE 'BAQR'.
04/015
Connected to remote server/host wg31 using lu/pool TCP00136 ↗ Adobe PDF on Documents\*.pdf
```



BAQHAREC - BAQ-ZCONNECT_AREA

The BAQ-ZCON-PARMS array

Providing credentials for basic authentication

```

File Edit Settings View Communication Actions Window Help
File Edit Edit_Settings Menu Utilities Compilers Test Help
EDIT ZCEE30.SBAQCDB(BAQHAREC) - 01.00 Columns 00001 00072
Command ==> **** Top of Data *****
000001 ****
000002 *
000003 * PID 5655-CES
000004 *
000005 * Copyright IBM Corp. 2023
000006 *
000007 **** This file contains the language structures required by COBOL
000008 * programs to work with the API requester Host API.
000009 ****
000010 01 BAQ-ZCONNECT-AREA.
000011    03 BAQ-ZCON-AREA-EYE          PIC X(4) VALUE 'BAQZ'.
000012    03 BAQ-ZCON-AREA-LENGTH        PIC 9(8) COMP-5 VALUE 4648.
000013    03 BAQ-ZCON-AREA-VERSION       PIC 9(8) COMP-5 VALUE 1.
000014    03 BAQ-ZCON-RESERVED-01       PIC 9(8) COMP-5 VALUE 0.
000015    03 BAQ-ZCON-PARM-INIT         VALUE LOW-VALUES.
000016    03 BAQ-ZCON-PARM-RESERVED-02  PIC X(3584).
000017    03 BAQ-ZCON-PARAMETERS       REDEFINES BAQ-ZCON-PARM-INIT.
000018    05 BAQ-ZCON-PARMS          OCCURS 64.
000019      07 BAQ-ZCON-PARM-NAME        PIC X(48).
000020      07 BAQ-ZCON-PARM-ADDRESS      USAGE POINTER.
000021      07 BAQ-ZCON-PARM-LENGTH      PIC 9(9) BINARY.
000022    03 BAQ-ZCON-RETURN-CODES.
000023      05 BAQ-ZCON-COMPLETION-CODE  PIC 9(8) COMP-5 VALUE 0.
000024      88 BAQ-SUCCESS            VALUE 0.
000025      88 BAQ-WARNING           VALUE 4.
000026      88 BAQ-ERROR              VALUE 8.
000027      88 BAQ-SEVERE             VALUE 12.
000028      88 BAQ-CRITICAL           VALUE 16.
000029    05 BAQ-ZCON-REASON-CODE     PIC 9(8) COMP-5 VALUE 0.
000030    05 BAQ-ZCON-SERVICE-ID      PIC 9(8) COMP-5 VALUE 0.
000031    05 BAQ-ZCON-SERVICE-CODE     PIC 9(8) COMP-5 VALUE 0.
000032    05 BAQ-ZCON-RESERVED-03       PIC 9(8) COMP-5 VALUE 0.
000033    03 BAQ-ZCON-RETURN-MESSAGE-LEN  PIC 9(8) COMP-5 VALUE 0.
000034    03 BAQ-ZCON-RETURN-MESSAGE   PIC X(1024).
000035 01 BAQ-REQUEST-AREA.
000036    03 BAQ-REQ-AREA-EYE        PIC X(4) VALUE 'BAQR'.
000037
000038
04/015
MA A
Connected to remote server/host wg31 using lu/pool TCP00136 & Adobe PDF on Documents\*.pdf

```

```

File Edit Settings View Communication Actions Window Help
File Edit Edit_Settings Menu Utilities Compilers Test Help
EDIT JOHNSON.ZCEE.SOURCE(BAQZUSER) - 01.00 Columns 00001 00072
Command ==> **** Top of Data *****
000001 * User credentials for basic authentication
000002 01 MY-USER PIC X(10) VALUE 'myUsername'.
000003 01 MY-PSWD PIC X(10) VALUE 'myPassword'.
000004 PROCEDURE DIVISION.
000005 ...
000006 * Set user credentials
000007 MOVE BAQZ-SERVER-USERNAME
000008   TO BAQ-ZCON-PARM-NAME OF BAQ-ZCON-PARMS(1)
000009 SET BAQ-ZCON-PARM-ADDRESS OF BAQ-ZCON-PARMS(1)
000010   TO ADDRESS OF MY-USER
000011 MOVE LENGTH OF MY-USER
000012   TO BAQ-ZCON-PARM-LENGTH OF BAQ-ZCON-PARMS(1)
000013 MOVE BAQZ-SERVER-PASSWORD
000014   TO BAQ-ZCON-PARM-NAME OF BAQ-ZCON-PARMS(2)
000015 SET BAQ-ZCON-PARM-ADDRESS OF BAQ-ZCON-PARMS(2)
000016   TO ADDRESS OF MY-PSWD
000017 MOVE LENGTH OF MY-PSWD
000018   TO BAQ-ZCON-PARM-LENGTH OF BAQ-ZCON-PARMS(2).
000019 * Make the BAQINIT call
000020 **** Bottom of Data *****

```

Overriding the URIMAP value

```

File Edit Settings View Communication Actions Window Help
File Edit Edit_Settings Menu Utilities Compilers Test Help
EDIT JOHNSON.ZCEE.SOURCE(BAQZUSER) - 01.01 Columns 00001 00072
Command ==> **** Top of Data *****
000001 IDENTIFICATION DIVISION.
000002 WORKING-STORAGE SECTION.
000003 01 WS-DYNAMIC-URIMAP PIC X(8) VALUE 'MYURIMAP'.
000004 PROCEDURE DIVISION.
000005 ...
000006 ...
000007 ***
000008 MOVE BAQZ-SERVER-URIMAP TO
000009   BAQ-ZCON-PARM-NAME OF BAQ-ZCON-PARMS(1).
000010 SET BAQ-ZCON-PARM-ADDRESS OF BAQ-ZCON-PARMS(1) TO
000011   ADDRESS OF WS-DYNAMIC-URIMAP.
000012 MOVE LENGTH OF WS-DYNAMIC-URIMAP TO
000013   BAQ-ZCON-PARM-LENGTH OF BAQ-ZCON-PARMS(1).
000014 ...
000015 **** Bottom of Data *****

```



BAQHAREC – BAQ-REQUEST-AREA/BAQ-RESPONSE-AREA

BAQ-REQUEST-AREA

Input to the [BAQEXEC](#) for an API call

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

BAQ-RESPONSE-AREA

Output from the [BAQEXEC](#) call

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

```
WG31 - 3270
File Edit Settings View Communication Actions Window Help
File Edit Edit_Settings Menu Utilities Compilers Test Help
EDIT ZCEE30.SBAQC0B(BAQHAREC) - 01.00
Command ==> - 01 BAQ-REQUEST-AREA.
000037 03 BAQ-REQ-AREA-EYE
000038 03 BAQ-REQ-AREA-LENGTH
000039 03 BAQ-REQ-AREA-VERSION
000040 03 BAQ-REQ-RESERVED-01
000041 03 BAQ-REQ-BASE-ADDRESS
000042 03 BAQ-REQ-BASE-LENGTH
000043 03 BAQ-REQ-PARM-INIT
000044 05 BAQ-REQ-RESERVED-02
000045 03 BAQ-REQ-PARAMETERS
000046 05 BAQ-REQ-PARMS
000047 07 BAQ-REQ-PARM-NAME
000048 07 BAQ-REQ-PARM-ADDRESS
000049 07 BAQ-REQ-PARM-LENGTH
000050
000051
000052 01 BAQ-RESPONSE-AREA.
000053 03 BAQ-RESP-AREA-EYE
000054 03 BAQ-RESP-AREA-LENGTH
000055 03 BAQ-RESP-AREA-VERSION
000056 03 BAQ-RESP-RESERVED-01
000057 03 BAQ-RESP-BASE-ADDRESS
000058 03 BAQ-RESP-BASE-LENGTH
000059 03 BAQ-RESP-RESERVED-02
000060 03 BAQ-RESP-STATUS-CODE
000061 03 BAQ-RESP-STATUS-MESSAGE-LEN
000062 03 BAQ-RESP-STATUS-MESSAGE
***** **** Bottom of Data ****
PIC X(4) VALUE 'BAQR'.
PIC 9(8) COMP-5 VALUE 3608.
PIC 9(8) COMP-5 VALUE 1.
PIC 9(8) COMP-5 VALUE 0.
USAGE POINTER.
PIC 9(8) BINARY.
VALUE LOW-VALUES.
PIC X(3584).
REDEFINES BAQ-REQ-PARM-INIT.
OCCURS 64.
PIC X(48).
USAGE POINTER.
PIC 9(9) BINARY.

PIC X(4) VALUE 'BAQP'.
PIC 9(8) COMP-5 VALUE 1060.
PIC 9(8) COMP-5 VALUE 1.
PIC 9(8) COMP-5 VALUE 0.
USAGE POINTER.
PIC 9(8) COMP-5 VALUE 0.
PIC X(1024).

04/015
Connected to remote server/host wg31 using lu/pool TCP00136 z Adobe PDF on Documents\*.pdf
```



The generated API information (“I”) copybook

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

```
EDIT      JOHNSON.RBK02I01.CPY
Command ==> -
*****
***** **** * ++++++ ++++++ ++++++ ++++++ ++++++ ++++++ ++++++ ++++++
000001   * ++++++ ++++++ ++++++ ++++++ ++++++ ++++++ ++++++ ++++++
000002   * This file contains the generated API information structure
000003   * which is passed to the Host API via the BAQEXEC call.
000004   * ++++++ ++++++ ++++++ ++++++ ++++++ ++++++ ++++++ ++++++
000005     01 BAQ-API-INFO-RBK02I01.
000006       03 BAQ-API-INFO-EYE
000007         VALUE 'BAQA'.
000008       03 BAQ-API-INFO-LENGTH
000009         VALUE 1052.
000010       03 BAQ-API-INFO-VERSION
000011         VALUE 1.
000012       03 BAQ-API-INFO-RESERVED01
000013         VALUE 0.
000014       03 BAQ-API-NAME
000015         VALUE 'RedbookApi'.
000016       03 BAQ-API-NAME-LEN
000017         VALUE 10.
000018       03 BAQ-API-PATH
000019         VALUE '%2Fredbooks'.
000020       03 BAQ-API-PATH-LEN
000021         VALUE 11.
000022       03 BAQ-API-METHOD
000023         VALUE 'GET'.
000024       03 BAQ-API-METHOD-LEN
000025         VALUE 3.
000026       03 BAQ-API-OPERATION
000027         VALUE 'getAllRedbooks'.
000028       03 BAQ-API-OPERATION-LEN
000029         VALUE 14.
000030
*****
***** **** * ++++++ ++++++ ++++++ ++++++ ++++++ ++++++ ++++++ ++++++
Bottom of Data **** * ++++++ ++++++ ++++++ ++++++ ++++++ ++++++ ++++++ ++++++
```

MA A
Connected to remote server/host wg31 using lu/pool TCP00110 and port 23 04/015
Adobe PDF on Documents*.pdf



Generated Request message (“Q”) copybook

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

INITIALIZE BAQBASE-RBK02Q01.

```
WG31 - 3270
File Edit Settings View Communication Actions Window Help
File Edit Edit_Settings Menu Utilities Compilers Test Help
EDIT      JOHNSON.RBK02Q01.CPY
Command ==> * ++++++*+++++*+++++*+++++*+++++*+++++*+++++*+
000001    * This file contains the generated language structure(s) for
000002    * request JSON schema 'getAllRedbooks_request.json'.
000003    * This structure was generated using 'DFHJS2LS' at mapping level
000004    * '5.0'.
000005    *
000006    *
000007    *
000008    * 01 BAQBASE-RBK02Q01.
000009    * 03 requestQueryParameters.
000010    *
000011    *
000012    * JSON schema keyword 'requestQueryParameters->author' is
000013    * optional. The existence of the field is indicated by field
000014    * 'Xauthor-existence'.
000015    * 06 Xauthor-existence          PIC S9(9) COMP-5 SYNC.
000016    *
000017    *
000018    * 06 Xauthor.
000019    *
000020    * Comments for field 'Xauthor2':
000021    * This field represents the value of JSON schema keyword
000022    * 'requestQueryParameters->author'.
000023    * JSON schema type: 'string'.
000024    * JSON schema keyword 'minLength' value: '0'.
000025    * JSON schema keyword 'maxLength' value: '40'.
000026    * This field contains a varying length array of characters or
000027    * binary data.
000028    * 09 Xauthor2-length          PIC S9999 COMP-5 SYNC.
000029    * 09 Xauthor2                PIC X(40).
000030    *
000031    *
000032    * ++++++*+++++*+++++*+++++*+++++*+++++*+++++*+
000033    *
000034    * 01 BAQBASE-RBK02Q01.
000035    * 03 requestQueryParameters.
000036    *
000037    * 06 Xauthor-existence        PIC S9(9) COMP-5 SYNC.
000038    *
000039    * 06 Xauthor.
```



Generated Response message (“P”) copybook

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

The screenshot shows a COBOL edit session titled "JOHNSON.RBK02P01.CPY". The code displays a hierarchical structure starting with a 01 level structure for BAQBASE, followed by multiple 03 level structures for response codes and their associated data areas. The code is color-coded for syntax highlighting, and the right margin shows column and scroll information.

```
EDIT      JOHNSON.RBK02P01.CPY
Command ==> -
000343
000344
000345
000346
000347
000348
000349
000350
000351
000352
000353
000354
000355
000356
000357
000358
000359
000360
000361
000362
000363
000364
000365
000366
000367
000368
000369
000370
000371
000372
000373
000374
000375
000376
000377
000378
000379
000380
000381

01 BAQBASE-RBK02P01.
  03 responseCode200-num
  03 responseCode200-dataarea
    PIC S9(9) COMP-5 SYNC.
    PIC X(16).

  03 responseCode404-existence
  03 responseCode404-dataarea
    PIC S9(9) COMP-5 SYNC.
    PIC X(16).

  01 RBK02P01-responseCode200.
    03 responseCode200.
      06 Xtitle-length
      06 Xtitle
        PIC S9999 COMP-5 SYNC.
        PIC X(80).

      06 authors-num
      06 authors-dataarea
        PIC S9(9) COMP-5 SYNC.
        PIC X(16).

      06 Xstatus-length
      06 Xstatus
        PIC S9999 COMP-5 SYNC.
        PIC X(9).
        PIC X(12).

      06 publicationDate-existence
        06 publicationDate.
          09 publicationDate2-length
          09 publicationDate2
            PIC S9999 COMP-5 SYNC.
            PIC X(32).

        06 documentType-existence
          06 documentType.
            09 documentType2-length
            09 documentType2
              PIC S9999 COMP-5 SYNC.
              PIC X(8).

        06 sizeMB-existence
          06 sizeMB
            PIC S9(9) COMP-5 SYNC.
            PIC 9(16)V9(2) COMP-3.

  04/015
```

Connected to remote server/host wg31 using lu/pool TCP00110 and port 23

Adobe PDF on Documents*.pdf



The generated response(R) copy book

- ❑ The Gradle build process creates a response message copybook for each of the API's operations. In each response message copybook, there is a level 01 COBOL structures for each operation's potential types of response status codes. The names of these structures are in the format **copybookName-reponseCode###** where ### is the API's defined response code, e.g., 200, 4XX, or Def (default).
- ❑ Also in each response message copybook is a level 01 COBOL structure named **BAQBASE-copybookName**. In this structure, there are two 03 level variables for each of the possible response messages, the names of these variables are in the format **responseCode###-dataarea** and **responseCode###-num or responseCode###-dataarea** and **responseCode###-existence**.
 - The **responseCode###-dataarea** variable contains a token that is used by the runtime to track and/or manage of the data areas holding the response results. The returned data areas are not in the program's working storage area but are addressable from the LINKAGE section.
 - If the JSON property was an *array*, then the variable name is appended with **-num** and the value of this variable provides the number of occurrences or array entries of this array, including 0. If the JSON variable was an *Object* type, then the variable name is appended with **-existence** and this variable contains either 0 or 1 to specify whether an object was returned in the response.
- ❑ Each **copybookName-reponseCode###** 01 level COBOL structure contains a variable for each JSON response property in the API's specification.
 - Some of the variables in these structure are the format **variableName-dataarea** and **variableName-num** or **variableName-existence**. The variable names with these extensions have the same usage as describe above.
 - String JSON properties that are not constrained by their minimum length equal to their maximum length are preceded by variables that contain the value that contain the actual length of the string. These variable names are in the format **variableName-length**.
 - Numeric fields appear as expected.
- ❑ Other 01 level COBOL structures are generated in the response copybook for each for each array that occurs in the response.



The z/OS Connect Host API Verbs

BAQINIT

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

BAQEXEC

- **EXEC**ute the call to the desired API taking request data as input and providing response data as output

BAQGETN

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

BAQFREE

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

BAQTERM

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



Host API Verbs – initialize(BAQINIT) & execute (BAQEXEC)

BAQ base structure in the response copy book

```
01 BAQBASE-RBK02P01.  
 03 responseCode200-num          PIC S9(9) COMP-5 SYNC.  
 03 responseCode200-dataarea     PIC X(16).  
  
 03 responseCode404-existence   PIC S9(9) COMP-5 SYNC.  
 03 responseCode404-dataarea     PIC X(16).
```

1. Initialize the Host API and fail if the initialization was not successful

```
Initialise the Host API  
CALL BAQ-INIT-NAME USING  
      BY REFERENCE BAQ-ZCONNECT-AREA.  
  
Exit if initialisation fails  
IF NOT BAQ-SUCCESS THEN GO TO EXIT-PROGRAM.
```

2. Ensure the request data is initialized and then call the API endpoint

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

3. If the call succeeded, address the response BAQBASE structure

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

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



Host API Verbs – get next array entry(BAQGETN)

Structure for HTTP 200 response

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

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

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

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

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

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

PERFORM WRITE-RESPONSE-MSG.



Host API Verbs – **free storage(BAQFREE)** & **terminate(BAQTERM)**

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

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

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

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



Host API Completion Codes

0000 - 1999

- z/OS Connect runtime messages

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

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

2000 - 2999

Host API messages

- More details on the next slide

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

3000 - 3999

- Host API running in CICS

- CICS specific messages

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



Host API Completion and Reason Codes

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

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

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

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



Configuration and Security Considerations



Resolving the endpoint's URL

An administrator configures the endpointConnection providing the host and port of the URL

CSC02I01

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

cscvinc.properties
connectionRef=cscvincAPI

Server Config
apiRequesterHTTPS.xml

Design Source

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

```
<zosconnect_authData id="mySAFAuth"
  user="USER1"
  password="user1" />
```

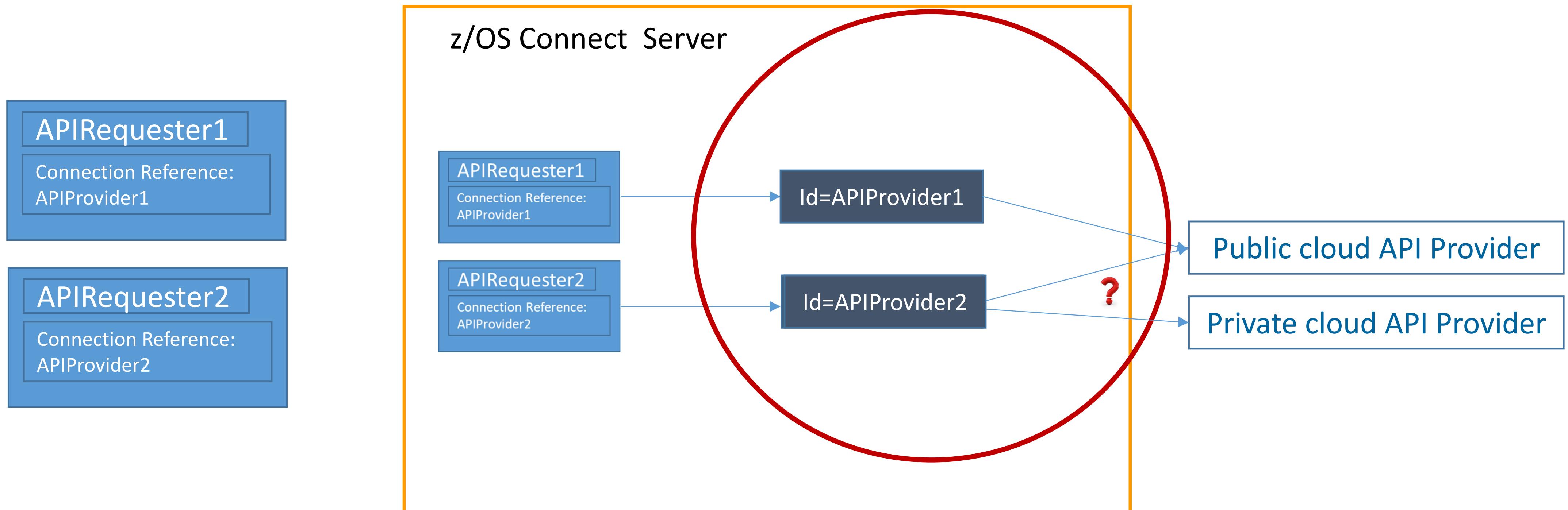
```
</server>
```

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



Tech-Tip: Use naming conventions for connection references

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



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

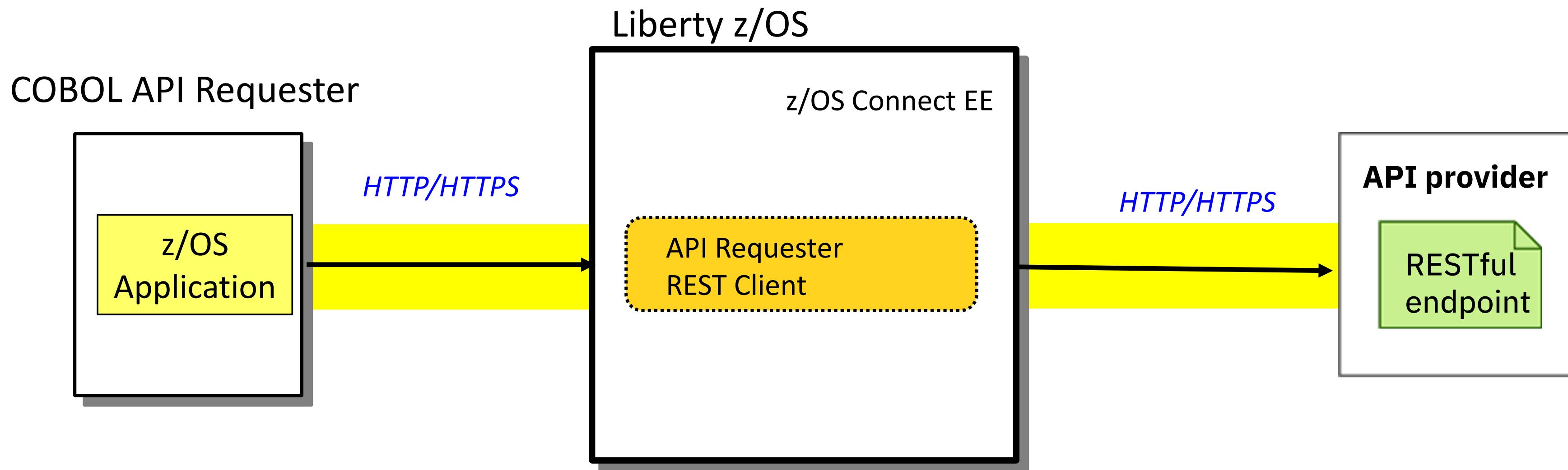
```
<webApplication location="${server.config.dir}/apps/cscvincapi-1.0.0.war">
  <appProperties> <property name="connectionRef" value="APIProvider2"/>
  </appProperties>
</webApplication>
```



**Now let's explore the security options for outbound
API Requester connections
and accessing remote resources**



End to end API requester to API Provider connection overview



MVS Batch, IMS HTTP and Db2 stored procedure connection details provided by:

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

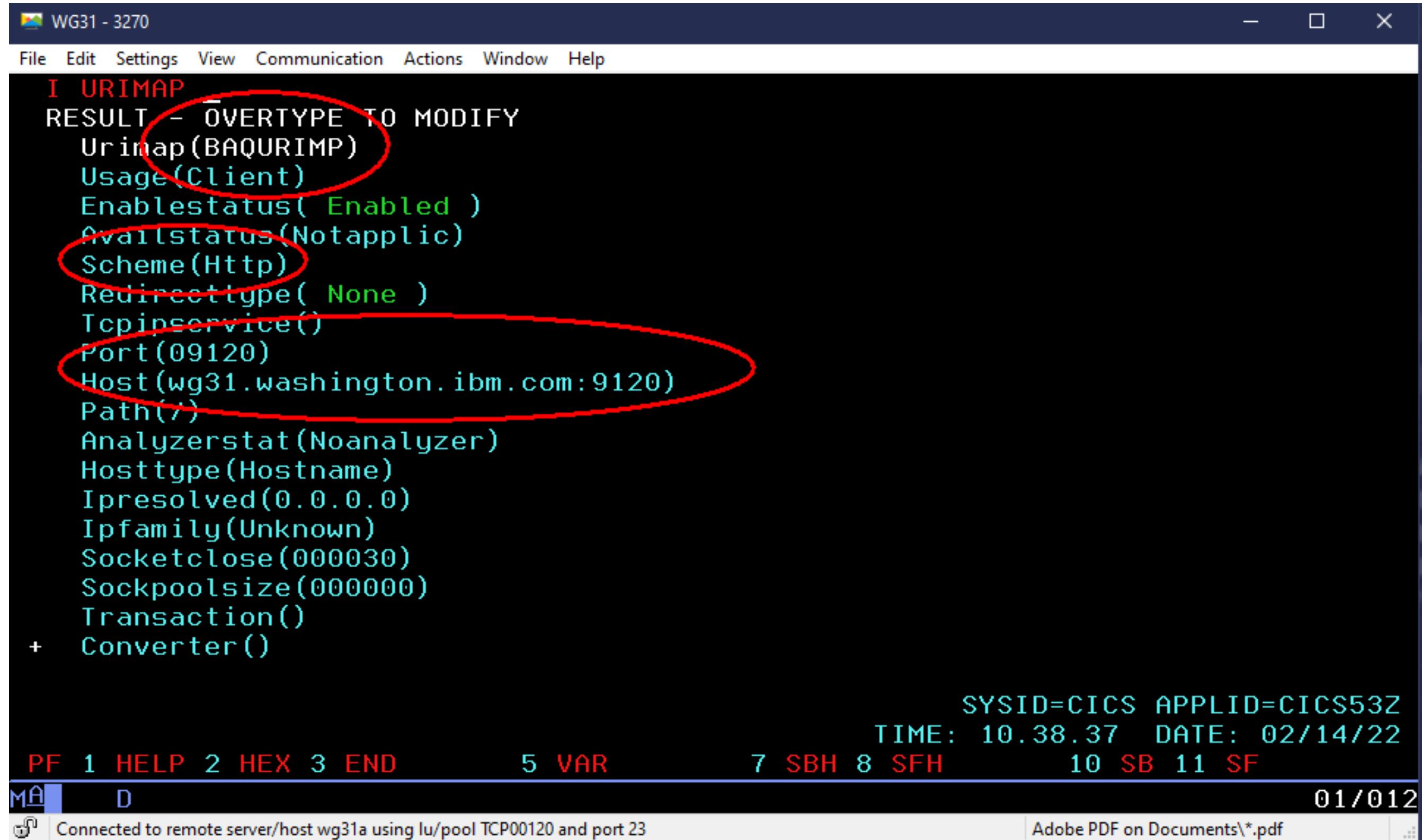
CICS HTTP connection details provided by:

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



Configuring connections to the z/OS API requester server

Default CICS URI MAP*



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

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

LE Environment Variables

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

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



Environment variables for non-CICS clients

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

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

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

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

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

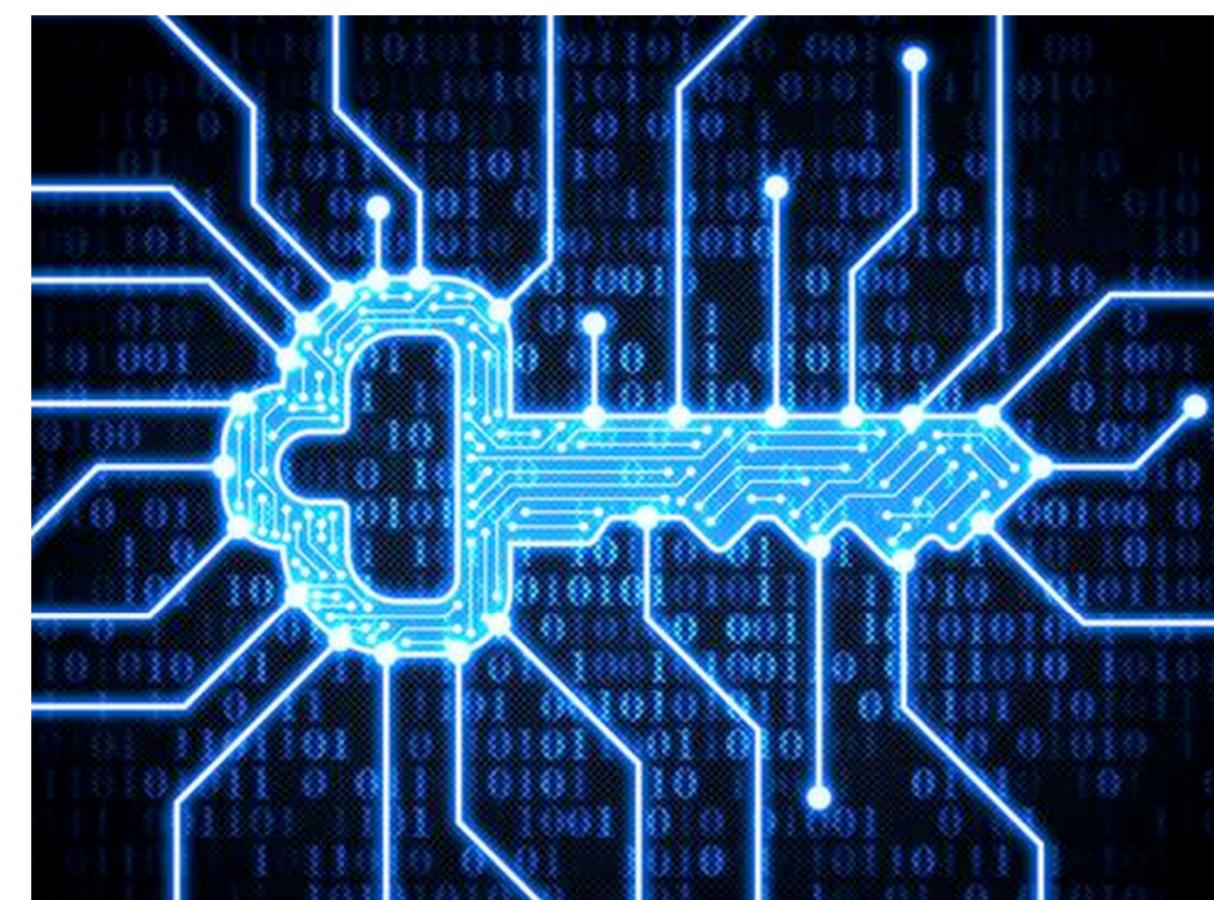
BAQUSERNAME - Specifies the username for connections if basic authentication is used.

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

General security terms or considerations

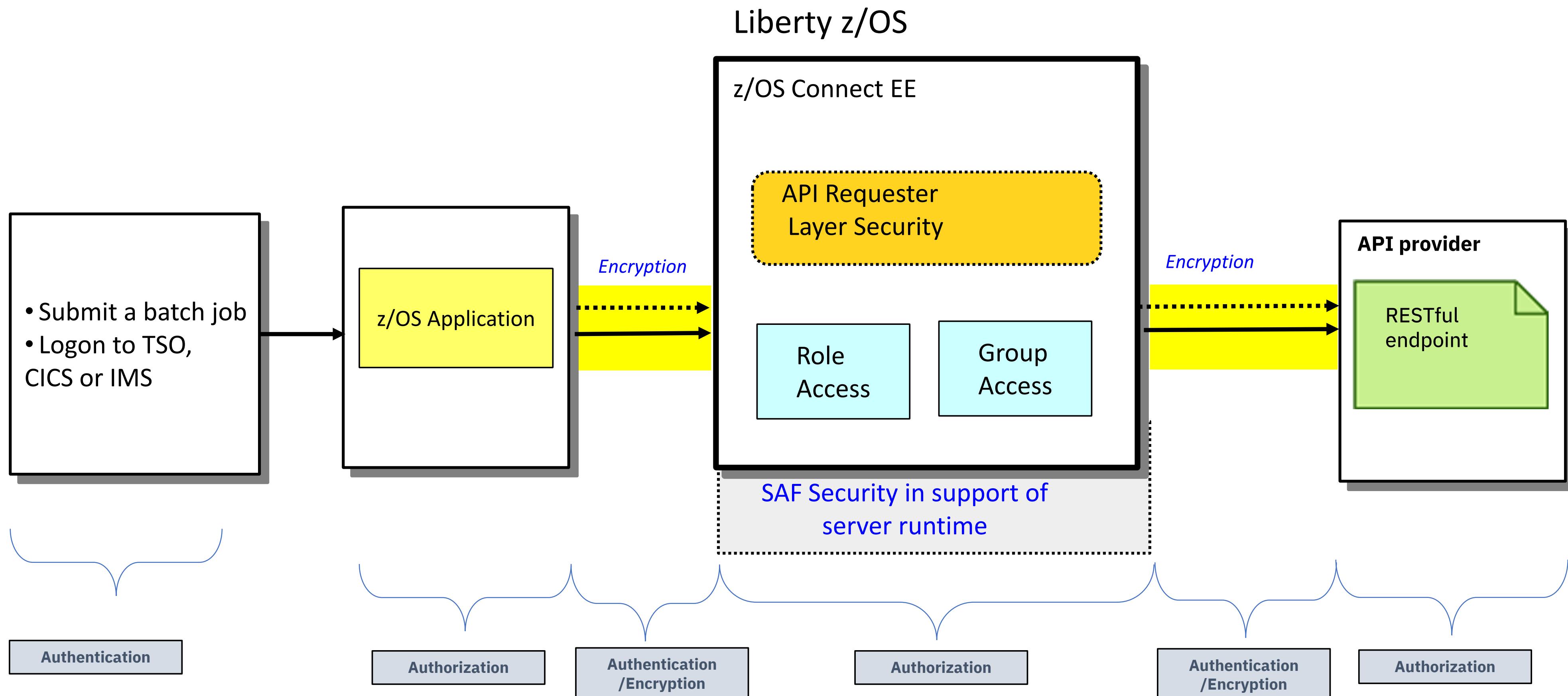
Security involves

- Identifying who or what is requesting access (**Authentication**)
 - Basic Authentication
 - Mutual Authentication using Transport Layer Security (TLS), formerly known as SSL
 - Third Party Tokens
- Ensuring that the message has not been altered in transit (**Data Integrity**) and ensuring the confidentiality of the message in transit (**Encryption**)
 - TLS (encrypting messages and using a digital signature)
- Controlling access (**Authorization**)
 - Is the authenticated identity authorized to access to z/OS Connect
 - Is the authenticated identity authorized to access a specific API, Services, etc.



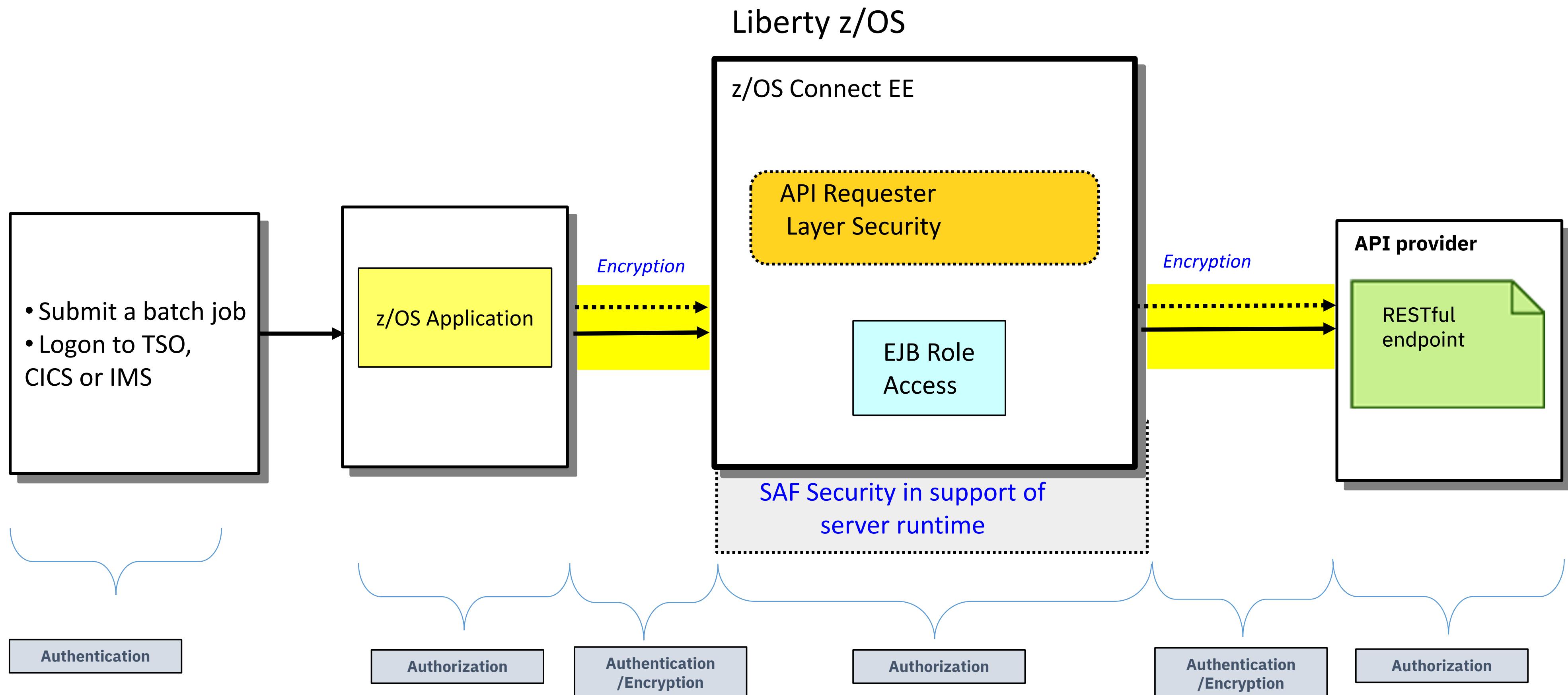


Outbound Authentication versus Authorization (OpenAPI 2)



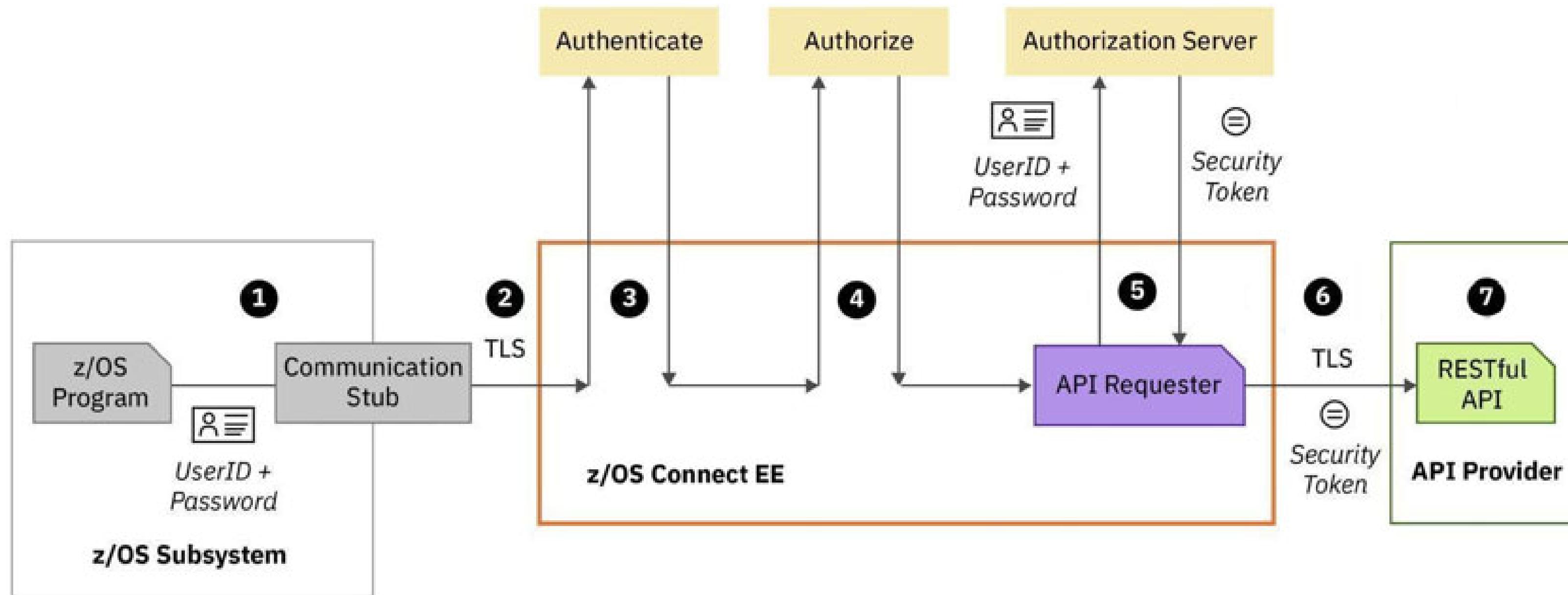


Outbound Authentication versus Authorization (OpenAPI 3)





Typical z/OS Connect EE API Requester security flow



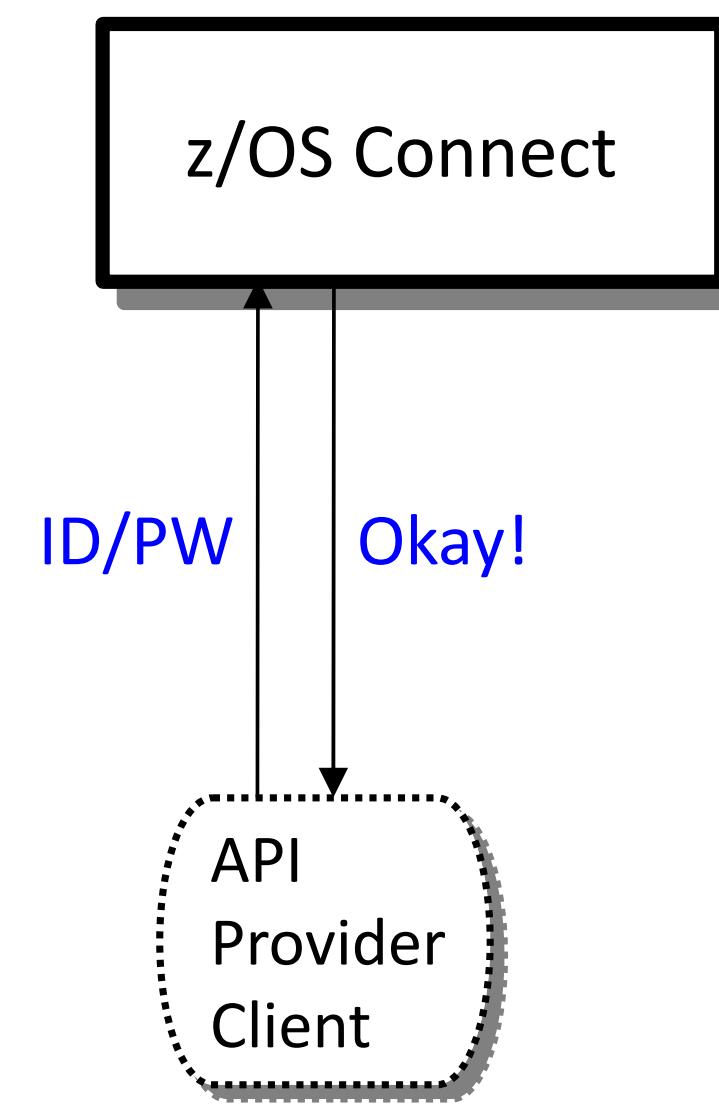
1. A user ID and password can be used for basic authentication by the z/OS Connect EE server
2. Connection between the CICS, IMS, or z/OS application and the z/OS Connect EE server can use TLS
3. Authenticate the CICS, IMS, or z/OS application.
4. Authorize the authenticated user ID to connect to z/OS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



API Requester – Security from the application to the z/OS Connect server

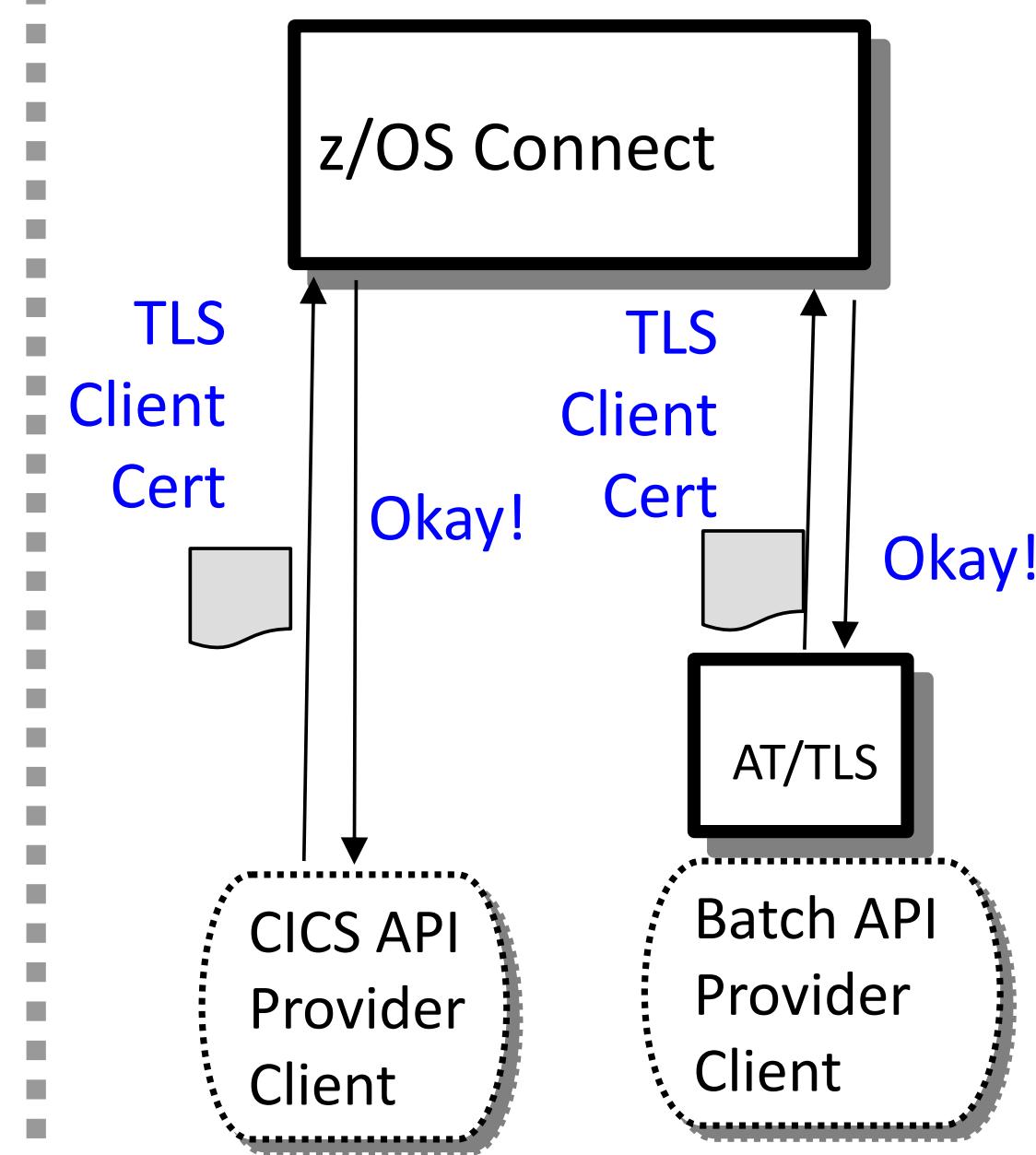
Two options for providing credentials for authentication

Basic Authentication



Application provides
ID/PW or ID/PassTicket

Client Certificate



z/OS Connect requests a client certificate

CICS or AT/TLS supplies a client certificate



Basic authentication – non-CICS COBOL API Requester

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

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

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

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

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

Tech/Tip: A PassTicket provides an alternative to a password



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

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



Tech/Tip: Generating PassTickets on z/OS

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

```

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

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

*-----*
MAINLINE SECTION.

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

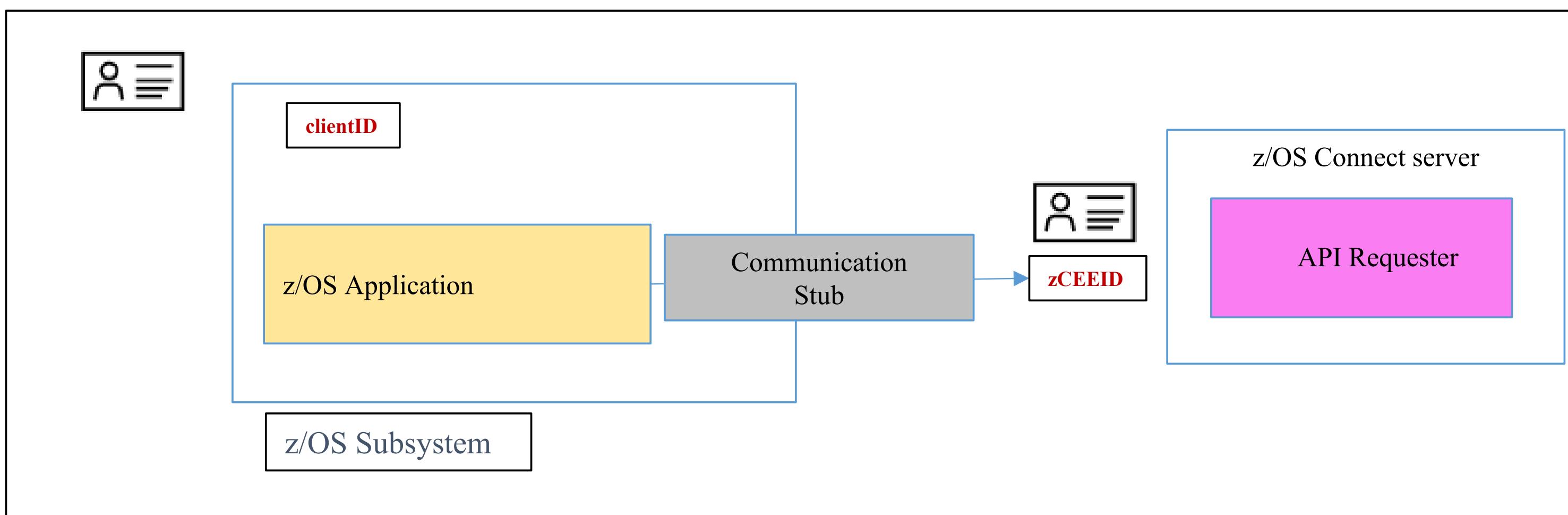
```

```

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

```

API Requester - basic authentication and identity assertion (OpenAPI 2 only)



clientID – the identity under which the z/OS application is executing.

- For CICS, the CICS task identity
- For IMS, the transaction owner
- For batch, the job card's USERID

zCEEID – The identity that is used for authenticating connectivity the z/OS subsystem to the zCEE server. It is configured using basic authentication or for CICS, TLS client authentication. For MVS batch, IMS and Db2 stored procedures, the **zCEEID** is provided by the environment variable **BAQUSERNAME**. For CICS, the value for **zCEEID** is usually provided by the identity mapped to the CICS client certificate.

requireAuth	idAssertion	Actions performed by z/OS Connect
true	OFF	Identity assertion is disabled. The zCEE server authenticates zCEEID and checks whether zCEEID has the authority to invoke an API requester.
	ASSERT_SURROGATE	Identity assertion is enabled. The zCEE server authenticates zCEEID and checks whether zCEEID is a surrogate of clientID . If zCEEID is a surrogate of clientID , the server further checks whether clientID has the authority to invoke an API requester; otherwise, a BAQR7114E message occurs.
	ASSERT_ONLY	Identity assertion is enabled. The zCEE server authenticates zCEEID and directly checks whether clientID has the authority to invoke an API requester.
false	OFF	Identity assertion is disabled. A BAQR0407W message occurs.
	ASSERT_SURROGATE	Identity assertion is enabled. The zCEE server checks whether clientID has the authority to invoke an API requester, and a warning message occurs to indicate that the ASSERT_ONLY value is used instead of the ASSERT_SURROGATE value.
	ASSERT_ONLY	Identity assertion is enabled. The zCEE server checks whether clientID has the authority to invoke an API requester.

```

<zosconnect_zosConnectManager
    requireAuth="true|false"
    requireSecure="true|false" />

<zosconnect_apiRequesters idAssertion="OFF">

<zosconnect_apiRequester name="cscvinc_1.0.0"
    requireAuth="true|false"
    requireSecure="true|false" />
    idAssertion="ASSERT_ONLY"> *

<zosconnect_apiRequester name="db2employee_1.0.0"
    requireAuth="true|false"
    requireSecure="true|false" />
    idAssertion="ASSERT_SURROGATE"> *

</zosconnect_apiRequesters>

```



Identity assertion requires setting a program control extended attribute

As root or superuser, set the *libifaedjreg64.so* program control extended attribute bit

- *Permit the server's identity to the required FACILITY resource*

**PERMIT BPX.SERVER CLASS(FACILITY) ID(*LIBSERV*) ACCESS(READ)
SETROPTS RACLIST(FACILITY) REFRESH**

- *Define a SURROGAT profile for the asserted identity and permit access to connection identity*

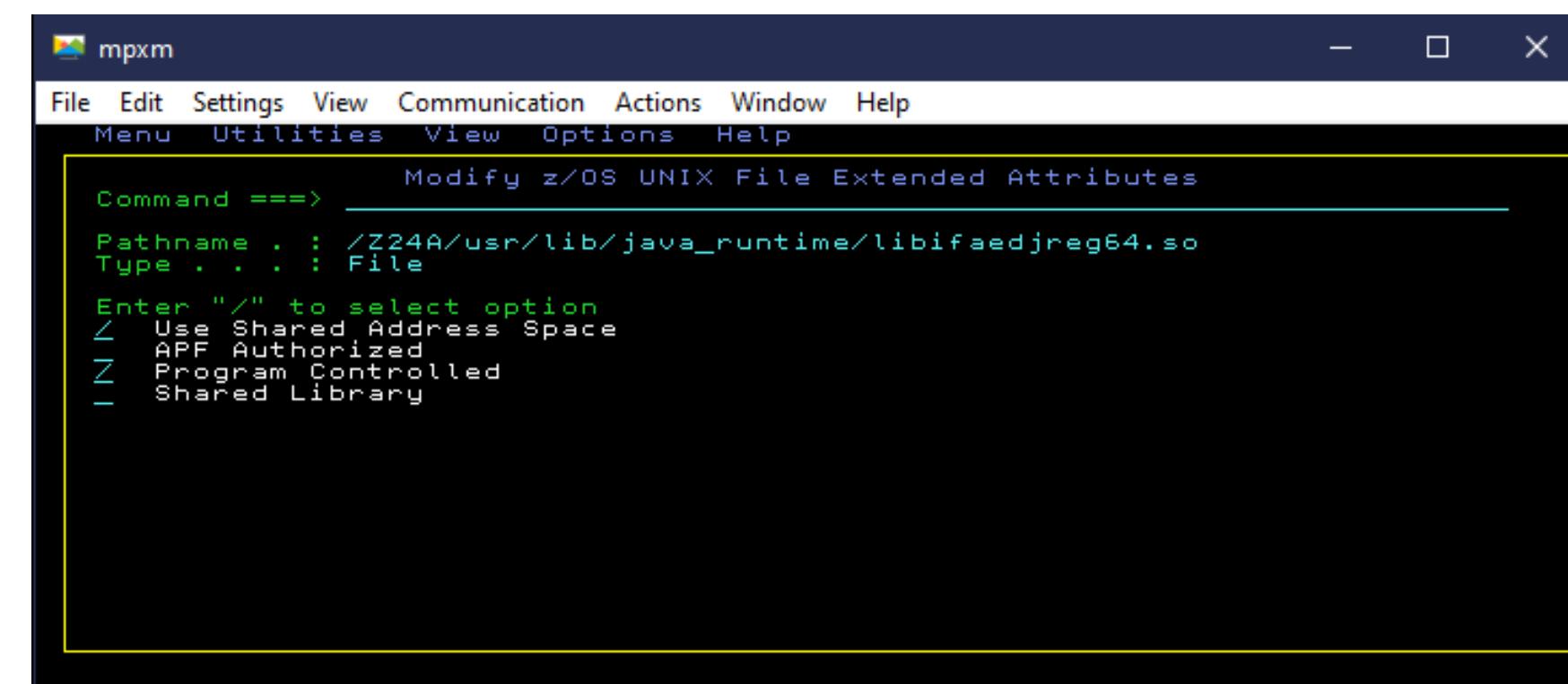
RDEFINE SURROGAT **clientID.BAQASSRT** UACC(NONE) OWNER(SYS1)
PERMIT **clientID.BAQASSRT** CLASS(SURROGAT) ACCESS(READ) ID(**zCEEID**)

OR

RDEFINE SURROGAT *.BAQASSRT UACC(NONE) OWNER(SYS1)
PERMIT *.BAQASSRT CLASS(SURROGAT) ACCESS(READ) ID(**zCEEID**)
SETROPTS RACLIST(SURROGAT) REFRESH

- *Enable the program control bit for Java shared object ifaedjreg64*

```
su  
cd /usr/lib/java_runtime  
extattr +p libifaedjreg64.so
```





API Requester – authorization (OpenAPI 3 only)

```
<safCredentials unauthenticatedUser="WSGUEST" profilePrefix="BBGZDFLT" />

<safRoleMapper profilePattern=%profilePrefix%.%resourceName%.%role%>

<webApplication location="${server.config.dir}/apps/cscvinc.war">
  <appProperties> <property name="connectionRef" value="cscvincConnection"/> </appProperties>
  <application-bnd> <security-role name="invoke"> <group name="staffGroup" /> <user name="fred" /> </security-role>
  </application-bnd>
</webApplication>
<webApplication name="catalogManager" location="${server.config.dir}/apps/catalog.war">
  <appProperties> <property name="connectionRef" value="catalogConnection"/> </appProperties>
  <application-bnd> <security-role name="invoke"> <group name="staffGroup" /> <user name="fred" /> </security-role>
  </application-bnd>
</webApplication>
```

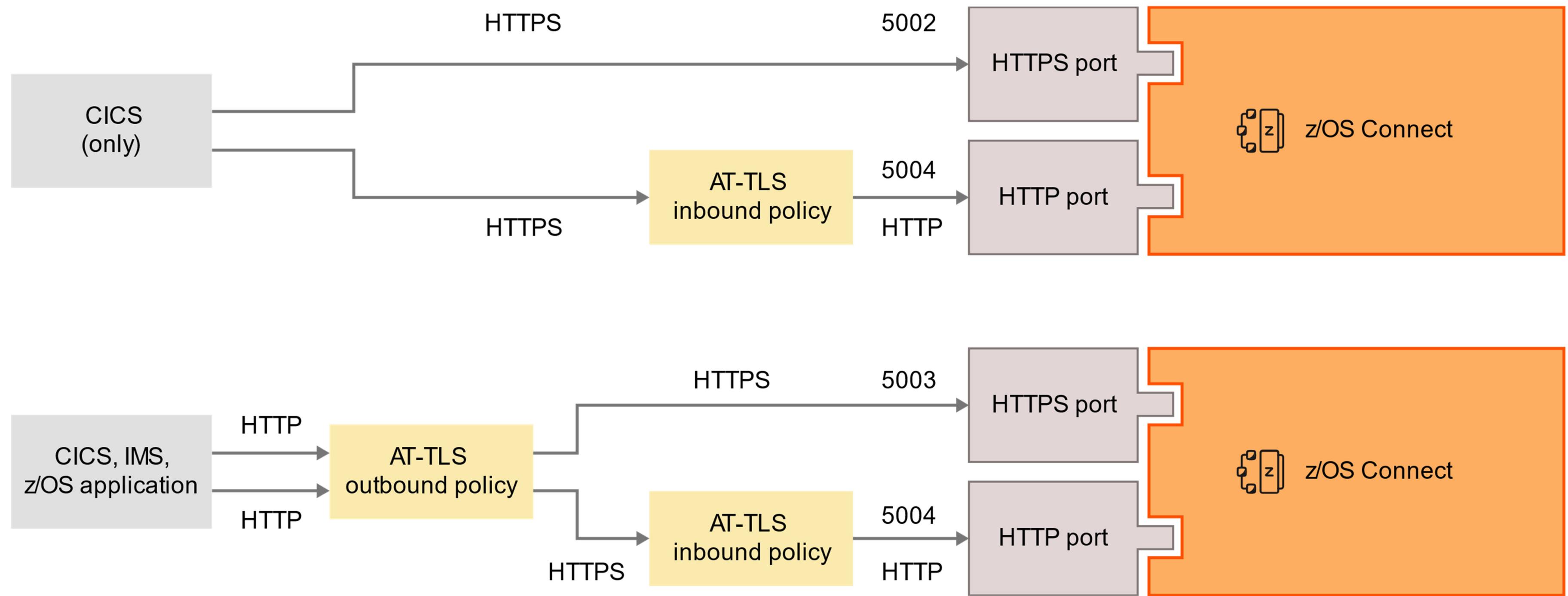
The *resourceName* defaults to the name of the WAR file if no name attribute is provided, otherwise the *resourceName* is value of the *name* attribute.

So, the required SAF EJB roles to be defined would be (*invoke* is the only role).

- *BBGZDFLT.cscvinc.invoke*
- *BBGZDFLT.catalogManager.invoke*

Authorization to invoke the API requester would require that the authenticated identity be a member of the STAFFGROUP or identity FRED.

TLS Connection options from an application to the z/OS Connect server





Tech/Tip: API Requester - HTTP v HTTPS

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

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

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

CICS URIMAPS

The image shows two CICS URIMAP configuration screens side-by-side, both titled 'WG31'. The left screen shows a configuration for port 9080, while the right screen shows a configuration for port 9443. Both screens display the same basic structure: CEDA ALter Urimap(BAQURIMP), followed by sections for Urimap, Group, Description, Status, Usage, and a large section for the UNIVERSAL RESOURCE IDENTIFIER. The 'HOST' field in both screens has 'wg31.washington.ibm.com' circled in red.

Left Screen (Port 9080):

```
OVERTYPE TO MODIFY
CEDA ALter Urimap( BAQURIMP )
Urimap      : BAQURIMP
Group       : SYSPGRP
Description ==> URIMAP for z/OS Connect EE server
Status      ==> Enabled          Enabled | Disabled
Usage       ==> Client           Server | Client | Pipeline |
                         | Jvmserver
UNIVERSAL RESOURCE IDENTIFIER
SCHEME      ==> HTTP            HTTP | HTTPS
PORT        ==> 09120           No | 1-65535
HOST        ==> wg31.washington.ibm.com
==>
PATH        ==> /
(Mixed Case) ==>
==>
==>
+ OUTBOUND CONNECTION POOLING
SYSID=CICS APPL
```

Right Screen (Port 9443):

```
OVERTYPE TO MODIFY
CICS RELEASE = 0710
CEDA ALter Urimap( BAQURIMP )
Urimap      : BAQURIMP
Group       : SYSPGRP
Description ==> URIMAP for z/OS Connect EE server
Status      ==> Enabled          Enabled | Disabled
Usage       ==> Client           Server | Client | Pipeline | Atom
                         | Jvmserver
UNIVERSAL RESOURCE IDENTIFIER
SCHEME      ==> HTTPS           HTTP | HTTPS
PORT        ==> 09443           No | 1-65535
HOST        ==> wg31.washington.ibm.com
==>
PATH        ==> /
(Mixed Case) ==>
==>
==>
+ OUTBOUND CONNECTION POOLING
SYSID=CICS APPLID=CICSS53Z
```

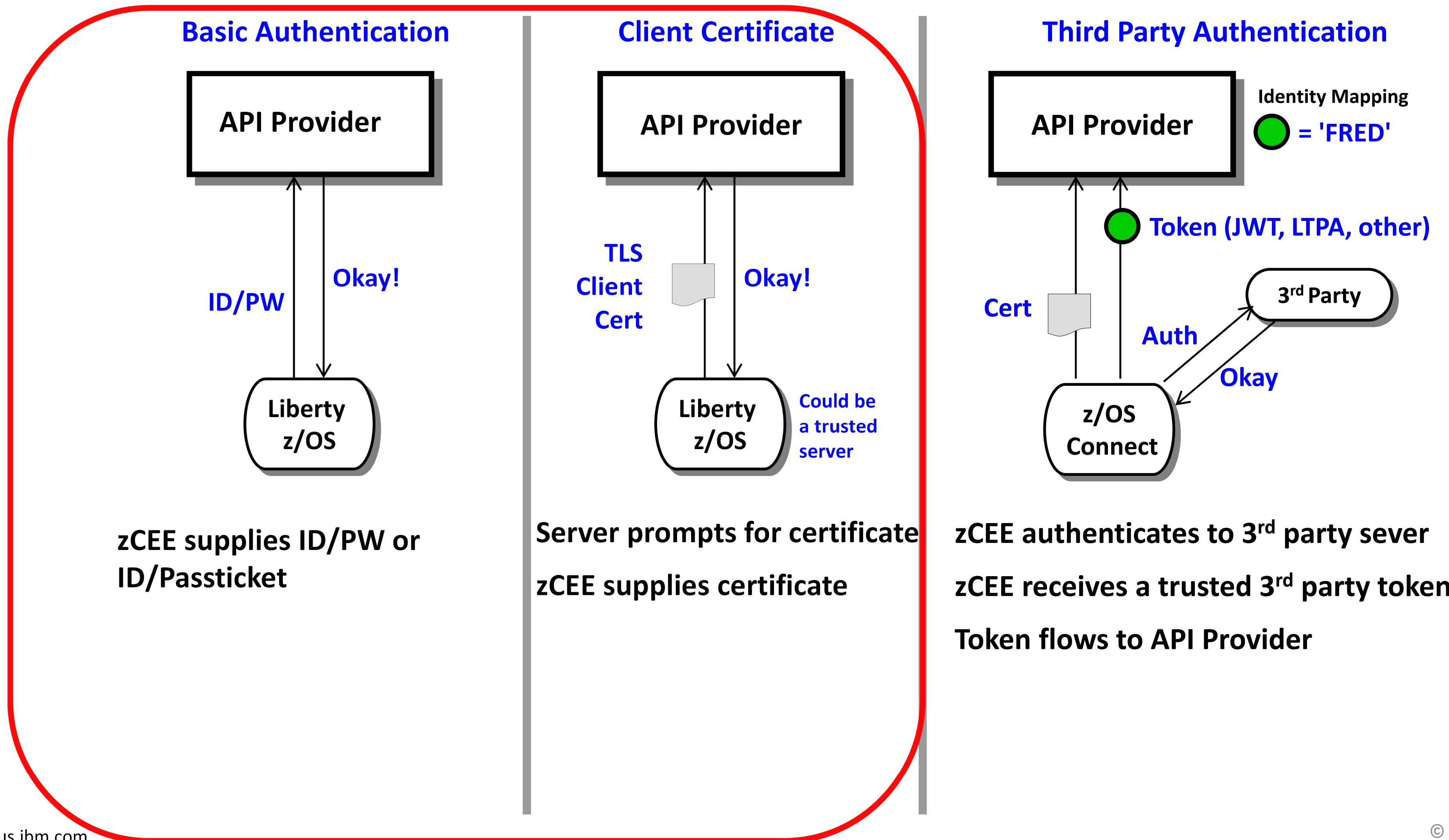
Field BAQ-ZCON-SERVER-URI was added to BAQRINFO in V3.0.37.

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



API Requester – Security from the z/OS Connect server to the API provider

Several different ways this can be accomplished:



Configuring Basic and/or TSL support – z/OS Connect API Requester



Basic authentication with HTTP protocol

```
<zosconnect_endpointConnection id="cscvincAPI"  
    host="http://wg31.washington.ibm.com" port="9080"  
    authenticationConfigRef="myAuthData" />  
  
<zosconnect_authData id="myAuthData"  
    user="zCEEclient" password="secret"/>
```

TLS with HTTPS protocol

```
<zosconnect_endpointConnection id="cscvincAPI"  
    host="https://wg31.washington.ibm.com" port="9443"  
    authenticationConfigRef="myAuthData" 1  
    sslCertsRef="OutboundSSLSettings" />  
  
<zosconnect_authData id="myAuthData" 1  
    user="zCEEclient" password="secret"/>
```

¹ Optional, if mutual authentication is enabled by the server endpoint



Sample JCL - Executing the Liberty *securityUtility* command

```
//*****  
/* Use securityUtility to encrypt a password using an  
/* encryption key stored in a certificate  
//*****  
//IKJEFT01 EXEC PGM=IKJEFT01,REGION=0M  
//SYSTSPRT DD SYSOUT=*  
//SYSERR DD SYSOUT=*  
//STDOUT DD SYSOUT=*  
//SYSTSIN DD *  
BPXBATCH SH +  
/usr/lpp/IBM/zosconnect/v3r0/wlp/bin/securityUtility encode +  
--encoding=aes +  
--keyring=safkeyring://JOHNSON/Liberty.KeyRing +  
--keyringType=JCERACFKS --keyLabel="Johnson Client Cert" +  
passwordToEncrypt
```

```
<featureManager>  
  <feature>zosPasswordEncryptionKey-1.0</feature>  
</featureManager>  
  
<zosPasswordEncryptionKey  
keyring="safkeyring://JOHNSON/Liberty.KeyRing"  
label="Johnson Client Cert" type="JCERACFKS"/>
```

```
//*****  
/* Use securityUtility to encrypt a password using an  
/* encryption key string  
//*****  
//IKJEFT01 EXEC PGM=IKJEFT01,REGION=0M  
//SYSTSPRT DD SYSOUT=*  
//SYSERR DD SYSOUT=*  
//STDOUT DD SYSOUT=*  
//SYSTSIN DD *  
BPXBATCH SH +  
/usr/lpp/IBM/zosconnect/v3r0/wlp/bin/securityUtility encode +  
--encoding=aes -key myEncryptionKey +  
passwordToEncrypt
```

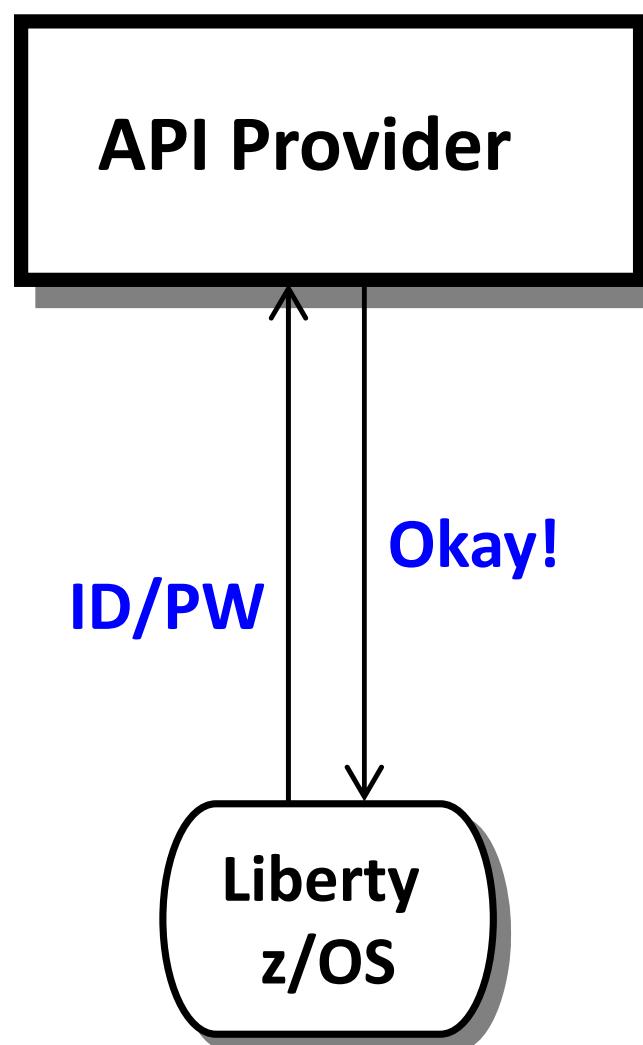
```
wlp.password.encryption.key=myEncryptionKey
```



API Requester – Security from the z/OS Connect server to the API provider

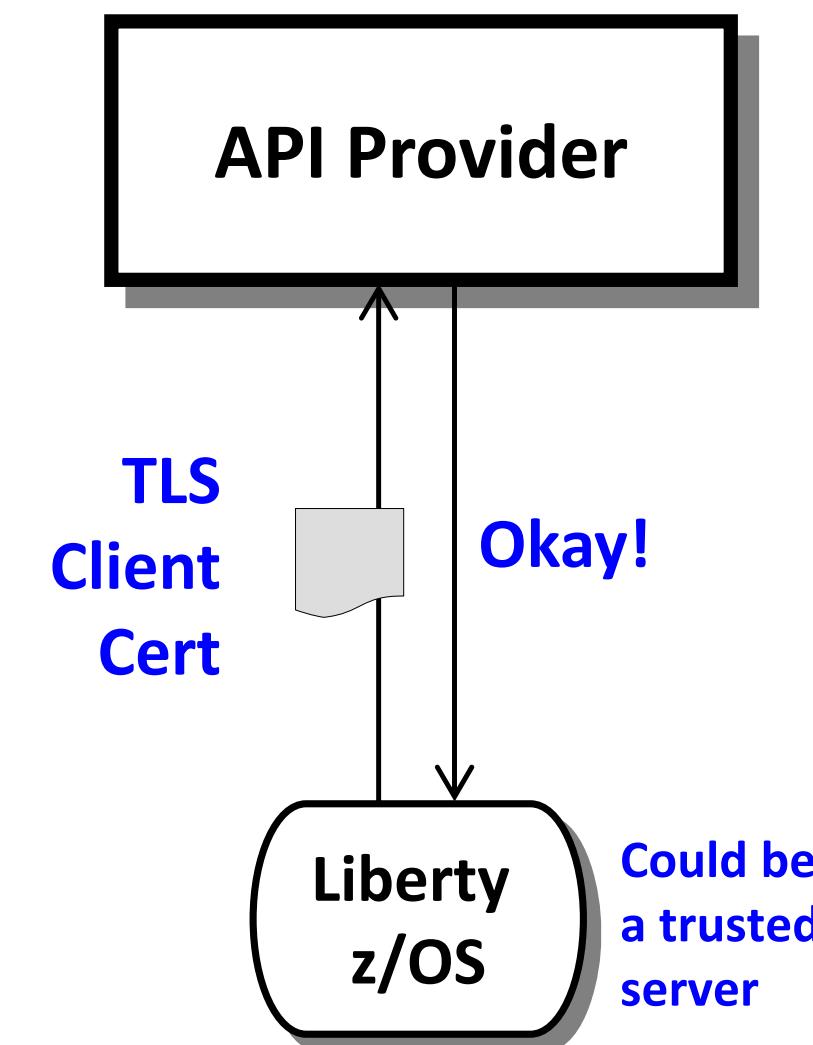
Several different ways this can be accomplished:

Basic Authentication



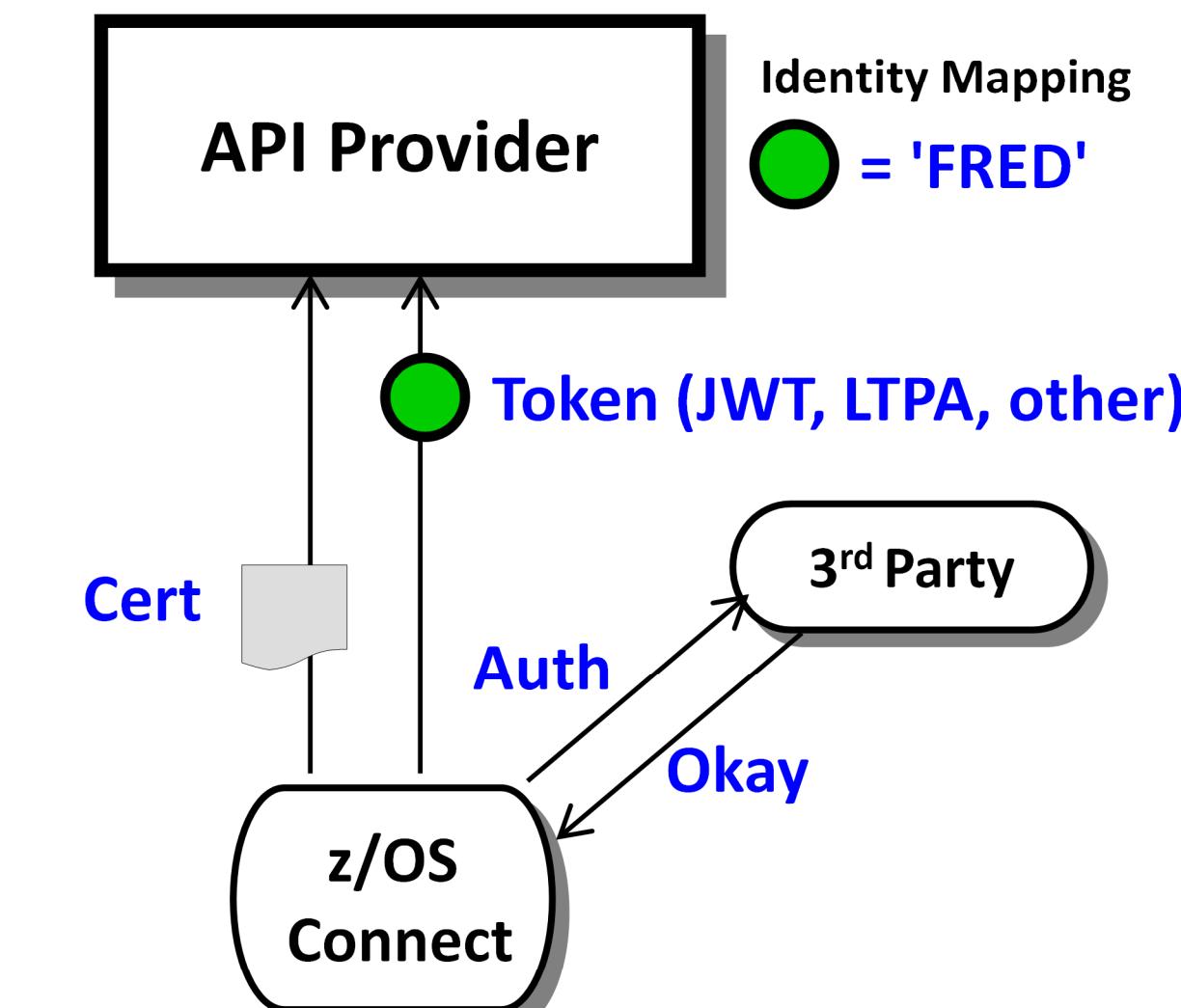
zCEE supplies ID/PW or
ID/Passticket

Client Certificate



Server prompts for certificate
zCEE supplies certificate

Third Party Authentication



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



Third Party Authentication Examples

The screenshot shows the UPS Sign Up page. At the top, there's a banner about UPS being open for business due to COVID-19. Below the banner, the UPS logo is displayed. A "Sign Up" button is prominent. Below it, a link to "Log in" is shown. There are sections for "Use one of these sites." with links to Google, Facebook, Amazon, and Apple. Another section for "Or enter your own information." contains fields for Name*, Email*, User ID*, Password*, and Phone. A "Show" link is next to the Password field.

The screenshot shows the myNCDMV Log In page. It features a "Log In" tab and a "Sign Up" tab. The "Log In" section includes fields for "Email Address" (with "name@example.com" entered) and "Password". There's a "Remember Me" checkbox, a "Log In" button, and a "Forgot Password" link. Below these are three social login options: "Continue with Apple", "Continue with Facebook", and "Continue with Google". A "Continue as Guest" link is also present. At the bottom, a notice for public computer users states: "NOTICE FOR PUBLIC COMPUTER USERS - If you sign in with Google, Apple, or Facebook you are also signing into that account on this computer. Remember to sign out when you're done." The page is powered by **payit**.



Open security standards

- **OAuth** is an open standard for access delegation, used as a way to grant websites or applications access to their information without requiring a password.
- **OpenID Connect** is an authentication layer on top of OAuth. It allows the verification of the identity of an end-user based on authentication performed by an authorization server.
- **JWT (JSON Web token)** defines a compact and self-contained way for securely transmitting information between parties as a JSON object

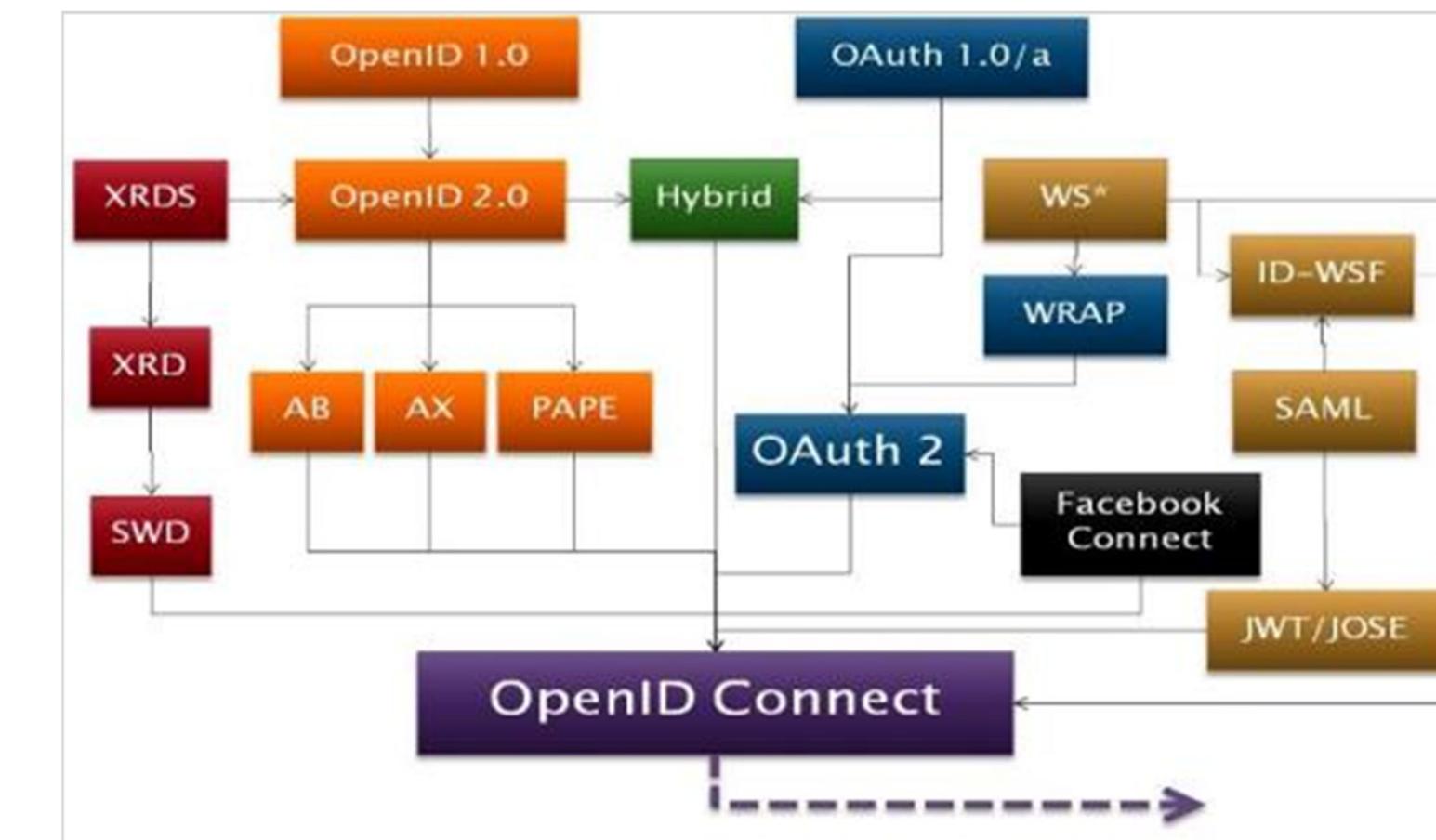
See the YouTube videos:

OAuth 2.0 and OpenID Connect (in plain English)

<https://www.youtube.com/watch?v=996OjexHze0>

OpenID Connect on Liberty

<https://www.youtube.com/watch?v=fuajCS5bG4c>





What is a 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 debugger interface. On the left, under 'Encoded', is a long string of characters: eyJraWQiOiiI0cWpYLWJrWE9Vd19GX...vT_Ez0fD-...vtHPxav4cBrGNRDR4ubRo-. In the center, under 'Decoded', are two JSON objects. The 'HEADER' object contains: { "kid": "4qjX-bkX0Uw_F_uccjRMkB9ivMjXSQwj0RrkyRJq8DM", "alg": "RS256" }. The 'PAYLOAD' object contains: { "sub": "Fred", "token_type": "Bearer", "scope": ["openid", "profile", "email"], "azp": "rpSsl", "iss": "https://wg31.washington.ibm.com:26213/oidc/endpoint/OP", "aud": "myZcee", "exp": 160433158, "iat": 160433158, "realmName": "zCEERealm", "uniqueSecurityName": "Fred" }. A red oval highlights the 'exp' and 'iat' fields in the payload.

Encoded	Decoded
eyJraWQiOiiI0cWpYLWJrWE9Vd19GX...vT_Ez0fD-...vtHPxav4cBrGNRDR4ubRo-	HEADER: { "kid": "4qjX-bkX0Uw_F_uccjRMkB9ivMjXSQwj0RrkyRJq8DM", "alg": "RS256" } PAYLOAD: { "sub": "Fred", "token_type": "Bearer", "scope": ["openid", "profile", "email"], "azp": "rpSsl", "iss": "https://wg31.washington.ibm.com:26213/oidc/endpoint/OP", "aud": "myZcee", "exp": 160433158, "iat": 160433158, "realmName": "zCEERealm", "uniqueSecurityName": "Fred" }

Values derived from the OAUTH configuration:

- signatureAlgorithm="**RS256**"
- accessTokenLifetime="**300**"
- resourceIds="**myZcee**"

<https://jwt.io>

© 2018, 2023 IBM Corporation

Slide 98

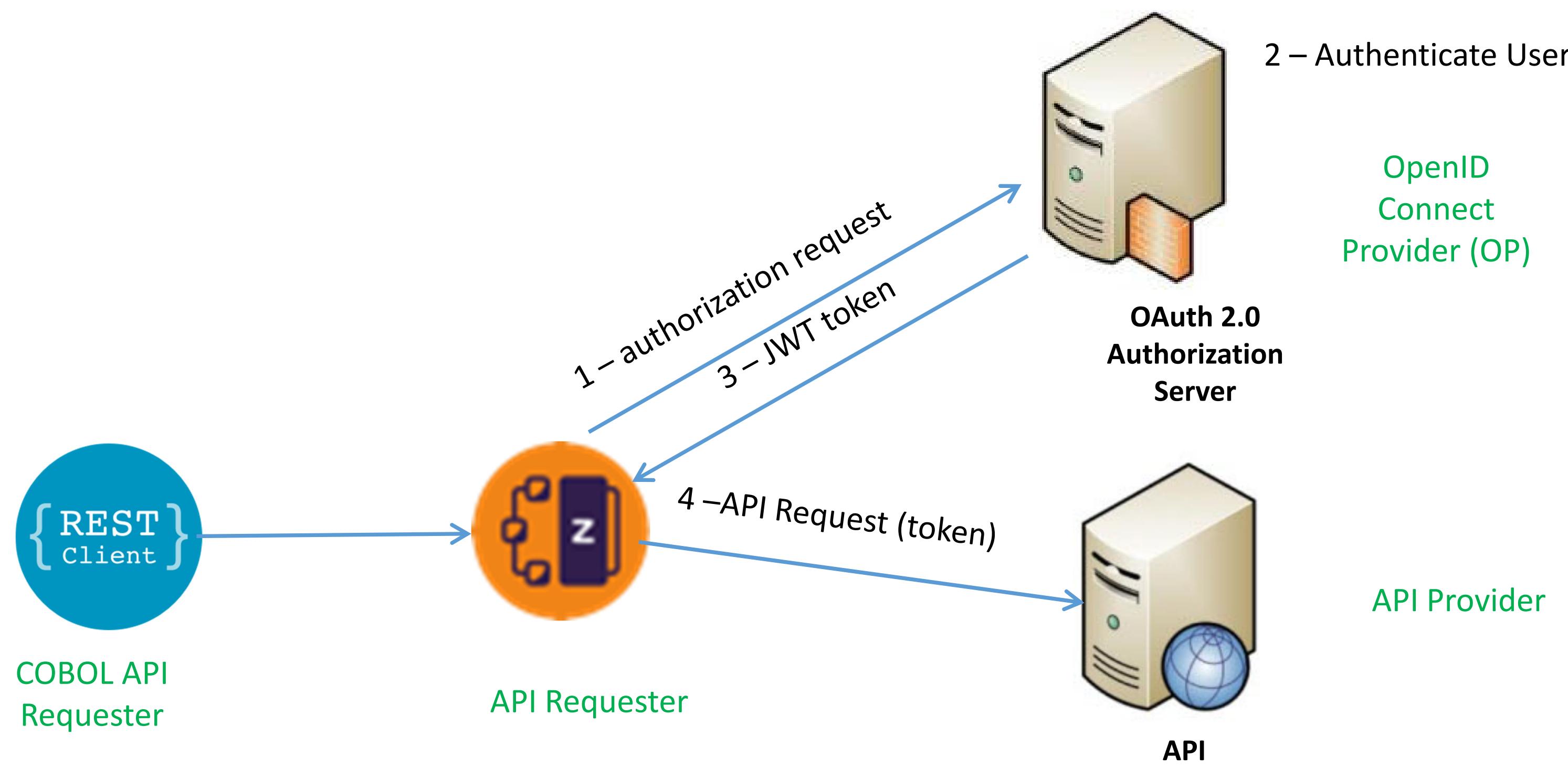
z/OS Connect API Requester - Token Support



z/OS Connect EE provides *three* ways of calling an API secured with a token

1. Use the OAuth 2.0 support when the request is part of an OAuth 2.0 flow. With OAAUTH configured, the token can be an opaque token or a JWT token.
1. In a non-OAuth 2.0 scenario, a JWT token is used in a custom flow, for example: when you need to specify the HTTP verb that is used in the request to the authentication server.
 - When you need to specify the HTTP verb that is used in the request to the authentication server
 - When you need to specify how the JWT is returned from the authentication server (for example, in an HTTP header or in a custom field in a JSON response message).
 - When you need to use a custom header name for sending the JWT to the request endpoint.
3. Use the locally generated JWT support when you need to send a JWT that is generated by the z/OS Connect EE server.

z/OS Connect OAuth Flow for API requester

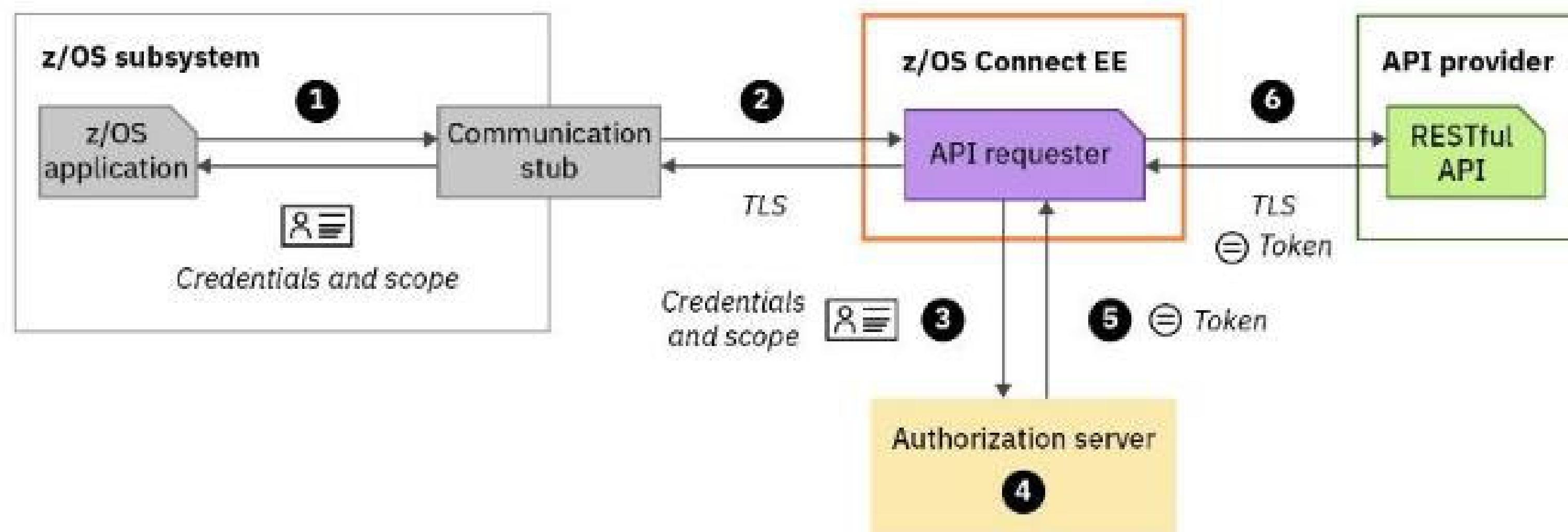


Grant Types:

- client_credentials
- password



Calling an API with OAuth 2.0 support





OAuth Grant Types Supported by z/OS Connect

client_credentials - the identity associated with the combination of the CICS, IMS, or z/OS region **and** the z/OS Connect EE server that calls the RESTful API on behalf of the CICS, IMS, or z/OS application. When this grant type is used, the z/OS Connect EE server sends the client credentials and the access scope to the authorization server.

```
<zosconnect_oAuthConfig id="myoAuthConfig"  
    grantType="client_credentials"  
    authServerRef="myoAuthServer"/>
```

password - The identity of the specific identity provided by the CICS, IMS, or z/OS application, or it might be another entity. When this grant type is used, the z/OS Connect EE server sends the resource owner's credentials, the client credentials, and the access scope to the authorization server.

```
<zosconnect_oAuthConfig id="myoAuthConfig"  
    grantType="password"  
    authServerRef="myoAuthServer"/>
```

OpenID Connect/OAuth and z/OS Connect



- **From the z/OS Connect Knowledge Center:** z/OS Connect EE security can operate with traditional z/OS security, for example, System Authorization Facility (SAF) and also with open standards such as Transport Layer Security (TLS), JSON Web Token (JWT), and **OpenID Connect**.
- **From the OpenID Core specification:** OpenID Connect 1.0 is a simple identity layer on top of the OAuth 2.0 protocol. It enables Clients to verify the identity of the End-User based on the authentication performed by an Authorization Server, as well as to obtain basic profile information about the End-User in an interoperable and REST-like manner.
- **OAuth 2.0 Core (RFC 6749) Specifications:** <https://tools.ietf.org/html/rfc6749>
- **OpenID Connect Core Specifications:** https://openid.net/specs/openid-connect-core-1_0.html
- **Again, for a very good explanation of this topic see YouTube video OAuth 2.0 and OpenID Connect (in plain English)**
<https://www.youtube.com/watch?v=996OiexHze0>

Configuring OAuth support – BAQRINFO copy book



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

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

Scope is always required.

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



Obtaining a JWT using request parameters

The image displays two terminal windows from the z/OS environment. The left window shows the definition of a host API named ZCEE30.SBAQCOB(BAQHCONC) with various parameter names and their types. The right window shows a CBL APOST script for generating a JWT, including sections for authentication credentials and sending them to z/OS Connect.

Left Window (Host API Definition):

```
BROWSE ZCEE30.SBAQCOB(BAQHCONC)
Command ===>
  * Host API Request parameter names
  77 BAQR-OAUTH-USERNAME      PIC X(22)
    VALUE 'BAQHAPI-oAuth-Username'.
  77 BAQR-OAUTH-PASSWORD      PIC X(22)
    VALUE 'BAQHAPI-oAuth-Password'.
  77 BAQR-OAUTH-SCOPE         PIC X(19)
    VALUE 'BAQHAPI-oAuth-Scope'.
  77 BAQR-OAUTH-CLIENT-ID     PIC X(22)
    VALUE 'BAQHAPI-oAuth-ClientId'.
  77 BAQR-OAUTH-CLIENT-SECRET PIC X(26)
    VALUE 'BAQHAPI-oAuth-ClientSecret'.
  77 BAQR-OAUTH-RESOURCE      PIC X(22)
    VALUE 'BAQHAPI-oAuth-Resource'.
  77 BAQR-OAUTH-AUDIENCE      PIC X(22)
    VALUE 'BAQHAPI-oAuth-Audience'.
  77 BAQR-OAUTH-CUSTOM-PARMS  PIC X(25)
    VALUE 'BAQHAPI-oAuth-CustomParms'.
  77 BAQR-JWT-USERNAME        PIC X(22)
    VALUE 'BAQHAPI-Token-Username'.
  77 BAQR-JWT-PASSWORD        PIC X(22)
    VALUE 'BAQHAPI-Token-Password'.

  * Host API ZCON parameter names
  77 BAQZ-TRACE-VERBOSE      PIC X(21)
    VALUE 'BAQHAPI-Trace-Verbose'.
  77 BAQZ-SERVER-URIMAP       PIC X(21)
    VALUE 'BAQHAPI-Server-URIMAP'.
  77 BAQZ-SERVER-HOST         PIC X(19)
```

Right Window (CBL APOST Script):

```
EDIT USER1.ZCEE30.SOURCE(GETAPI) - 01.02
Command ===>
***** **** Top of Data ****
000001 CBL APOST
000002
000003 * Authentication server credentials
000004 01 JWT-USER PIC X(10) VALUE 'myUsername'.
000005 01 JWT-PSWD PIC X(10) VALUE 'myPassword'.
000006
000007
000008
000009
000010 * Send JWT credentials to z/OS Connect
000011 MOVE BAQR-TOKEN-USERNAME TO
000012   BAQ-REQ-PARM-NAME OF BAQ-REQ-PARMS(1)
000013 SET BAQ-REQ-PARM-ADDRESS OF
000014   BAQ-REQ-PARMS(1) TO ADDRESS OF JWT-USER
000015 MOVE LENGTH OF JWT-USER TO
000016   BAQ-REQ-PARM-LENGTH OF BAQ-REQ-PARMS(1)
000017 MOVE BAQR-TOKEN-PASSWORD TO
000018   BAQ-REQ-PARM-NAME OF BAQ-REQ-PARMS(2)
000019 SET BAQ-REQ-PARM-ADDRESS OF
000020   BAQ-REQ-PARMS(2) TO ADDRESS OF JWT-PSWD
000021 MOVE LENGTH OF JWT-PSWD TO
000022   BAQ-REQ-PARM-LENGTH OF BAQ-REQ-PARMS(2)
000023 * Call the API endpoint using BAQEXEC
000024
000025
000026
000027
```



Sample program and JCL

COBOL Application

```
MOVE "ATSOAUTHUSERNAME" to envVariableName.  
PERFORM CALL-CEEENV THRU CALL-CEEENV-END  
MOVE VAR(1:valueLength) to BAQ-OAUTH-USERNAME  
MOVE valueLength TO BAQ-OAUTH-USERNAME-LEN  
MOVE "ATSOAUTHPASSWORD" to envVariableName.  
PERFORM CALL-CEEENV THRU CALL-CEEENV-END  
MOVE VAR(1:valueLength) to BAQ-OAUTH-PASSWORD  
MOVE valueLength to BAQ-OAUTH-PASSWORD-LEN  
MOVE SPACES      to BAQ-OAUTH-CLIENTID.  
MOVE 0          to BAQ-OAUTH-CLIENTID-LEN.  
MOVE SPACES      to BAQ-OAUTH-CLIENT-SECRET.  
MOVE 0          to BAQ-OAUTH-CLIENT-SECRET-LEN.  
MOVE "openid"    to BAQ-OAUTH-SCOPE.  
MOVE 6          to BAQ-OAUTH-SCOPE-LEN.  
SET BAQ-OAUTH-SCOPE-PTR TO ADDRESS OF BAQ-OAUTH-SCOPE .
```

Note that this example is using environment variables to provide OAuth credentials, as documented in the z/OS Connect Advanced Topics Guide.

Execution JCL

```
//GETAPI EXEC PGM=GETAPIPT,PARM='111111'  
//STEPLIB DD DISP=SHR,DSN=USER1.ZCEE30.LOADLIB  
//           DD DISP=SHR,DSN=ZCEE30.SBAQLIB  
//           DD DISP=SHR,DSN=JOHNSON.ZCEE.SDFHLOAD  
//SYSOUT   DD SYSOUT=*  
//SYSPRINT DD SYSOUT=*  
//CEEOPTS DD *  
POSIX(ON),  
ENVAR ("BAQURI=wg31.washington.ibm.com",  
"BAQPORT=9080",  
"BAQUSERNAME=USER1",  
"ATSAPPL=BBGZDFLT",  
"ATSOAUTHUSERNAME=distuser1",  
"ATSOAUTHPASSWORD=pwd")
```



Tech/Tip: Accessing environment variables from COBOL application

```
*****
** Get the BAQ-OAUTH-USERNAME environment variable
*****
MOVE "ATSOAUTHUSERNAME" TO envVariableName.
PERFORM CALL-CEEENV THRU CALL-CEEENV-END
IF valueLength NOT = 0 THEN
  MOVE VAR(1:valueLength) TO BAQ-OAUTH-USERNAME
  MOVE valueLength TO BAQ-OAUTH-USERNAME-LEN
  DISPLAY "BAQ-OAUTH-USERNAME: "
    BAQ-OAUTH-USERNAME(1:BAQ-OAUTH-USERNAME-LEN)
ELSE
  DISPLAY "BAQ-OAUTH-USERNAME: Not found"
END-IF.
*****
** Get the BAQ-OAUTH-PASSWORD environment variable
*****
MOVE "ATSOAUTHPASSWORD" TO envVariableName.
PERFORM CALL-CEEENV THRU CALL-CEEENV-END
IF valueLength NOT = 0 THEN
  MOVE VAR(1:valueLength) TO BAQ-OAUTH-PASSWORD
  MOVE valueLength TO BAQ-OAUTH-PASSWORD-LEN
  DISPLAY "BAQ-OAUTH-PASSWORD: "
    BAQ-OAUTH-PASSWORD(1:BAQ-OAUTH-PASSWORD-LEN)
ELSE
  DISPLAY "BAQ-OAUTH-PASSWORD: Not found"
```

```
CALL-CEEENV.
  MOVE 1 TO functionCode.
  MOVE ZERO TO ws-length.
  INSPECT FUNCTION REVERSE (envVariableName)
    TALLYING ws-length FOR LEADING SPACES.
  COMPUTE envVariableNameLength =
    LENGTH OF envVariableName - ws-length.
  MOVE " " TO VAL.
  MOVE 0 TO valueLength.
  CALL "CEEENV" USING functionCode,
    envVariableNameLength,
    envVariableName,
    valueLength,
    valuePointer,
    feedbackCode.

  IF valueLength NOT = 0 THEN
    SET ADDRESS OF VAR TO valuePointer .

CALL-CEEENV-END.,
```

Configuring OAuth support – z/OS Connect API Requester



```
<zosconnect_endpointConnection id="cscvincAPI"
    host="http://wg31.washington.ibm.com" port="9080"
    authenticationConfigRef="myoAuthConfig"/>

<zosconnect_oAuthConfig id="myoAuthConfig"
    grantType="client_credentials|password"
    authServerRef="myoAuthServer"/>

<zosconnect_authorizationServer id="myoAuthServer"
    tokenEndpoint="https://wg31.washington.ibm.com:59443/oidc/endpoint/OP/token1
    basicAuthRef="tokenCredential" 2
    sslCertsRef="OutboundSSLSettings" />

<zosconnect_authData id="tokenCredential" 2
    user="zCEEClient" password="secret"/>
```

```
openidConnectProvider id="OP"
    signatureAlgorithm="RS256"
    keyStoreRef="jwtStore"
    oauthProviderRef="OIDCssl" >
</openidConnectProvider>
```

¹See URL https://www.ibm.com/support/knowledgecenter/SS7K4U_liberty/com.ibm.websphere.wlp.zseries.doc/ae/twlp_oidc_token_endpoint.html

² These credentials can be specified by the application

Security Scenarios



```
BAQ-OAUTH-USERNAME: distuser1  
BAQ-OAUTH-PASSWORD: pwd  
EmployeeNumber: 111111  
EmployeeName: C. BAKER  
USERID: USER1
```

distuser1 is mapped to RACF identity USER1 who has full access

```
BAQ-OAUTH-USERNAME: distuserx  
BAQ-OAUTH-PASSWORD: pwd  
Error code: 00000500  
Error msg:{ "errorMessage": "BAQR1092E: Authentication or authorization failed for the z/OS Connect EE server." }
```

distuserx is unknown by the OAuth Provider

```
BAQ-OAUTH-USERNAME: auser  
BAQ-OAUTH-PASSWORD: pwd  
Error code: 0000000403  
rror msg:{ "errorMessage": "BAQR1144E: Authentication or authorization failed for the z/OS Connect EE server."  
Syslog:  
ICH408I USER(ATSSERV ) GROUP(ATSGRP ) NAME(LIBERTY SERVER  
DISTRIBUTED IDENTITY IS NOT DEFINED:  
auser zCEERealm
```

auser is not mapped to a valid RACF identity

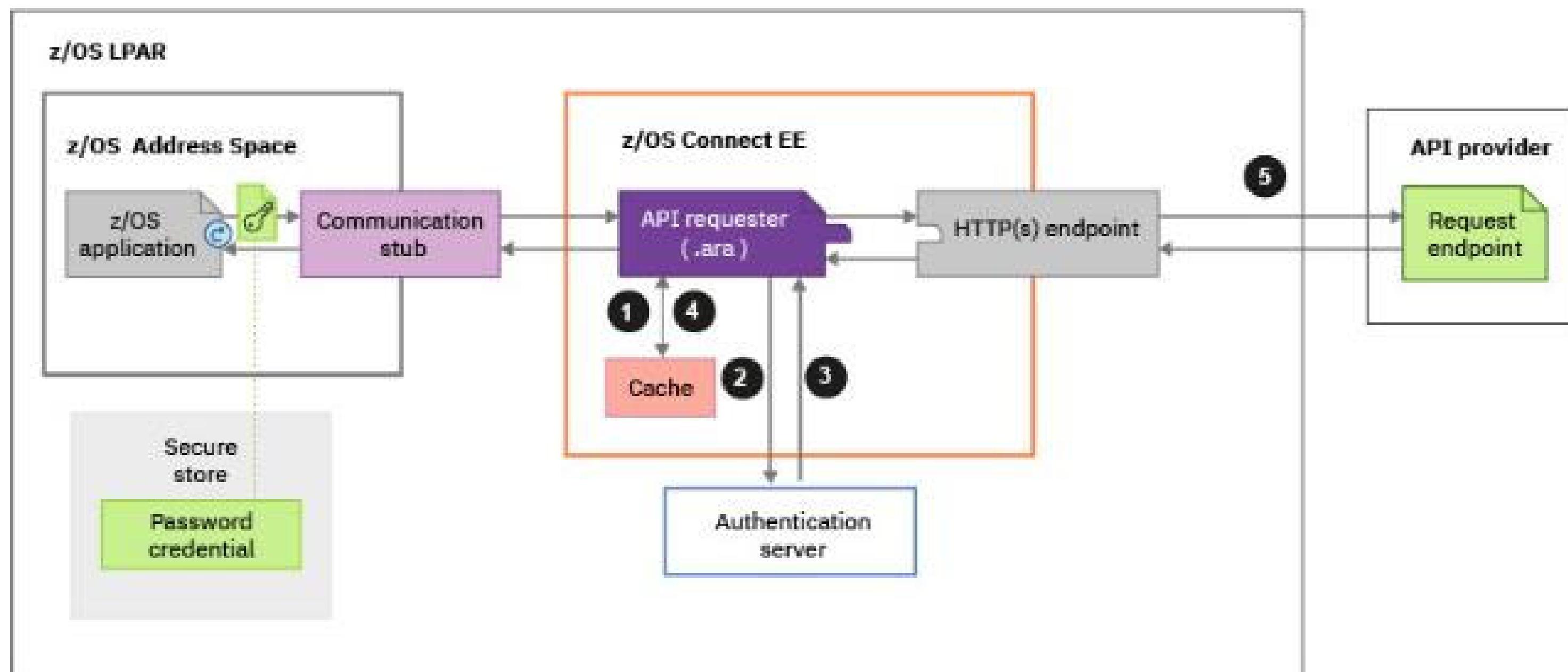
```
BAQ-OAUTH-USERNAME: distuser2  
BAQ-OAUTH-PASSWORD: pwd  
Error code: 0000000403  
Error msg:{ "errorMessage": "BAQR1144E: Authentication or authorization failed for the z/OS Connect EE server."  
Syslog:  
ICH408I USER(USER2 ) GROUP(SYS1 ) NAME(WORKSHOP USER2  
ATSZDFLT.zos.connect.access.roles.zosConnectAccess  
CL(EJBROLE )  
INSUFFICIENT ACCESS AUTHORITY  
ACCESS INTENT(READ ) ACCESS ALLOWED(NONE )
```

)
distuser2 is mapped to RACF identity USER2 which has no access to the EJBRole protecting z/OS Connect



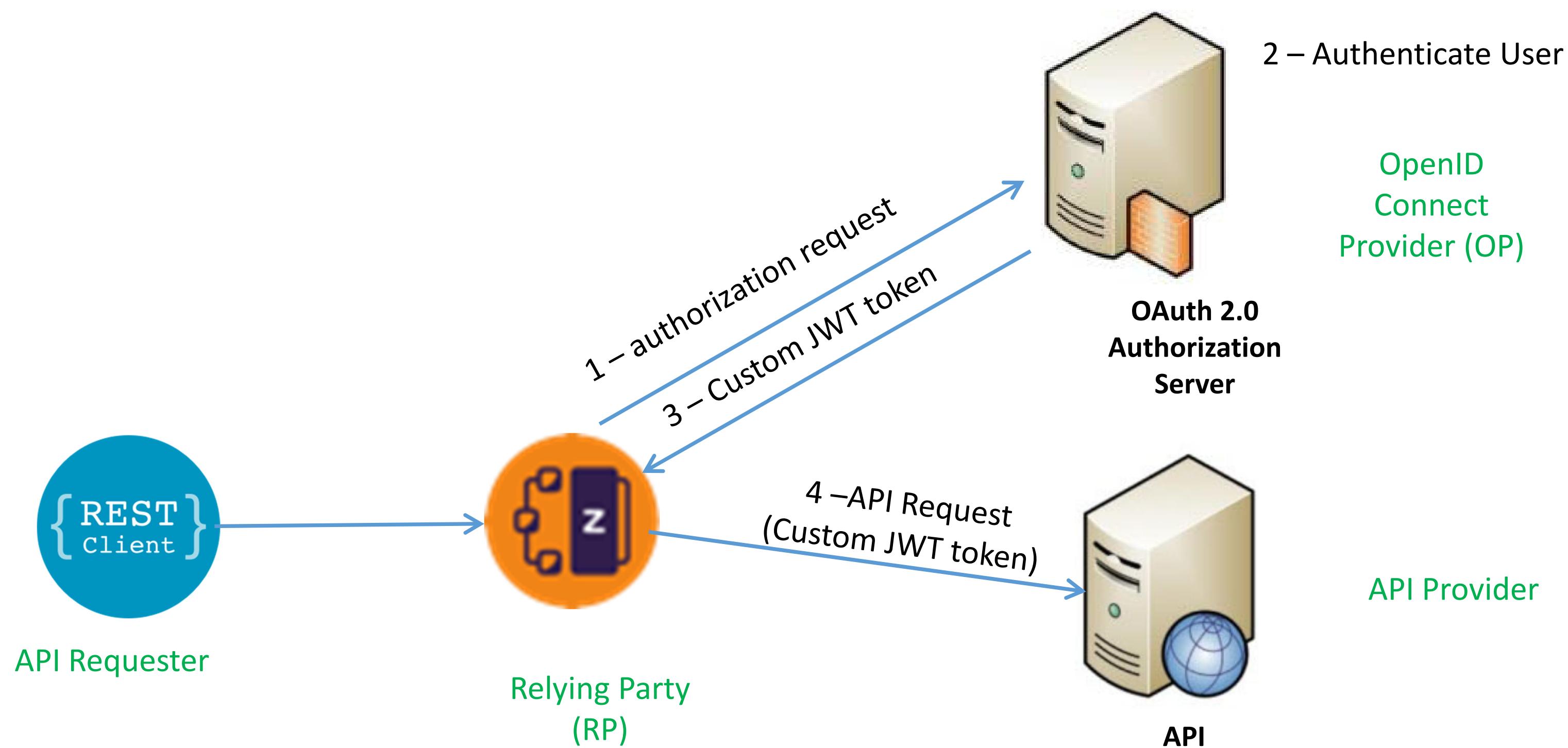
Calling an API with using a JWT custom flow

- ❑ In a non-OAuth 2.0 scenario, a JWT token is used in a custom flow, for example:
 - When you need to specify the HTTP verb that is used in the request to the authentication server.
 - When you need to specify how the JWT is returned from the authentication server (for example, in an HTTP header or in a custom field in a JSON response message).
 - When you need to use a custom header name for sending the JWT to the request endpoint.





z/OS Connect OAuth Custom Flow





API Requester – JWT Custom flow

BAQRINFO copy book

```
wg31 master
File Edit Settings View Communication Actions Window Help
Menu Utilities Compilers Help
BROWSE ZCEE30.SBAQCDB(BAQRINFO) Line 0000000028 Col 001 080
Command ==> Scroll ==> PAGE
01 BAQ-REQUEST-INFO.
  03 BAQ-REQUEST-INFO-COMP-LEVEL PIC S9(9) COMP-5 SYNC VALUE 4.
  03 BAQ-REQUEST-INFO-USER.
    05 BAQ-DAUTH.
      07 BAQ-DAUTH-USERNAME          PIC X(256).
      07 BAQ-DAUTH-USERNAME-LEN      PIC S9(9) COMP-5 SYNC
                                      VALUE 0.
      07 BAQ-DAUTH-PASSWORD         PIC X(256).
      07 BAQ-DAUTH-PASSWORD-LEN     PIC S9(9) COMP-5 SYNC
                                      VALUE 0.
      07 BAQ-DAUTH-CLIENTID        PIC X(256).
      07 BAQ-DAUTH-CLIENTID-LEN     PIC S9(9) COMP-5 SYNC
                                      VALUE 0.
      07 BAQ-DAUTH-CLIENT-SECRET   PIC X(256).
      07 BAQ-DAUTH-CLIENT-SECRET-LEN PIC S9(9) COMP-5 SYNC
                                      VALUE 0.
      07 BAQ-DAUTH-SCOPE-PTR       USAGE POINTER.
      07 BAQ-DAUTH-SCOPE-LEN        PIC S9(9) COMP-5 SYNC
                                      VALUE 0.
    05 BAQ-AUTHTOKEN.
      07 BAQ-TOKEN-USERNAME         PIC X(256).
      07 BAQ-TOKEN-USERNAME-LEN     PIC S9(9) COMP-5 SYNC
                                      VALUE 0.
      07 BAQ-TOKEN-PASSWORD        PIC X(256).
      07 BAQ-TOKEN-PASSWORD-LEN     PIC S9(9) COMP-5 SYNC
                                      VALUE 0.
  05 BAQ-ZCON-SERVER-URI        PIC X(256)
                                      VALUE SPACES.
MA A
Connected to remote server/host wg31z using lu/pool TCP00145 04/015
```

COBOL application

```
MOVE "ATSTOKENUSERNAME" to envVariableName.
PERFORM CALL-CEEENV THRU CALL-CEEENV-END
MOVE VAR(1:valueLength) to BAQ-TOKEN-USERNAME
MOVE valueLength TO BAQ-TOKEN-USERNAME-LEN
MOVE "ATSTOKENPASSWORD" to envVariableName.
PERFORM CALL-CEEENV THRU CALL-CEEENV-END
MOVE VAR(1:valueLength) to BAQ-TOKEN-PASSWORD
MOVE valueLength to BAQ-TOKEN-PASSWORD-LEN
```

Note that this example is using environment variables to provide token credentials, as documented in the z/OS Connect Advanced Topics Guide.



Configuring JWT Custom flow

```
<zosconnect_endpointConnection id="cscvincAPI"
    host="http://wg31.washington.ibm.com" port="9080"
    authenticationConfigRef="myJWTConfig"/>

<zosconnect_authToken id="myJWTConfig" authServerRef="myJWTServer"
    header="myJWT-header-name"
    <tokenRequest/>      See next slide
    <tokenReponse/>      See next slide
</zosconnect_authToken>

<zosconnect_authorizationServer id="myJWTServer"
    tokenEndpoint=https://wg31.washington.ibm.com:59443/oidc/endpoint/OP/token1
    basicAuthRef="tokenCredential" 2
    sslCertsRef="OutboundSSLSettings" />

<zosconnect_authData id="tokenCredential" 2
    user="zCEEClient" password="secret"/>
```

¹See URL https://www.ibm.com/support/knowledgecenter/SS7K4U_liberty/com.ibm.websphere.wlp.zseries.doc/ae/twlp_oidc_token_endpoint.html

² These credentials can be specified by the application



Configuring Custom JWT flow

Request Token Example 1

```
<tokenRequest  
    credentialLocation="header"  
    header="Authorization"  
    requestMethod="GET" />
```

Response Token

```
<tokenResponse  
    tokenLocation="header"  
    header="JWTAuthorization" />
```

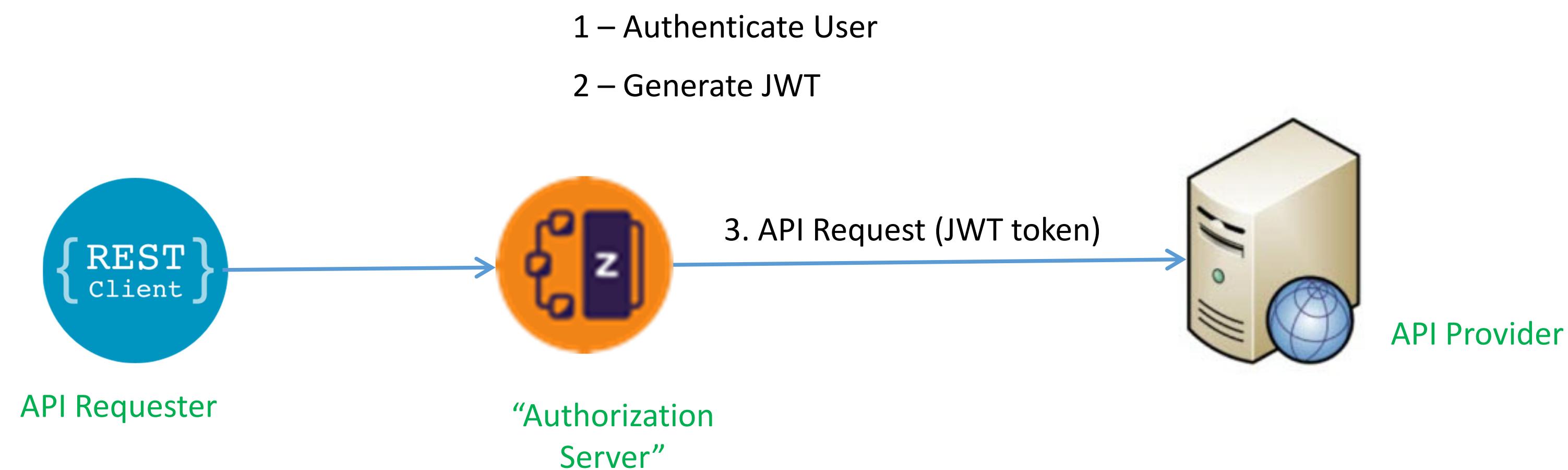
Response Token Example 2

```
<tokenRequest credentialLocation="body"  
    requestMethod="POST"  
    // Use XML escaped characters in requestBody  
    requestBody="
```

Response Token

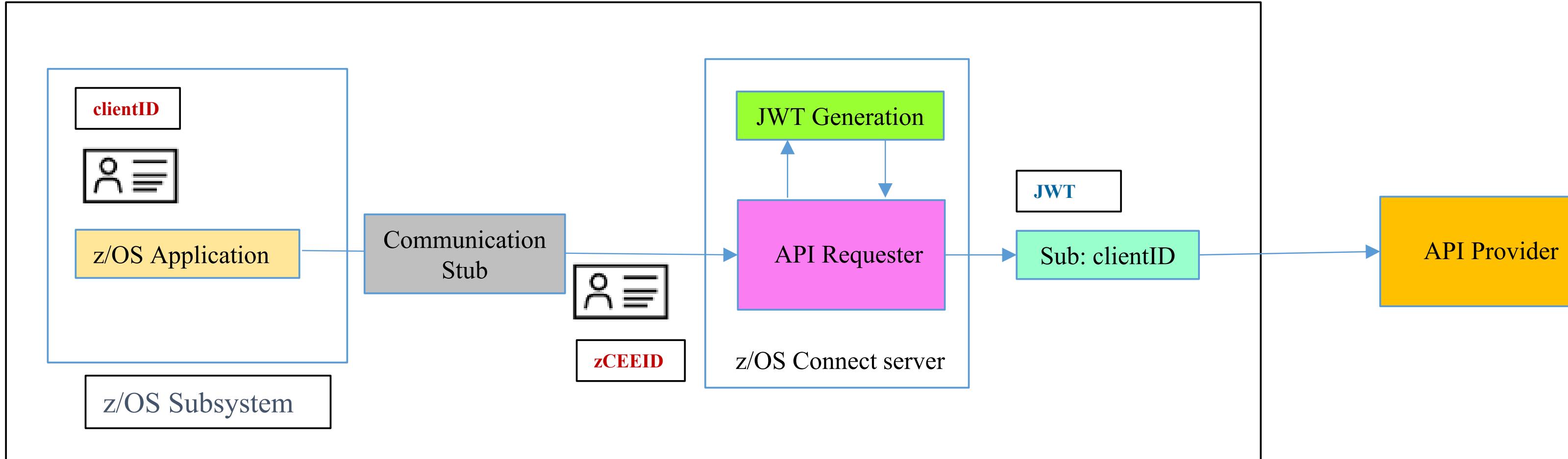
```
<tokenResponse  
    tokenLocation="body"  
    responseFormat="JSON"  
    tokenPath=".tokenname" />
```

z/OS Connect JWT Generation – V3.0.43





API Requester – JWT Generation



zCEEID – The identity that is used for authenticating connectivity the z/OS subsystem to the zCEE server. It is configured using basic authentication or for CICS, TLS client authentication.

clientID – the identity under which the z/OS application is executing.

- For CICS, the task owner
- For IMS, the transaction owner
- For batch, the job owner

requireAuth	idAssertion	Actions performed by z/OS Connect
true	ASSERT_SURROGATE	Identity assertion is enabled. The zCEE server authenticates <i>zCEEID</i> and checks whether <i>zCEEID</i> is a surrogate of <i>clientID</i> . If <i>zCEEID</i> is a surrogate of <i>clientID</i> , the server further checks whether <i>clientID</i> has the authority to invoke an API requester; otherwise, a BAQR7114E message occurs.
	ASSERT_ONLY	Identity assertion is enabled. The zCEE server authenticates <i>zCEEID</i> and directly checks whether <i>clientID</i> has the authority to invoke an API requester
false	ASSERT_SURROGATE	Identity assertion is enabled. The zCEE server checks whether <i>clientID</i> has the authority to invoke an API requester, and a warning message occurs to indicate that the ASSERT_ONLY value is used instead of the ASSERT_SURROGATE value.
	ASSERT_ONLY	Identity assertion is enabled. The zCEE server checks whether <i>clientID</i> has the authority to invoke an API requester



JWT generation requires setting a program control extended attribute

As root or superuser, set the *libifaedjreg64.so* program control extended attribute bit

- *Permit the server's identity to the required FACILITY resource*

PERMIT BPX.SERVER CLASS(FACILITY) ID(LIBSERV**) ACCESS(READ)**

SETROPTS RACLIST(FACILITY) REFRESH

- *Define a SURROGAT profile for the asserted identity and permit access to connection identity*

RDEFINE SURROGAT **clientID.BAQASSRT UACC(NONE) OWNER(SYS1)**

PERMIT **clientID.BAQASSRT CLASS(SURROGAT) ACCESS(READ) ID(**zCEEID**)**

OR

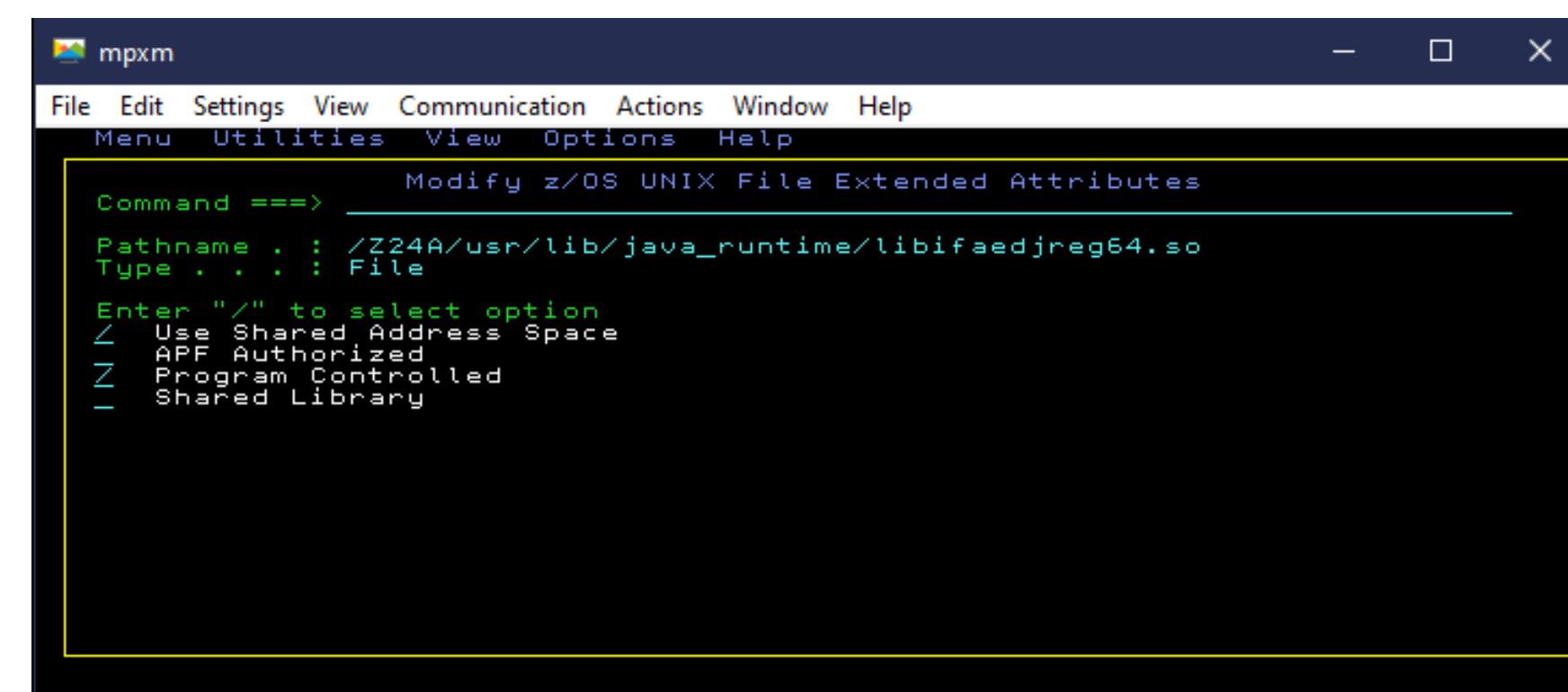
RDEFINE SURROGAT *.BAQASSRT UACC(NONE) OWNER(SYS1)

PERMIT *.BAQASSRT CLASS(SURROGAT) ACCESS(READ) ID(zCEEID**)**

SETROPTS RACLIST(SURROGAT) REFRESH

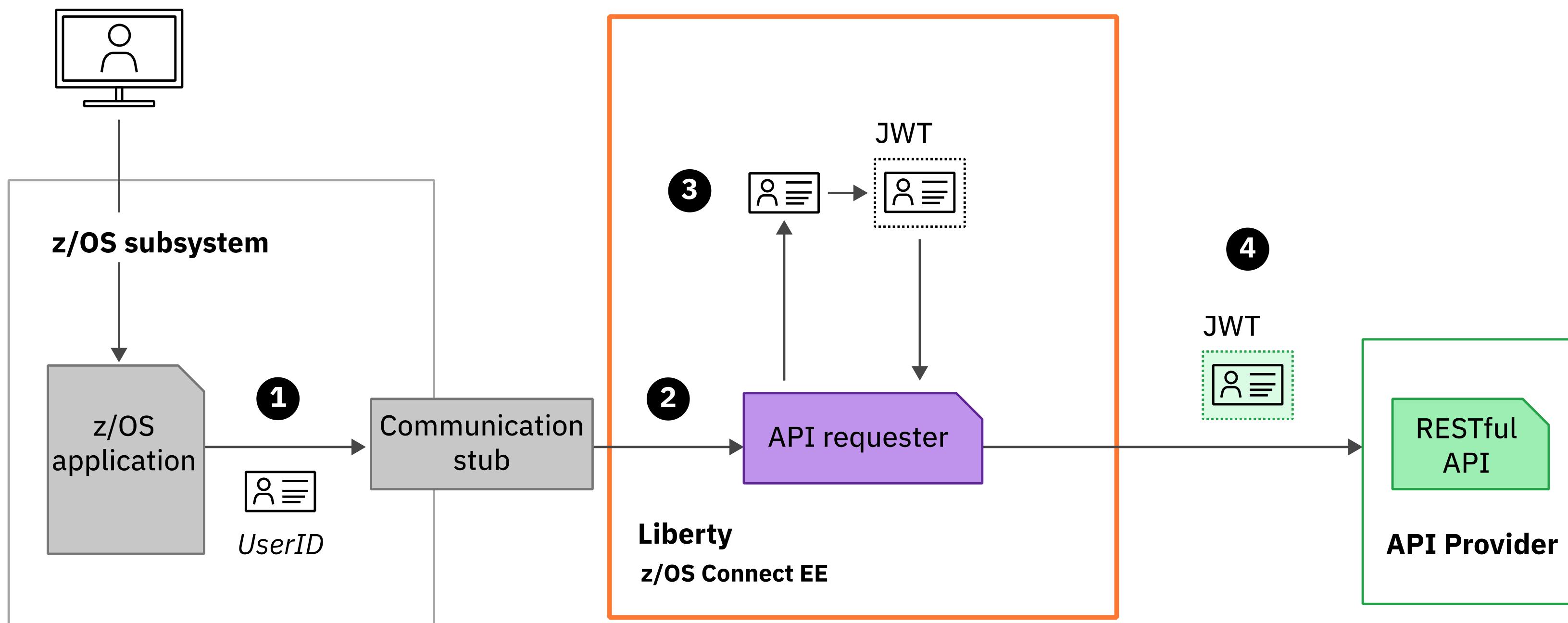
- *Enable the program control bit for Java shared object ifaedjreg64*

```
su  
cd /usr/lib/java_runtime  
extattr +p libifaedjreg64.so
```





JWT Generation



- 1 Communication stub extracts the ID from the application environment
- 2 z/OS Connect generates a JWT token containing the z/OS application asserted user ID
- 3 The JWT is used to authorise the request to the API endpoint



Configuring JWT Generation support

```
<zosconnect_endpointConnection id="conn"  
    host="http://api.server.com" port="8080"  
    authenticationConfigRef="jwtConfig" />  
  
<zosconnect_authTokenLocal id="jwtConfig"  
    tokenGeneratorRef="jwtBuilder"  
    header="Authorization" >  
    <claims>{ "name":"JohnSmith,  
        "ID":"1234567890" }  
    </claims>  
  
<jwtBuilder id="jwtBuilder"  
    scope="scope1"  
    audiences="myApp1"  
    jti="true"  
    signatureAlgorithm="RS256"  
    keyStoreRef="myKeyStore"  
    keyAlias="jwtSigner"  
    issuer="z/OS Connect EE Default"/>
```

One or more Public claim (e.g., *aud,exp,nbf,iat,jti*) or
one or more private claims

The "sub" claim value will be application asserted user ID.



z/OS Connect Wildfire Github Site

<https://ibm.biz/BdPRGD>

The screenshot displays three GitHub repository pages, each featuring a red oval highlighting specific file uploads:

- Top Repository (zCONNEE-Wildfire-Workshop / OpenAPI2):** Shows three PDF files: "Developing CICS API Requester Applications.pdf", "Developing IMS API Requester Applications.pdf", and "Developing MVS Batch API Requester Applications.pdf".
- Middle Repository (zCONNEE-Wildfire-Workshop / APIRequesters):** Shows three PDF files: "Developing CICS API Requester Applications.pdf", "Developing IMS API Requester Applications.pdf", and "Developing MVS Batch API Requester Applications.pdf".
- Bottom Repository (zCONNEE-Wildfire-Workshop):** Shows several PDF files, including "ZCREQUEST - Introduction to zOS Co...", "zos Connect FF V3 Advanced Topics ...", and "zos Connect EE V3 Getting Started.pdf". A red oval highlights the first few items.

Left Panel (Code View):

- Master branch: 1 branch, 0 tags
- Commits:
 - emitchj Add files via upload (8e503b9)
 - AdminSecurity Delete ZC2OMVS2.jcl
 - OpenAPI2 Delete Developing
 - cobol Add files via upload
 - xml Add files via upload
 - README.md Update README.md
 - ZCADMIN - zOS Connect Administrat... Add files via upload
 - ZCESEC - zOS Connect Security.pdf Add files via upload
 - ZCINTRO - Introduction to zOS Conn... Add files via upload
 - ZCREQUEST - Introduction to zOS Co... Add files via upload
 - zos Connect FF V3 Advanced Topics ... Add files via upload
 - zos Connect EE V3 Getting Started.pdf Add files via upload
- README.md

This repository contains material from the z/OS Connect EE Wildfire workshops run by the I Center. It is should be referenced frequently for updates to the presentations, exercises, sam

- Contact your IBM representative to schedule access to these exercises

WSC wants your feedback!

What you will see:

From: IBM Client Feedback <ibm@feedback.ibm.com>

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



Dear

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

You can begin the survey by answering this question.

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



Sincerely,

IBM Advocacy Team

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



Thank you for listening and your questions.

And thank you for completing the Medallia survey

mitchj@us.ibm.com

© 2018, 2023 IBM Corporation
Slide 125