



Accessing REST APIs from z/OS using IBM z/OS Connect

Mitch Johnson mitchj@us.ibm.com

John Brefach John.J.Brefach@ibm.com

Jeff Bisti Jbisti@us.ibm.com

Washington System Center



Agenda

- What is REST and what are REST APIs?
- Using a z/OS Connect API requester to access a REST API
- General API requester COBOL client programming considerations
- Developing API requesters for Swagger 2.0 REST APIs
- Developing API requesters for OpenAPI 3 REST APIs
- Configuration and Security considerations for API requesters

Notes and Disclaimers



- There will be additional information on slides that will be designated as Tech/Tips. These contain information that hopefully will be useful to the reader.
- **IBM z/OS Connect (Swagger 2.0)** refers the z/OS Connect EE product prior to service level V3.0.55. **IBM z/OS Connect (OpenAPI 3)** refers to the additional functions and features added with service level V3.0.55. Important - servers configured for OpenAPI 2 can will continue to operate as is with service level V3.0.55 and later.
- A z/OS Connect Swagger 2.0, icon or a z/OS Connect OpenAPI 3 icon will appear on slides where the information is specific to these products. Don't hesitate to ask questions as to why the icon does or does not appear on certain slides
- A Liberty icon will appear on slides where the information both products. Don't hesitate to ask questions as to why the icon does or does not appear on certain slides.

Note: For our purposes, the terms OpenAPI 2.0 and Swagger 2.0 are interchangeable.



Accessing z/OS resources with REST APIs using IBM z/OS Connect

Mitch Johnson mitchj@us.ibm.com

John Brefach John.J.Brefach@ibm.com

Washington System Center

mitchj@us.ibm.com



WebSphere Liberty Profile Administration

Managing WebSphere Liberty Profile servers for IBM z/OS Connect (OpenAPI 2 and OpenAPI 3), IBM MQ REST Console and zOSMF

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



© 2017, 2023 IBM Corporation
Slide 1



z/OS Connect OpenAPI 3

Designer and z/OS Native server Experiences and Observations

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



mitchj@us.ibm.com

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

© 2018, 2024 IBM Corporation
Page 4

z/OS Connect Github Site

<https://ibm.biz/zCEEWorkshopMaterial>



Screenshot of the GitHub repository page for [zCONNEE-Wildfire-Workshop](#).

The repository is public and contains 1 branch and 0 tags. It has 635 commits from user [emitchj](#). A red oval highlights the first commit, which is an upload of files.

Commit	Author	Message	Date
Add files via upload	emitchj	Add files via upload	d1c0657 last week
APIRequesters		Add files via upload	2 months ago
AdminSecurity		Delete AdminSecurity/ctl directory	2 months ago
COBOL Samples		Add files via upload	5 months ago
JCL Samples		Add files via upload	2 months ago
OpenAPI2		Add files via upload	2 months ago
OpenAPI3		Add files via upload	last week
XML Samples		Add files via upload	last year
archive		Add files via upload	last month
README.md		Update README.md	2 years ago
ZCADMIN - zOS Connect Administrat...		Add files via upload	2 months ago
ZCEESEC - zOS Connect Security.pdf		Add files via upload	last year
ZCINTRO - Introduction to zOS Conn...		Add files via upload	last month
ZCREQUEST - Introduction to zOS Co...		Add files via upload	last month
zOS Connect EE V3 Advanced Topics ...		Add files via upload	2 years ago
zOS Connect EE V3 Getting Started.pdf		Add files via upload	2 years ago

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

[Readme](#) [Activity](#) [22 stars](#) [16 watching](#) [16 forks](#) [Report repository](#)

Releases
No releases published [Create a new release](#)

Packages
No packages published [Publish your first package](#)

Deployments 1
[github-pages](#) 2 years ago

IBM's Z Virtual Access (zVA) provides access to hand-on exercises



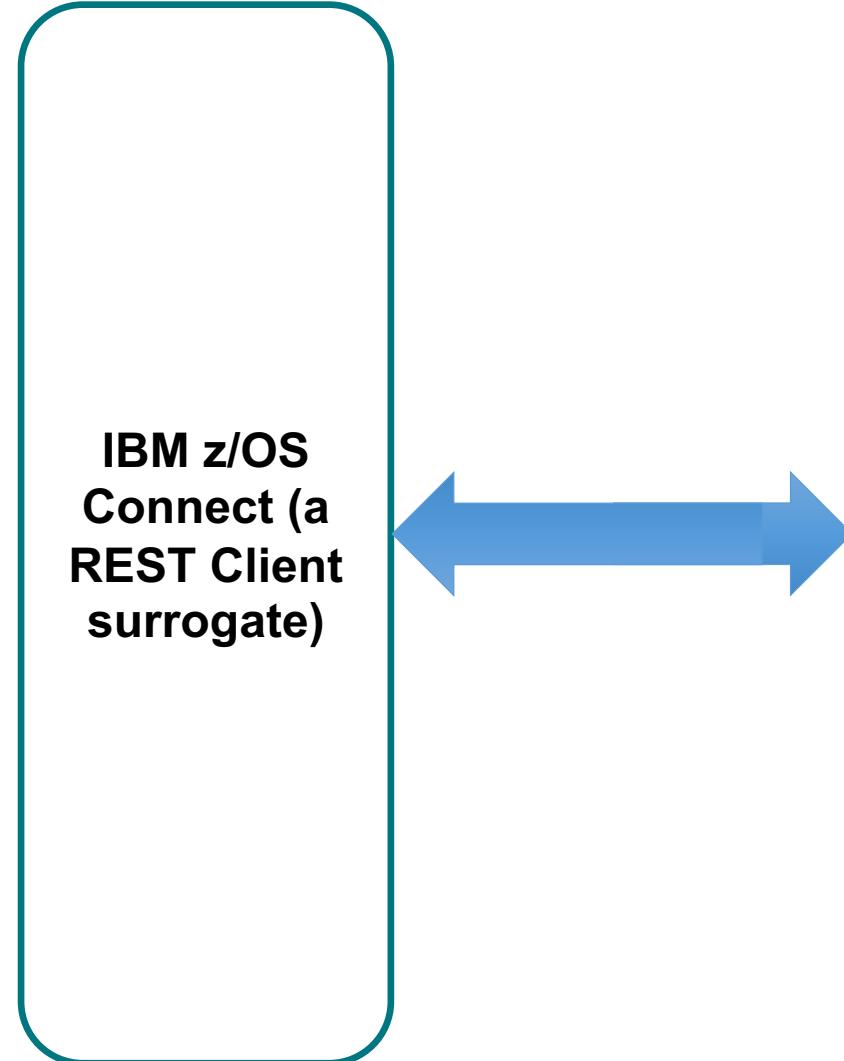
The screenshot shows a Windows desktop environment within the zVA interface. The desktop background is blue with a glowing window effect. A command prompt window is open in the foreground, displaying the following text:

```
c:\ Select Administrator: Command Prompt
6 File(s) 16,849 bytes
4 Dir(s) 54,848,405,504 bytes free

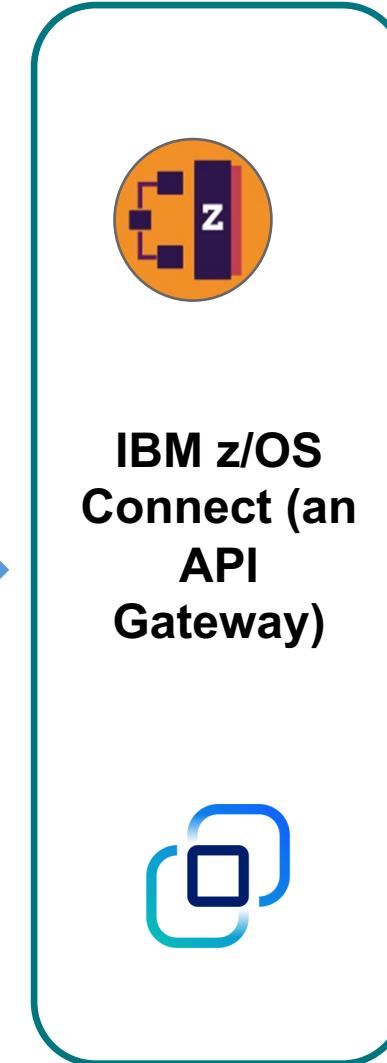
c:\z\apiRequester\cscvinc>cscvinc
c:\z\apiRequester\cscvinc>c:\z\software\zconbt\bin\zconbt.bat -p=../cscvinc.properties -f=../cscvinc.ara
BAQ0000I: z/OS Connect Enterprise Edition 3.0 Build Toolkit Version 1.14 (20230417-1402).
BAQ00008I: Creating API requester archive from configuration file ./cscvinc.properties.
BAQ0040I: The generated API requester is automatically named cscvinc_1.0.0 based on the title cscvinc and version 1.0.0
of the API to be called.
BAQ0049W: Specifying a header parameter with the name Authorization is not supported, the header parameter is ignored.
BAQ0049W: Specifying a header parameter with the name Authorization is not supported, the header parameter is ignored.
BAQ0025W: Multiple HTTP status codes are not supported in the response of operation (operationId: getCscvincSelectService,
relativePath: /employee/{employee}, method: GET). 200 used and 404 ignored by the build toolkit.
BAQ0049W: Specifying a header parameter with the name Authorization is not supported, the header parameter is ignored.
BAQ0025W: Multiple HTTP status codes are not supported in the response of operation (operationId: putCscvincUpdateService,
relativePath: /employee/{employee}, method: PUT). 200 used and 400 ignored by the build toolkit.
BAQ0049W: Specifying a header parameter with the name Authorization is not supported, the header parameter is ignored.
BAQ0015I: Start processing operation (operationId: postCscvincInsertService, relativePath: /employee, method: POST).
DFHPI9586W A reserved word "address" has been detected in the input document, it has been changed to "Xaddress".
DFHPI9586W A reserved word "date" has been detected in the input document, it has been changed to "Xdate".
BAQ0016I: Successfully processed operation (operationId: postCscvincInsertService, relativePath: /employee, method: POST).
BAQ0015I: Start processing operation (operationId: getcscvincSelectService, relativePath: /employee/{employee}, method:
GET).
DFHPI9586W A reserved word "address" has been detected in the input document, it has been changed to "Xaddress".
DFHPI9586W A reserved word "date" has been detected in the input document, it has been changed to "Xdate".
BAQ0016I: Successfully processed operation (operationId: getcscvincSelectService, relativePath: /employee/{employee}, m
ethod: GET).
BAQ0015I: Start processing operation (operationId: putCscvincUpdateService, relativePath: /employee/{employee}, method:
```

The desktop icons include This PC, Ikeyman, IMS, IBM, Google Chrome, Exercises, Command Prompt, Firefox, Recycle Bin, LibreOffice 7.4, IBM Explorer for zOS, CopyPaste Files, wg31, Postman, and MQ Explorer. A PDF file named 'Important--- Me.pdf' is visible on the desktop.

z/OS Connect “API requester support” exposes external REST APIs in the “cloud” to z/OS applications



Note that z/OS Connect also supports accessing
z/OS resources from clients in the “cloud” via RESTful APIs



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

+ HCL and Rocket Software

*Other Vendors or your own implementation

Let's start by defining REST

/what_is_REST?

And what makes an API “RESTful”?



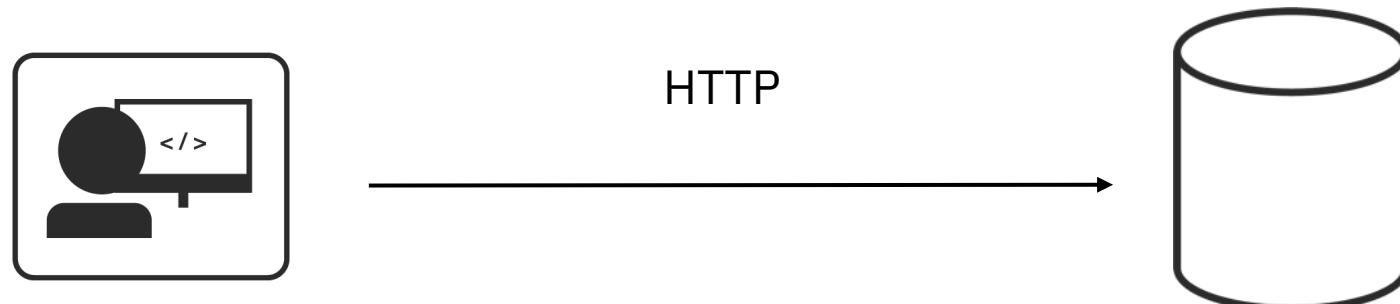
REST is architectural programming style

REST is an acronym standing for **R**epresentational **S**tate **T**ransfer*

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

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

Providing a simple and intuitive programming style 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.



REST APIs follow two key principles

REST APIs use a HTTP verb to specify the operations, e.g.,

Create (POST), **R**ead(GET),
Update(PUT), **D**elete(DELETE),
for more information on

CRUD, see URL

https://en.wikipedia.org/wiki/Create,_read,_update_and_delete

POST
GET
PUT
DELETE

Uniform Resource Identifier (URI) path identifies the resource (s)

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

Use **Path** and **Query** parameters to refine the request

Uniform Resource Locator (URL) identifies the protocol, host and port and includes the URI path

REST APIs use JSON
(**J**a**s** **S**cript **O**bject
Notation) to represent the
data object in the
Request/Response
message bodies

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" }



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>



Let's start with 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.

For more information see, https://en.wikipedia.org/wiki/OpenAPI_Specification



There are a number of tools available to aid in the creation and consumption of APIs

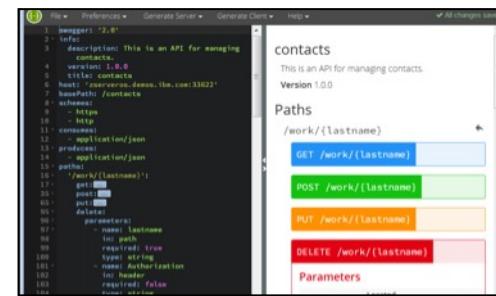
Code Generation* - create stub code, to consume APIs from various languages, e.g., *Java data beans, COBOL copy books that describe the request and response messages.*



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, Liberty, etc.

<https://swagger.io/>
<https://jsonformatter.org/>

Important - You may have used or heard of the term Swagger with the use of APIs. Swagger was the original name of the specification but in 2016 the Swagger specification was renamed the OpenAPI Specification (OAS).



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

The OpenAPI Specification (OAS) is the industry standard framework for describing REST APIs



- **Swagger 2.0 (more formally known as OpenAPI Specification 2) was supported initially by z/OS Connect**

Accessing z/OS resources was the only goal when developing APIs .The interactions with the z/OS resources was driven by the layout of the CICS COMMAREA or CONTAINER, the IMS or MQ messages or the Db2 REST service.

- The details of the interactions with the z/OS resource determined the contents of the API request and response messages and the subsequent specification document.
- **z/OS Connect produces the specification document that describes the methods and request and response messages.**



- **OpenAPI Specification 3 (OAS3), supported by z/OS Connect starting in March 2022, service V3.0.55**

As companies mature their API strategy, they begin to introduce API governance boards to drive consistency in their API design. As more public APIs are created, government and industry standards bodies begin to regulate and drive for standardization. This drives the need for “API first” functional mapping capabilities within the integration platform. The external API design determined the layouts of the API request and response messages provided by the specification documents which was consumed by z/OS Connect to describe the z/OS resource interactions.

- The API details of the methods and layouts of request and response messages are provided in advance and access to the z/OS resource is driven by the API design
- **z/OS Connect consumes the specification document that describes the methods and request and response messages**

Note: For our purposes, the terms OpenAPI 2.0 and Swagger 2.0 are interchangeable.

Also, the term OAS3 is a recognized shortcut for OpenAPI3 while OAS2 is not for OpenAPI2.



Contrast the Swagger 2.0 versus OpenAPI 3 specification

z/OS Connect
OpenAPI2 tooling
produces an
Swagger 2.0
specification
document (aka
Swagger 2.0.0),
where the details of
the methods and
request/response
messages in the API
specification are
driven by the nature
of the z/OS resource
(JSON format).

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

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

```

z/OS Connect
OpenAPI3 tooling
consumes an
OpenAPI3 specification
document (aka OSA3)
and the details of the
methods and
request/response
messages are driven by
the API specification
(YAML format*) and
not the nature of the
z/OS resource. Also,
JSONata can be
used to augment
the API response

*Yet Another Markup Language

Tech-Tip: A detailed look at an OPENAPI specification documentation



The screenshot shows an IDE window with the file 'cscvinc.yaml' open. The code defines a 'get' operation for the endpoint '/employee/{employee}'. It includes a 'tags' section with 'Employee', an 'operationId' of 'getEmployeeSelectService', and a 'x-ibm-zcon-roles-allowed' header with 'Staff'. The 'parameters' section contains two items: 'Authorization' (header, required false) and 'employee' (path, required true). The 'responses' section defines three status codes: '200' (OK), '404' (Not Found), and '500' (Severe Error). The '200' response is detailed with a schema reference to '#/components/schemas/getEmployeeSelectService_response_200'. The '404' and '500' responses also have schema references. The 'put' operation is also partially defined.

Annotations:

- URI Path/Method**: Circles the top part of the 'get' operation definition.
- URI Path Parameters**: Circles the 'parameters' section.
- Response Codes**: Circles the 'responses' section.

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

The screenshot shows an IDE window with the file 'cscvinc.yaml' open. It focuses on the 'getEmployeeSelectService_response_200' schema. This schema is an object with properties for 'summary' (referencing '#/components/schemas/getEmployeeSelectService_response_200_message') and 'detail' (referencing '#/components/schemas/getEmployeeSelectService_response_200_detail'). It also includes a 'getEmployeeSelectService_response_200_message' schema (string type) and a 'getEmployeeSelectService_response_200_detail' schema (EmployeeSelectServiceOperationResponse object). The EmployeeSelectServiceOperationResponse schema has properties for 'employeeData' (object) and 'response' (object). The 'employeeData' schema has properties for 'employeeDetails' (object) and 'name' (string, max length 20). The 'response' schema has properties for 'employeeNumber' (string, max length 6), 'name' (string, max length 20), 'address' (string, max length 20), and 'phoneNumber' (string, max length 8).

Annotations:

- Response message layout**: Circles the top part of the schema definition.
- Response message fields**: Circles the detailed properties of the schemas.

```
getEmployeeSelectService_response_200:  
  type: object  
  properties:  
    summary:  
      $ref: '#/components/schemas/getEmployeeSelectService_response_200_message'  
    detail:  
      $ref: '#/components/schemas/getEmployeeSelectService_response_200_detail'  
  getEmployeeSelectService_response_200_message:  
    type: object  
    properties:  
      message:  
        type: string  
    example:  
      message: record retrieved  
  getEmployeeSelectService_response_200_detail:  
    type: object  
    properties:  
      EmployeeSelectServiceOperationResponse:  
        type: object  
        properties:  
          employeeData:  
            type: object  
            properties:  
              employeeDetails:  
                type: object  
                properties:  
                  employeeNumber:  
                    type: string  
                    maxLength: 6  
                  name:  
                    type: string  
                    maxLength: 20  
              address:  
                type: string  
                maxLength: 20  
              phoneNumber:  
                type: string  
                maxLength: 8
```



One of the goals of REST is to have 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);
    }
} catch (IOException e) {
    e.printStackTrace();
}
bufferReader.close();
conn.disconnect();
```

```
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);
    }
} catch (IOException e) {
    e.printStackTrace();
}
bufferReader.close();
conn.disconnect();
```

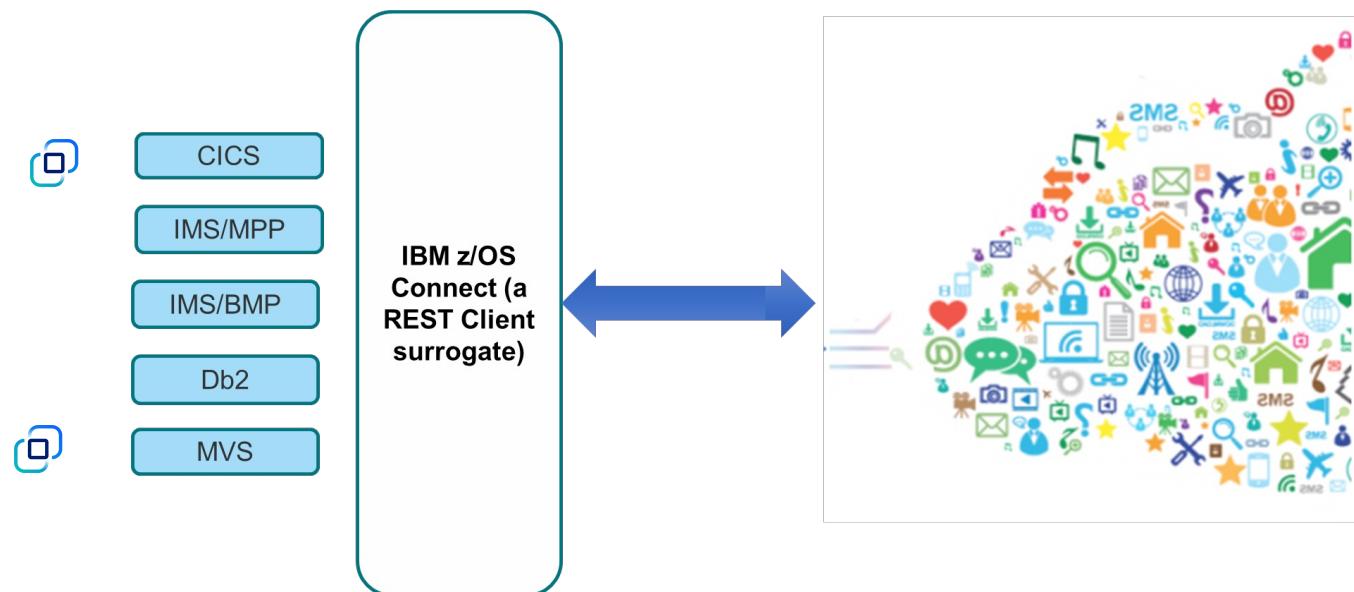
In these examples, these Java clients have the same application programming pattern.

- They provide an URL, specify a method and provide a request message.
- Then use code to send a request to the API provider using HTTP.
- A response message is returned from 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 knowing the infrastructure or coding for ECI, JDBC, OTMA, JMS, J2C, etc.



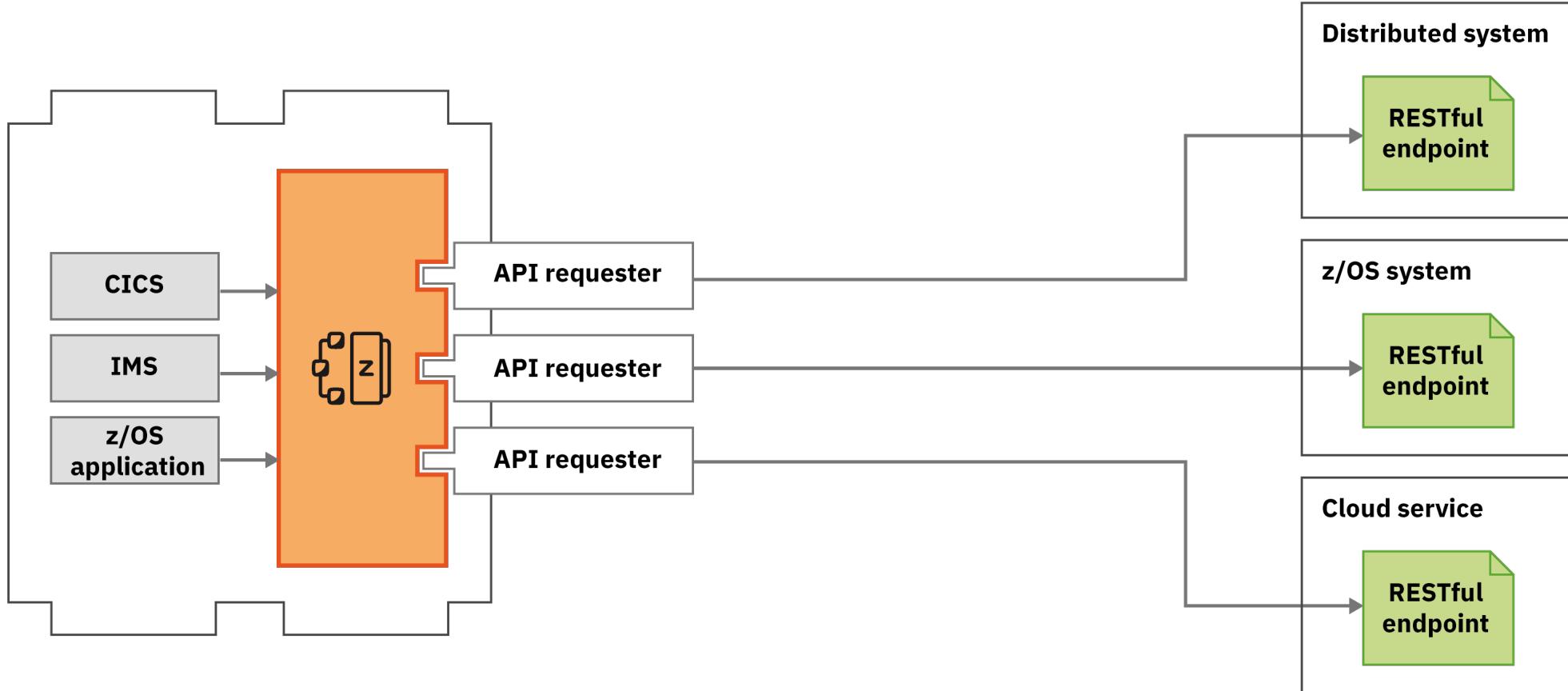
Using an z/OS Connect API requester to access a REST APIs

Developing COBOL API Requester applications

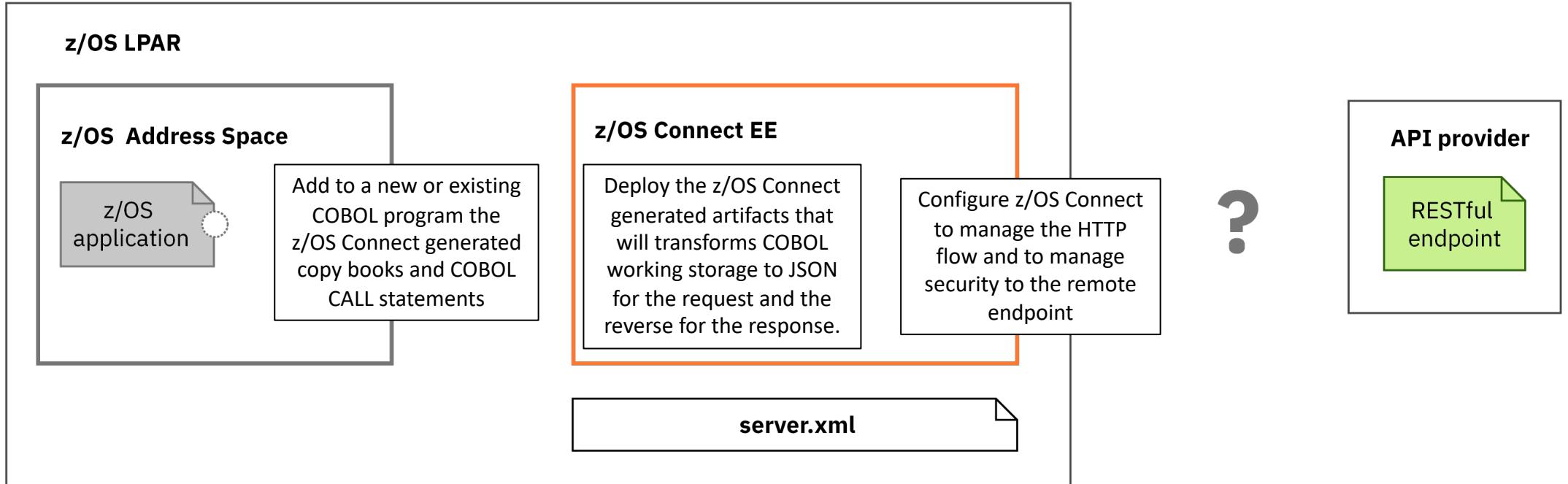




z/OS Connect tooling generates the artifacts used to invoke APIs from z/OS application



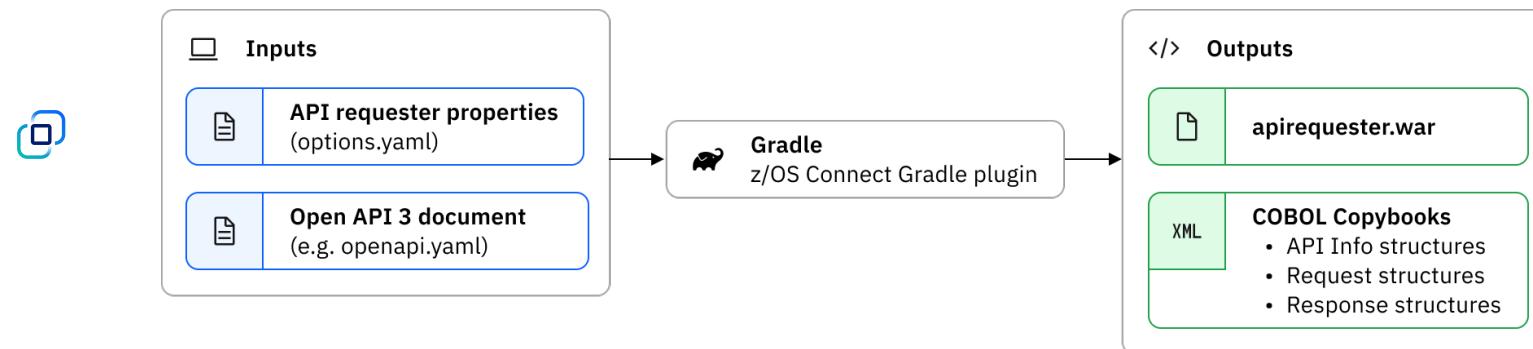
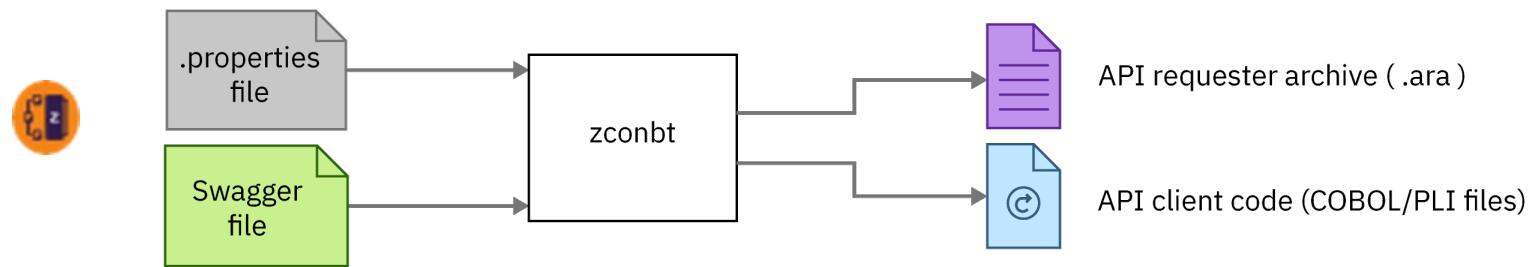
A detailed look at the steps required to invoke a remote API



Development starts with the OAS specification document of the API



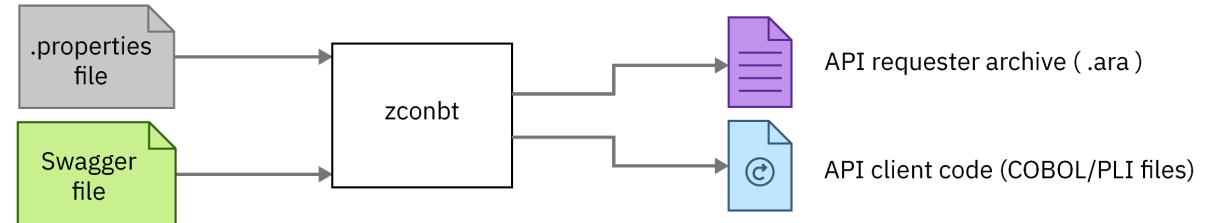
z/OS Connect tooling consuming the API's specification description to generate COBOL copy books and an API requester archive (ARA) file for an Swagger 2.0 API or a web archive (WAR) file for an OpenAPI 3 API.



Use the z/OS Connect build toolkit (zconbt) for APIs described using Swagger 2.0



The screenshot shows a browser window displaying a Swagger 2.0 JSON API definition. The URL is `/C:/apiRequester/cscvinc/cscvinc`. The JSON structure includes fields like `swagger`, `info`, `schemes`, `consumes`, `produces`, and `paths`. A specific path `/employee/{employee}` is expanded, showing a `get` method with parameters `employee` and `path`, and a response `200` with a schema.



properties file#

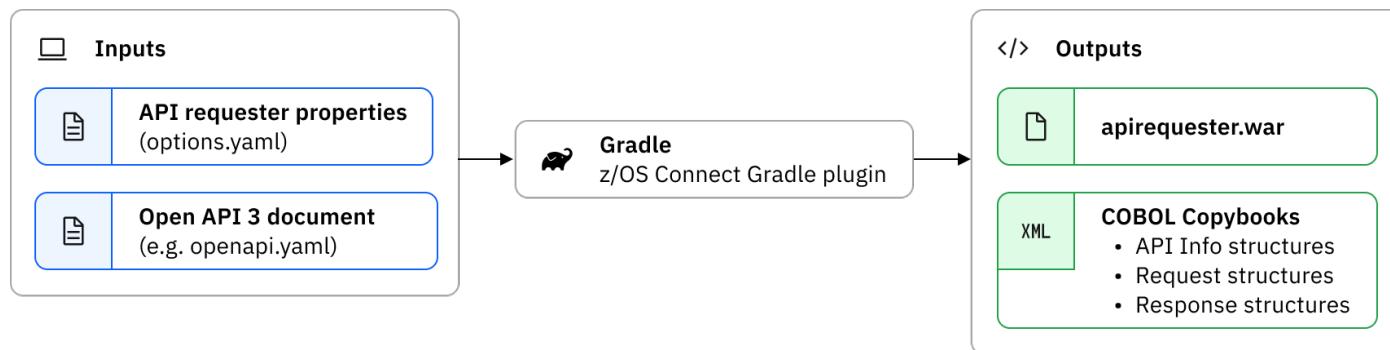
```
apiDescriptionFile=./cscvinc.json
dataStructuresLocation=./syslib
apiInfoFileLocation=./syslib
logFileDirectory=./logs
language=COBOL
connectionRef=cscvincAPI
requesterPrefix=csc
```

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

Use the z/OS Connect Gradle plug-in for APIs describe using OpenAPI 3



```
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 tools 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 Swagger 2.0 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 this null messages must be represented by 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.

Tech-Tip: BTW, the Gradle toolkit can be executed on z/OS

```
//*****  
/* SET SYMBOLS  
*****  
//EXPORT EXPORT SYMLIST=(*  
// SET JAVAHOME='/usr/lpp/java/J8.0_64'  
*****  
/* Step ZCEEGRDL - Use Gradle command to create artifacts  
*****  
//ZCEESRVR EXEC PGM=IKJEFT01,REGION=0M  
//SYSTSPRT DD SYSOUT=*  
//SYSERR DD SYSOUT=*  
//STDOUT DD SYSOUT=*  
//SYSTSIN DD *,SYMBOLS=EXECSYS  
BPXBATCH SH +  
export JAVA_HOME=&JAVAHOME; +  
cd gradle/cscvinc; +  
gradle build
```



COBOL client programming considerations for both Swagger 2.0 and OpenAPI 3 APIs

Let's look at what a COBOL equivalent program might look like



An example of using the COBOL JSON PARSE

```
mpz3
File Edit Settings View Communication Actions Window Help
File Edit Edit_Settings Menu Utilities Compilers Test Help
EDIT      USER1.ZCEE.SOURCE(TEST) - 01.01
Command ==>
000030      *****
000100      * In this example, input to JSON PARSE star
000200      * 1047 and then be converted to UTF-8 (code
000300      * Convert to specific codepages using the d
000400      * The first argument to display-of should b
000500      * which the COBOL compiler represents in UT
000600      *****
000700          Move function display-of(
000800              function national-of(
000900                  jtxt-1047-client-data) 1208)
001000          to jtxt-1208(1:function length(jtx
001100          Json parse jtxt-1208 into client-dat
001200          with detail
001300          suppress transactions
001400          not on exception
001500              display "Successful JSON Parse"
001600          end-json.
001800          Move 2 to txnum.
001900          Initialize jtxt-1208 all value.
002000          Move function display-of(
002100              function national-of(
002200                  jtxt-1047-transactions) 1208)
002300              to jtxt-1208(1:function length(jtx
002400          Json parse jtxt-1208 into transaction
002500          with detail
002600          name tx-price is 'tx-priceinUS$'
002700          not on exception
002800              display "Successful JSON Parse"
```

An example of invoking a Java class from COBOL

```
WG31 - 3270
File Edit Settings View Communication Actions Window Help
File Edit Edit_Settings Menu Utilities Compilers Test Help
EDIT      JOHNSON.PASSTCKT.SOURCE(ATSPKTJ) - 01.06
Command ==>
000049      Procedure division.
000050          Set address of JNIEnv to JNIEnvPtr
000051          Set address of JNICALLInterface to JNIEnv
000052          *-----
000053          * Redirect output to SYSOUT
000054          *-----
000055          * Invoke ZUtil "redirectStandardStreams"
000056          Perform JavaExceptionCheck.
000057          *-----
000058          * Use JNI services to convert appl to jstring
000059          *-----
000060          Call "NewStringPlatform"
000061              using by value JNIEnvPtr
000062                  address of zAppl address of jAppl 0
000063                  returning rc
000064          If rc not = zero then
000065              Display "Error occurred creating jAppl object"
000066              Stop run
000067          End-if
000068          *-----
000069          * Invoke method GetPassTicket in Java class
000070          * com.ibm.ats.GetPassTicket
000071          *-----
000072          * Invoke PassTicketClass "getPassTicket"
000073              using by value jIdentity jAppl returning jResponse.
000074          Perform JavaExceptionCheck
000075          *-----
000076          * Use JNI services to obtain size of response returned
000077          *-----
000078          Call "GetStringPlatformLength"
000079              using by value JNIEnvPtr
000080                  jResponse address of jResponseLen 0 returning rc
000081          If rc not = zero then
000082              Display "Error obtaining length of jResponse"
000083              Stop run
000084          End-if
000085          *-----
000086          * Use JNI services to convert jstring to string
000087          *-----
```



z/OS Connect performs the JSON transformations and the HTTP interactions

```
// 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());
```

```
// 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");
```

z/OS Connect provides same simple programming pattern for z/OS applications. The z/OS application provides an URI path, the method and a request message in working storage. z/OS Connect cod, encapsulated within a z/OS Connect server, that transforms COBOL working storage to and from JSON and interacts with end API endpoint, therefore there is a minimum need for new/changed COBOL code.

application is unaware of the underlying infrastructure. No dependencies on coding for ECI, JDBC, OTMA, JMS, J2C, etc.



First, be aware of COBOL working storage implications

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

```
/C:/apiRequester/ATS/ATSContactX
```

File:///C:/apiRequester/ATS/ATSContactPreferences

JSON Raw Data Headers

Save Copy Collapse All Expand All Filter JSON

```
maxItems: 10
communicationPreferences:
  items:
    $ref: "#/definitions/member-communication-preferences"
    type: "array"
  description: "Multiple member codes"
memberCodeableConcept:
  description: "Multiple member codes"
  items:
    $ref: "#/definitions/member-codeable-concept"
    type: "array"
    type: "object"
member-contacts-request:
  title: "Member Contacts Request"
  description: "Read-only request data to search for member contact information."
  properties:
    umi:
      description: "Unique Member Id. This value is at a contract level. All members under one contract have the same UMI."
      example: "122222444001"
      type: "string"
    firstName:
      description: "Member first name or given name."
      example: "Arthur"
      type: "string"
    lastName:
      description: "Member last name or family name."
      example: "Smith"
      type: "string"
    birthDate:
      description: "Member date of birth in the format mm/dd/yyyy."
      example: "12/19/2019"
      type: "string"
```

```
ATS01P01 - Notepad
```

File Edit Format View Help

```
* ++++++RespBody.+++++
```

```
06 RespBody.
```

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

```
09 memberContactsResponse OCCURS 255.
```

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

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

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

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

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

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

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

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

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

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



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

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

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

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

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



Tech-Tip: Consider adding explicit constraints to the properties

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

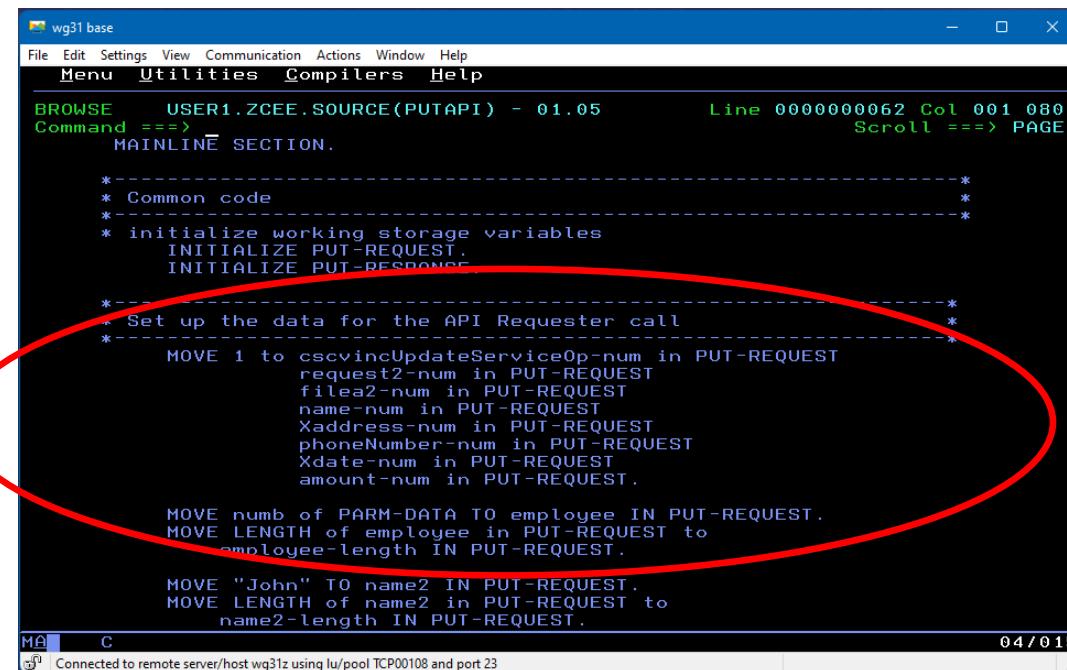
```
/C:/apiRequester/ATS/MemberCo X +  
file:///C:/apiRequester/ATS/MemberContactPrefe  
JSON Raw Data Headers  
Save Copy Collapse All Expand All Filter JSON  
member-communication-preferences  
type: "array"  
communicationPreferences:  
items:  
$ref: "#/definitions/member-communication-preferences"  
type: "array"  
maxItems: 10  
memberCodeableConcept:  
description: "Multiple member codes"  
items:  
$ref: "#/definitions/member-codeable-concept"  
type: "array"  
type: "object"  
member-contacts-request:  
title: "Member Contacts Request"  
description: "Read-only request data to search for member contact information."  
properties:  
umi:  
description: "Unique Member Id. This value is at a contract level. All members under one contract have the same UMI."  
example: "12222444001"  
type: "string"  
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  
12 pin-num PIC S9(9) COMP-5 SYNC.  
12 pin.  
15 pin2-length  
15 pin2  
12 firstName-num PIC S9(9) COMP-5 SYNC.  
12 firstName.  
15 firstName2-length  
15 firstName2  
12 middleName-num PIC S9(9) COMP-5 SYNC.  
12 middleName.  
15 middleName2-length  
15 middleName  
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.



```
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-RESPONSE.

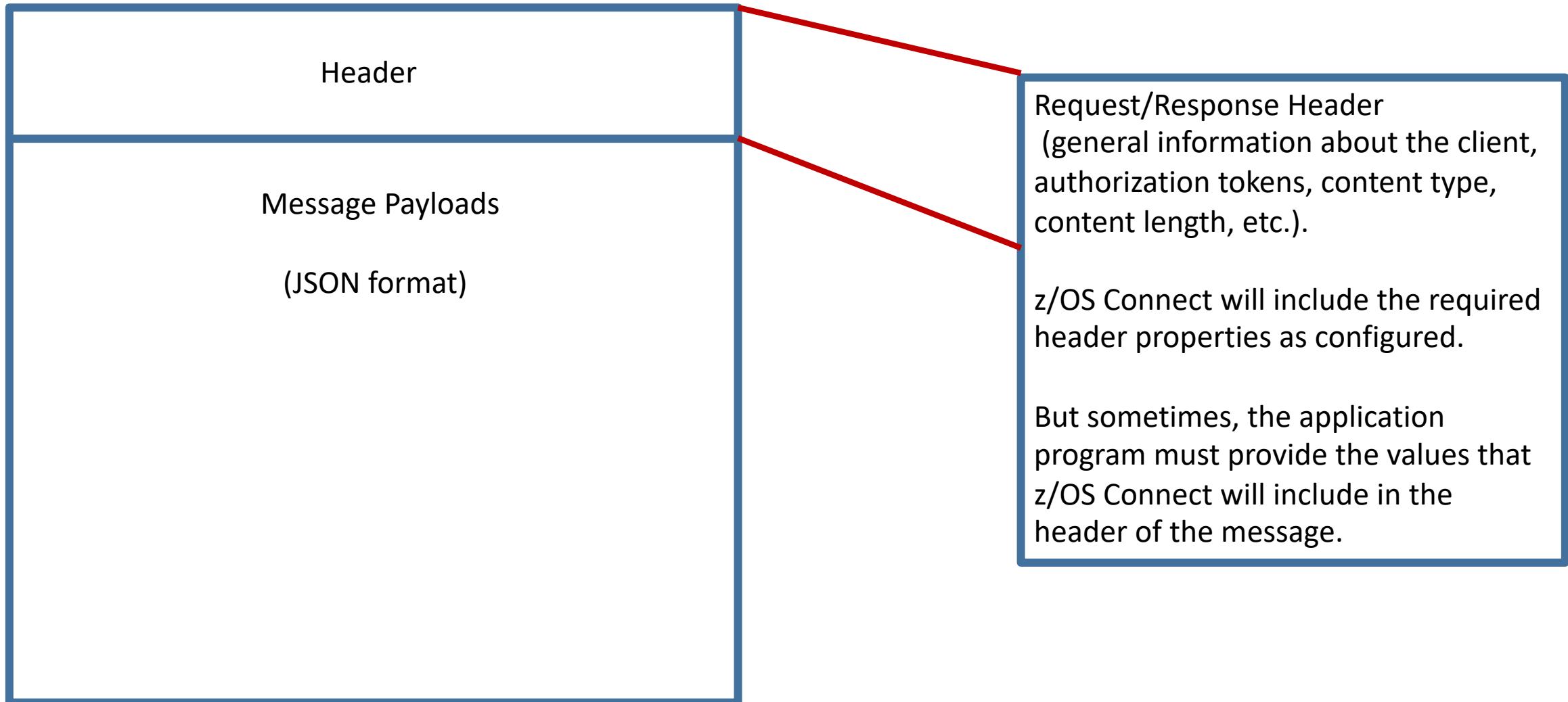
*-----*
* Set up the data for the API Requester call
*-----*
  MOVE 1 to csvIncUpdateServiceOp-num in PUT-REQUEST
  request2-num in PUT-REQUEST
  filea2-num in PUT-REQUEST
  name-num in PUT-REQUEST
  Xaddress-num in PUT-REQUEST
  phoneNumer-num in PUT-REQUEST
  Xdate-num in PUT-REQUEST
  amount-num in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

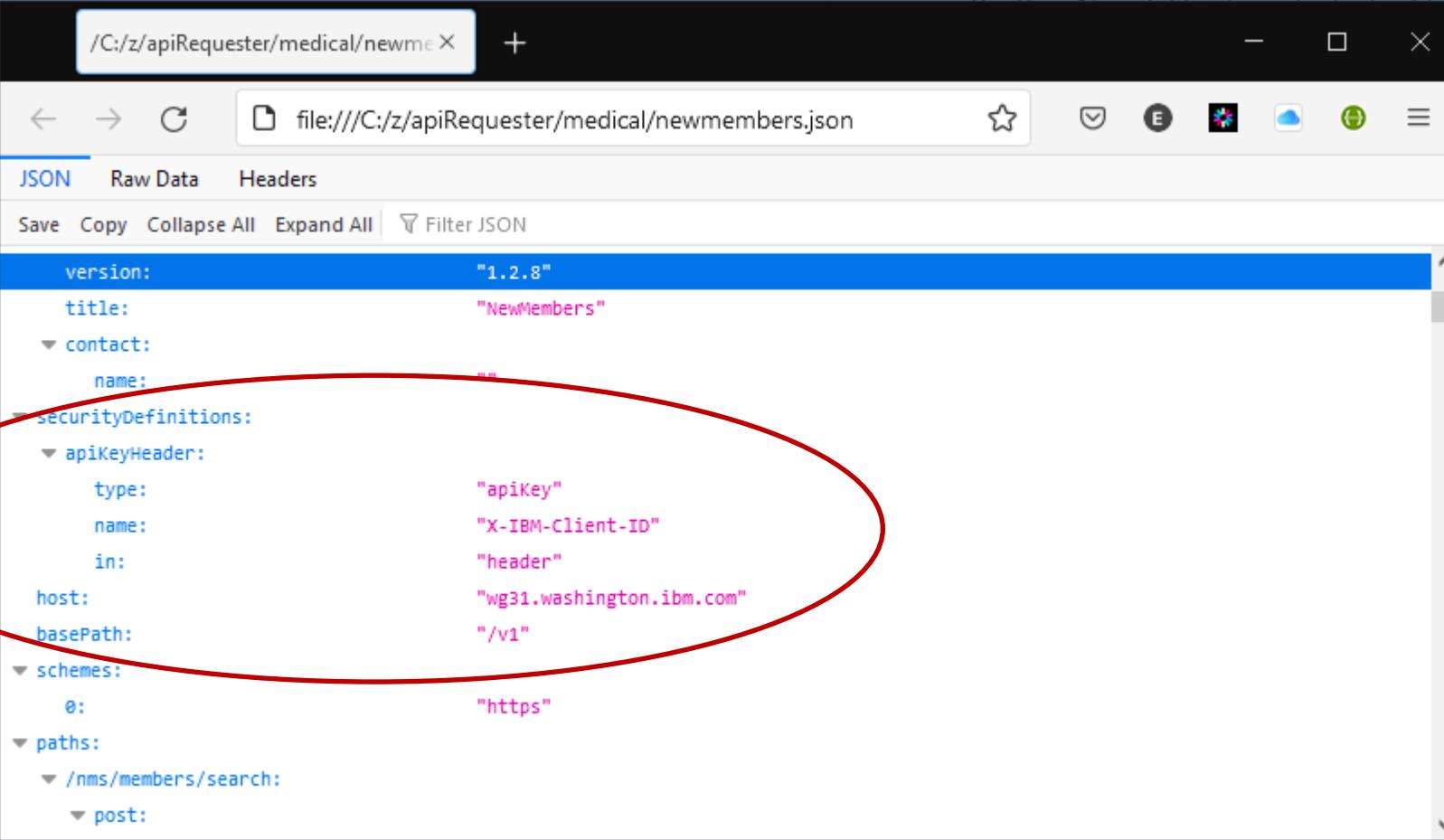
MAIN C                                         04 / 015
Connected to remote server/host wg31z using lu/pool TCP00108 and port 23
```

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



An API may require an API key (aka password)

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



A screenshot of a JSON editor window titled "file:///C:/z/apiRequester/medical/newmembers.json". The editor shows a JSON object with several fields. A red oval highlights the "securityDefinitions" field, which contains an "apiKeyHeader" object. This object specifies the header name "X-IBM-Client-ID" and its value "apiKey". Other fields in the JSON include "version" (1.2.8), "title" (NewMembers), "contact" (name: null), "host" (wg31.washington.ibm.com), "basePath" (/v1), and "schemes" (https). Below this, there's a "paths" object with a "nms/members/search" entry and a "post" method.

```
version: "1.2.8"
title: "NewMembers"
contact:
  name: null
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

* 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-ClientSessionI-length PIC S9999 COMP-5
SYNC.
12 X-HZN-ClientSessionId2 PIC X(255).
09 X-HZN-UserRole-num PIC S9(9) COMP-5 SYNC.
09 X-HZN-UserRole.
12 X-HZN-UserRole2-length PIC S9999 COMP-5
SYNC.
12 X-HZN-UserRole2 PIC X(255).
09 X-HZN-UserAssociationI-num PIC S9(9) COMP-5 SYNC.
09 X-HZN-UserAssociationI-length

File Edit Settings View Communication Actions Window Help

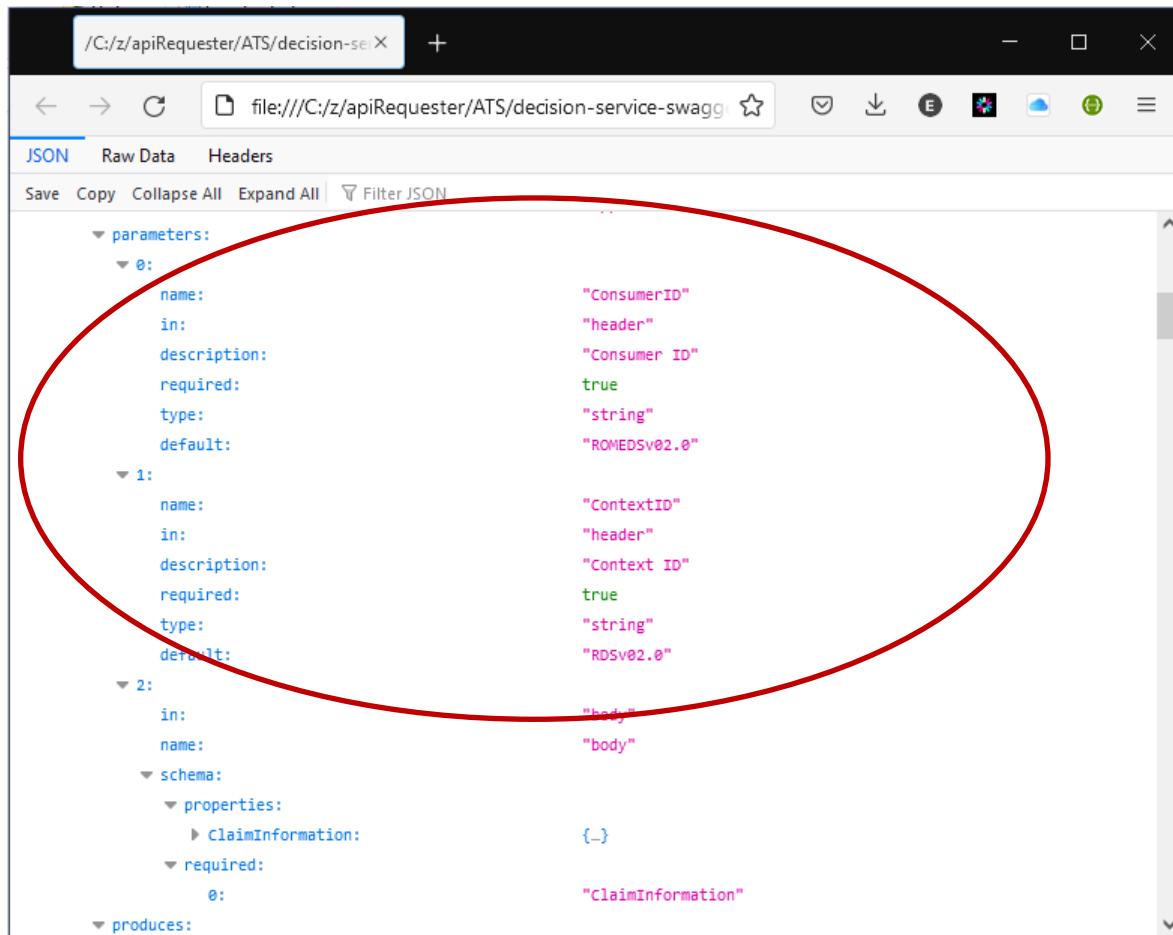
File Edit Edit_Settings Menu Utilities Compilers Test Help

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

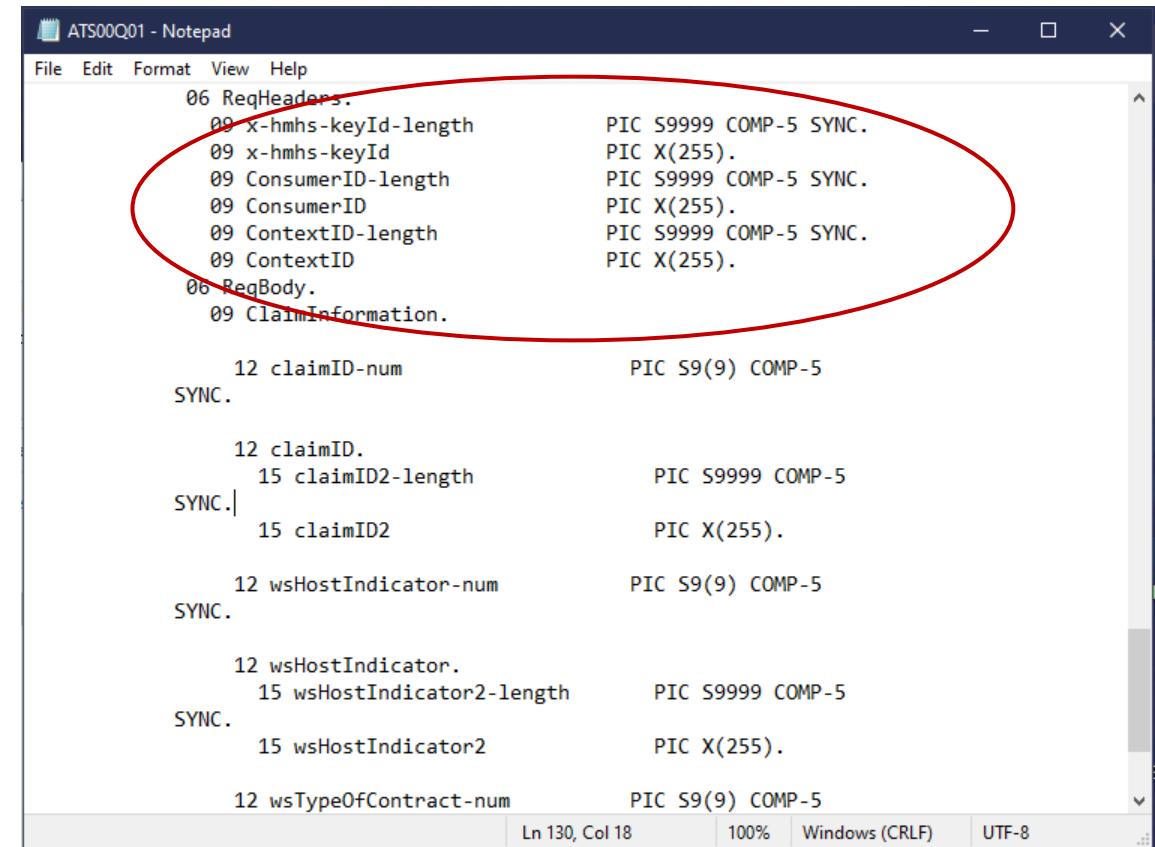


And there may be other required custom header properties

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.



```
parameters:
  0:
    name: "ConsumerID"
    in: "header"
    description: "Consumer ID"
    required: true
    type: "string"
    default: "ROMEDSV02.0"
  1:
    name: "ContextID"
    in: "header"
    description: "Context ID"
    required: true
    type: "string"
    default: "RDSV02.0"
  2:
    in: "body"
    name: "body"
    schema:
      properties:
        ClaimInformation: {}
      required:
        0: "ClaimInformation"
  produces:
```



```
File Edit Format View Help
06 ReqHeaders.
  09 x-hmhs-keyId-length PIC S9999 COMP-5 SYNC.
  09 x-hmhs-keyId PIC X(255).
  09 ConsumerID-length PIC S9999 COMP-5 SYNC.
  09 ConsumerID PIC X(255).
  09 ContextID-length PIC S9999 COMP-5 SYNC.
  09 ContextID PIC X(255).

06 ReqBody.
  09 ClaimInformation.

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

  12 claimID.
    15 claimID2-length PIC S9999 COMP-5
  SYNC.
    15 claimID2 PIC X(255).

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

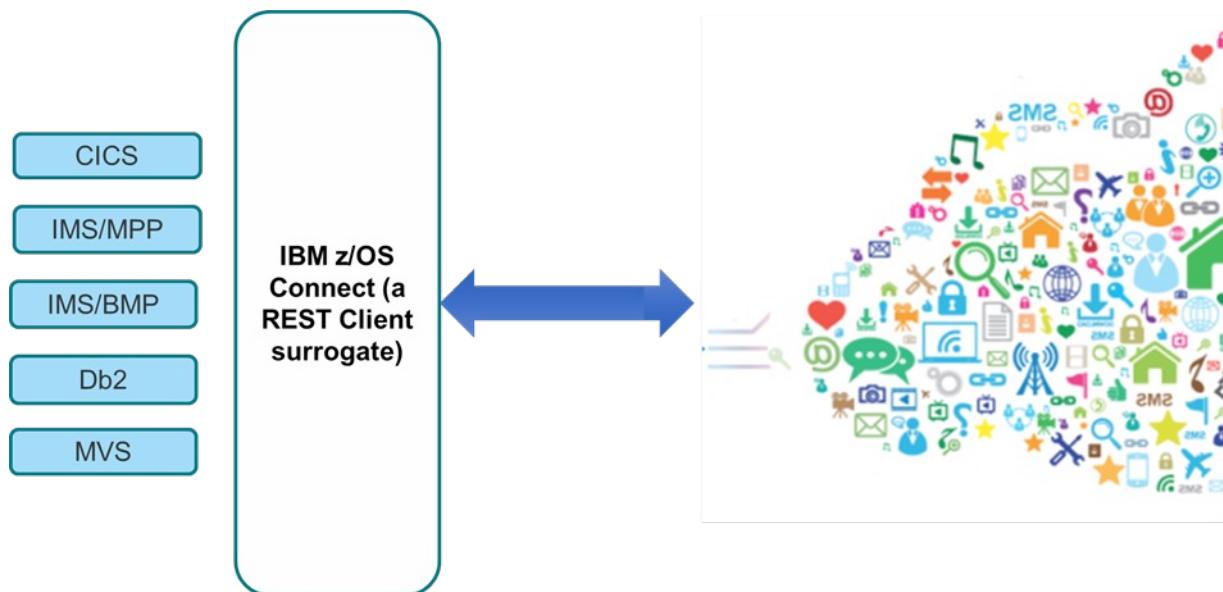
  12 wsHostIndicator.
    15 wsHostIndicator2-length PIC S9999 COMP-5
  SYNC.
    15 wsHostIndicator2 PIC X(255).

  12 wsTypeOfContract-num PIC S9(9) COMP-5
```



Developing API Requesters

For APIs defined using a Swagger 2.0 specification document



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 window 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" and "3.0", with "Installing the build toolkit" being the active section. The main content area contains the title, a "Last Updated: 2023-05-26" timestamp, a brief description of the toolkit's purpose, an "About this task" section, and a "Procedure" section with two numbered steps.

Installing the IBM z/OS Connect build toolkit

Last Updated: 2023-05-26

Install the IBM® z/OS® Connect build toolkit to create service archive (.sar) files or artifacts for an API requester.

About this task

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](#).

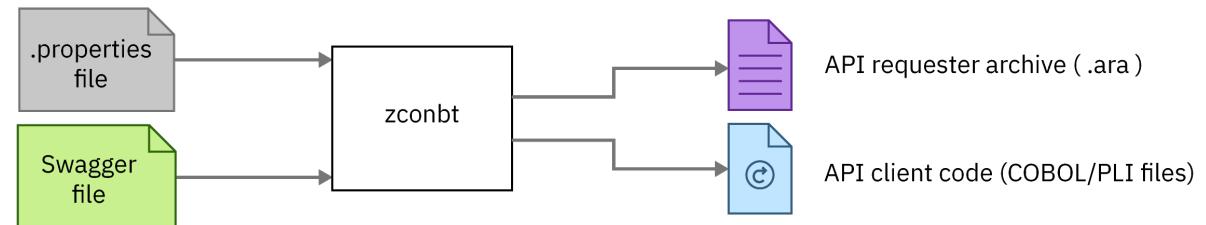
Procedure

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 Swagger 2.0

The screenshot shows a browser window displaying a Swagger 2.0 JSON API definition. The URL is `/C:/apiRequester/cscvinc/cscvinc`. The JSON structure includes fields like `swagger: "2.0"`, `info: { description: "", version: "1.0.0", title: "cscvincapi", basePath: "/cscvincapi" }`, and a detailed `paths` section for the `/employee/{employee}` endpoint, including methods `get`, `post`, `put`, and `delete`.



properties file#

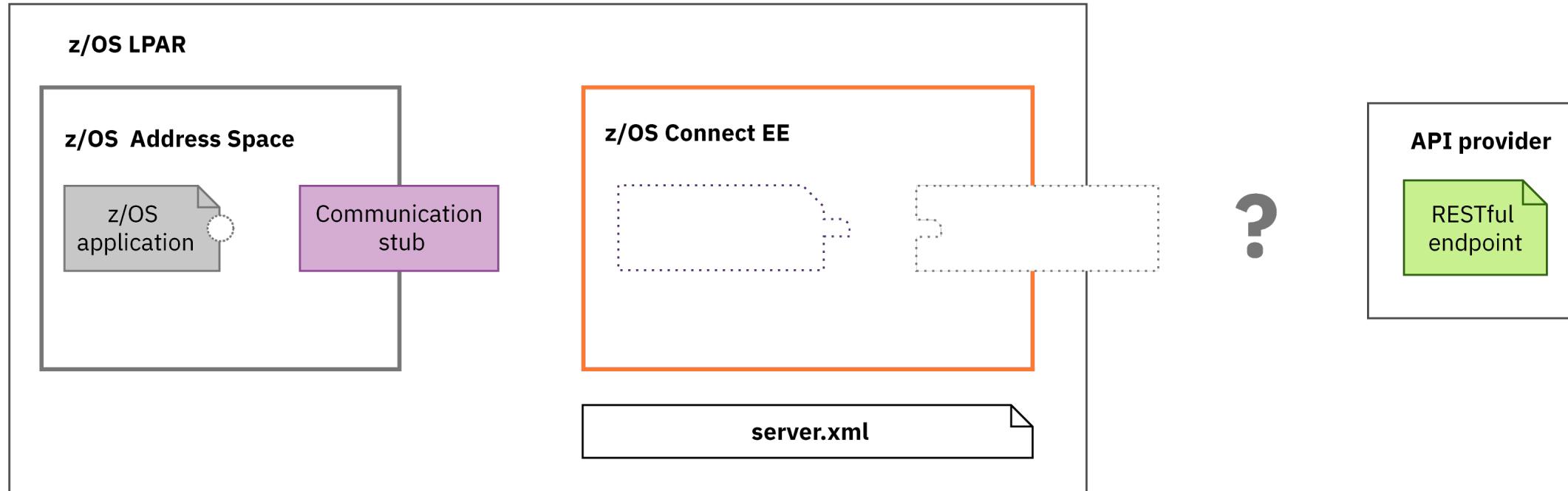
```
apiDescriptionFile=./cscvinc.json
dataStructuresLocation=./syslib
apiInfoFileLocation=./syslib
logFileDirectory=./logs
language=COBOL
connectionRef=cscvincAPI
requesterPrefix=csc
```

#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/zos-connect/3.0?topic=toolkit-build-properties-file>

How to access an API from a COBOL program (Swagger 2.0)



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.  
      BY REFERENCE    BAQ-RETURN-CODE-1  
      BY REFERENCE    BAQ-RETURN-INFO-1  
      BY REFERENCE    BAQ-RETURN-PTR-1  
      BY REFERENCE    BAQ-RETURN-LEN-1
```





Steps to calling a remote API

Include the generated copy books in the COBOL program

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

  * Copy API Requester required copybook
  COPY BAQRINFO.

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

  01 API_RESPONSE.
    COPY CSC02P01.

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

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

API-REQUEST

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

API-RESPONSE

```
CSC00I01  CSC00Q01  CSC00P01 X
  * ++++++
  06 RespBody.
    09 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 X
  03 BAQ-APINAME          PIC X(255)
    VALUE 'cscvincapi_1.0.0'.
  03 BAQ-APINAME-LEN      PIC S9(9) COMP-5 SYNC
    VALUE 16.
  03 BAQ-APIPATH          PIC X(255)
    VALUE '%2Fscvincapi%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 diagram illustrates the steps to calling a remote API. It shows the main program code in the 'GETAPI' window and three copy book definitions in separate windows.

Main Program Code (GETAPI):

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

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

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

* The BAQ-RETURN-CODE field in 'BAQRINFO' indicates whether this
```

Copy Book Definitions:

- CSC00101:** Contains definitions for BAQ-APINAME, BAQ-APINAME-LEN, BAQ-APIPATH, BAQ-APIPATH-LEN, BAQ-APIMETHOD, and BAQ-APIMETHOD-LEN.
- CSC00Q01:** Contains definitions for employee-length, employee, ReqPathParameters, employee-length, employee, and RespBody.
- CSC00P01:** Contains definitions for cscvincSelectServiceOp-num, cscvincSelectServiceOperatio, Container1, RESPONSE-CONTAINER2-num, and SYNC.

Steps to calling a remote API



Access the results that have been placed in working storage

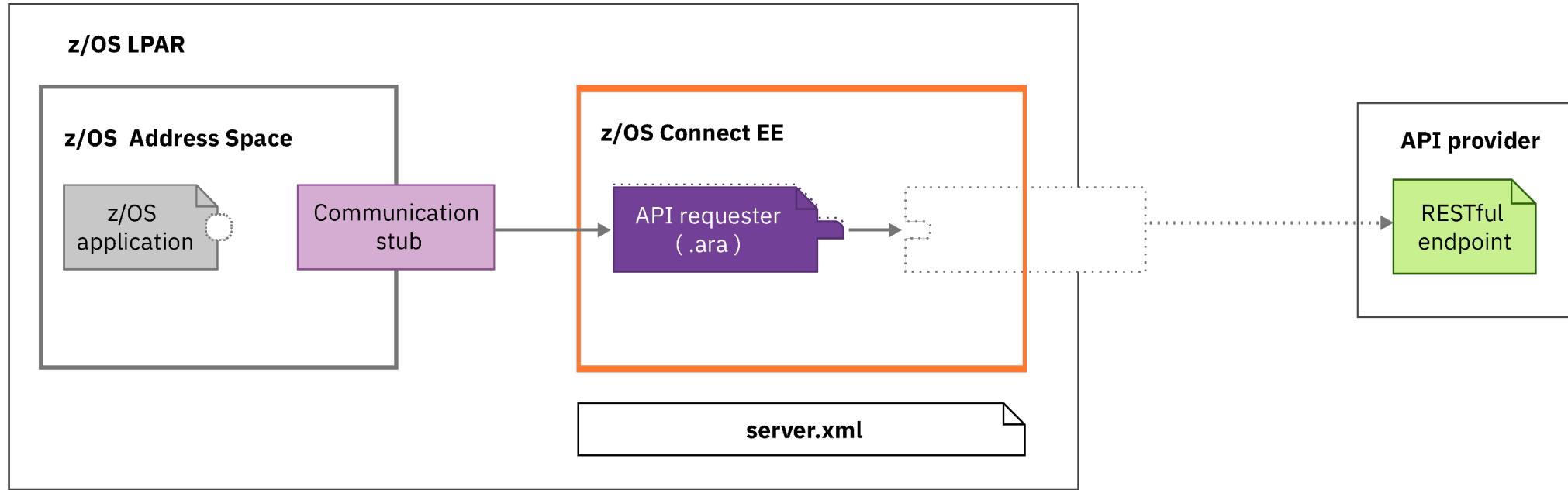
```
BY REFERENCE BAQ-RESPONSE-LEN.  
* The BAQ-RETURN-CODE field in 'BAQRINFO' indicates whether this  
* API call is successful.  
  
* When BAQ-RETURN-CODE is 'BAQ-SUCCESS', response is  
* successfully returned and fields in RESPONSE copybook  
* can be obtained. Display the translation result.  
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) Line 0000000066 Col 001 080  
Command ==> Scroll ==> PAGE  
01 BAQ-RESPONSE-INFO.  
03 BAQ-RESPONSE-INFO-COMP-LEVEL PIC S9(9) COMP-5 SYNC VALUE 0.  
03 BAQ-STUB-NAME PIC X(8)  
03 BAQ-RETURN-CODE PIC S9(9) COMP-5 SYNC.  
    88 BAQ-SUCCESS VALUE 0.  
    88 BAQ-ERROR-IN-API VALUE 1.  
    88 BAQ-ERROR-IN-ZCEE VALUE 2.  
    88 BAQ-ERROR-IN-STUB VALUE 3.  
    88 BAQ-ERROR-NO-RESPONSE VALUE 4.  
03 BAQ-STATUS-CODE PIC S9(9) COMP-5 SYNC.  
03 BAQ-STATUS-MESSAGE PIC X(1024).  
03 BAQ-STATUS-MESSAGE-LEN PIC S9(9) COMP-5 SYNC.  
***** Bottom of Data *****  
18/05/8  
Connected to remote server/host mpz3 using lu/pool MPZ30021 and port 23
```

Note that the return code only addresses the 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: Deploy API requester archive file using Postman or cURL

- 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. The top navigation bar includes File, Edit, View, Help, Home, Workspaces, API Network, Reports, Explore, Search Postman, Invite, and Upgrade. The main workspace shows an overview of a collection named "https://wg31.washington.ibm.com:9483/zosConnect/apiRequesters". A POST request is selected with the URL "https://wg31.washington.ibm.com:9483/zosConnect/apiRequesters". The "Body" tab is active, showing the file "filea.ara" attached. The file details are: 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". The "Send" button is highlighted in blue. Below the request, the response status is shown as 201 Created with a time of 863 ms and a size of 380 B. The results section displays the JSON response.

Command:

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

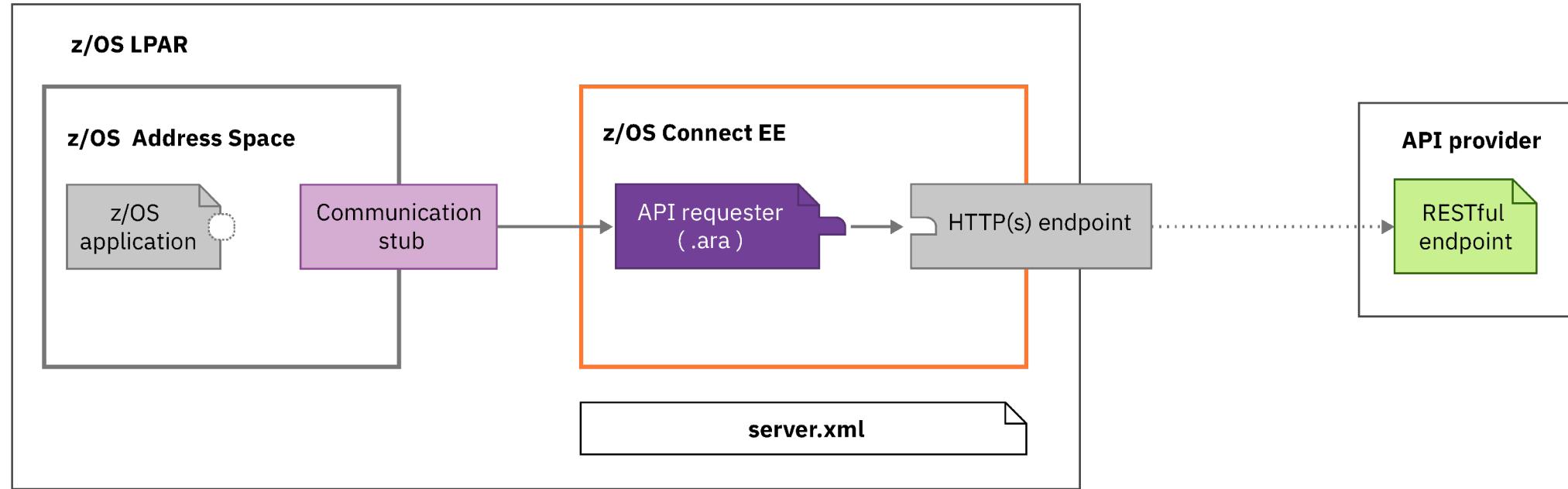
Results:

```
{"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 a remote API

Configure HTTP(S) endpoint configuration element

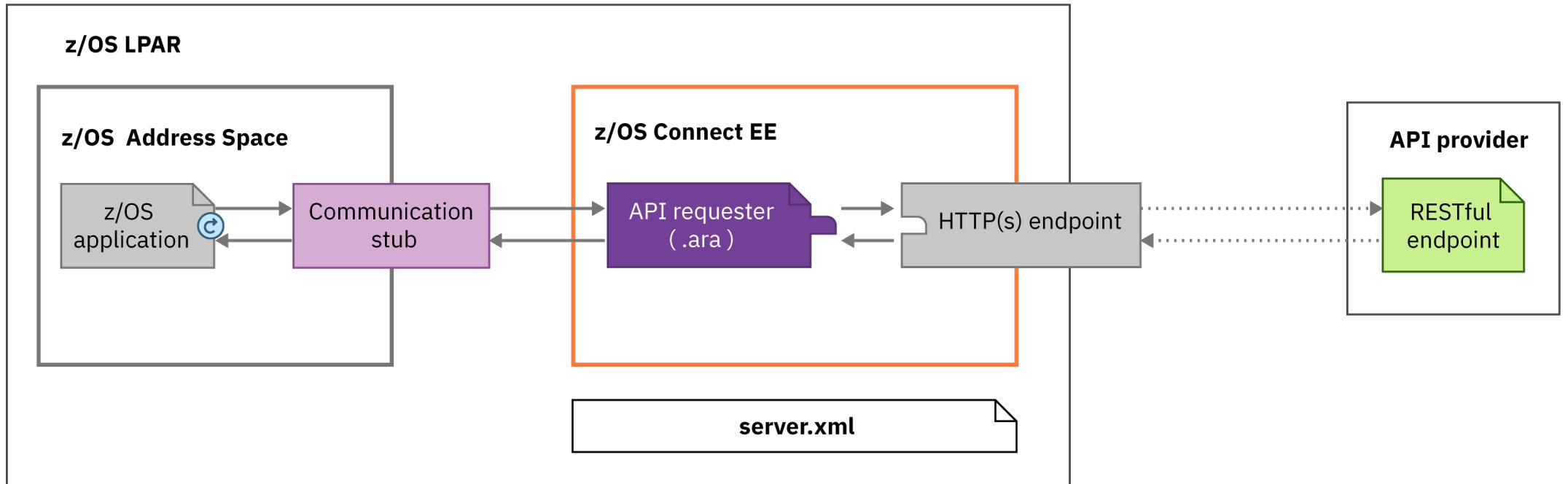


Configure the connection between z/OS Connect EE and the remote API.

ibm.biz/zosconnect-configure-endpoint-connection



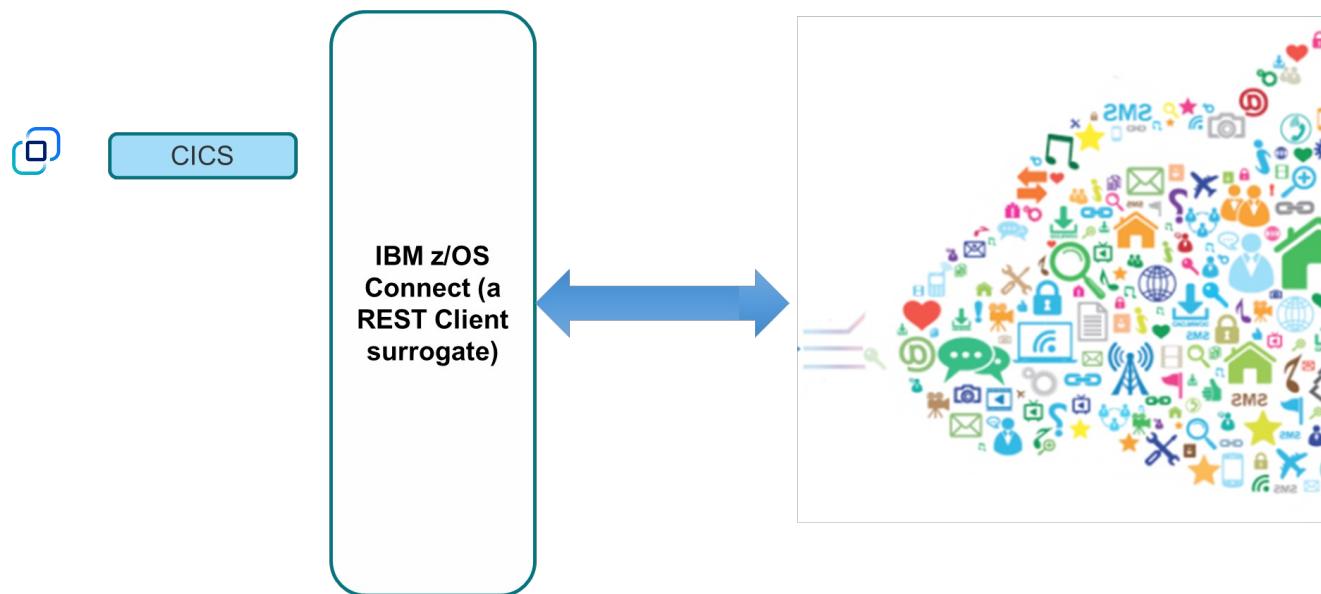
Results - A complete solution for calling a remote Swagger 2.0 API





Developing z/OS Connect API Requesters

For APIs defined using an OpenAPI 3 specification

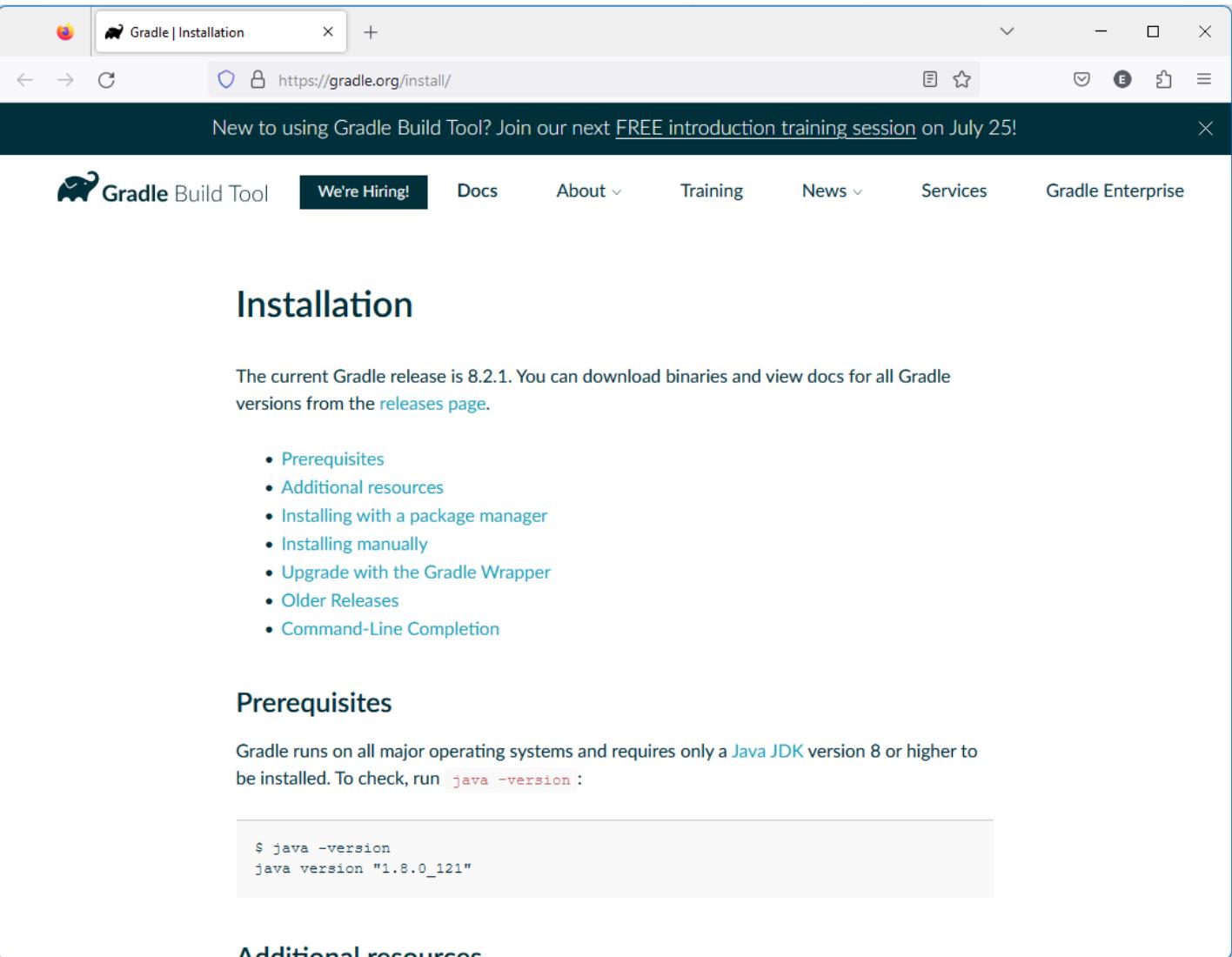


Gradle Build Tool – Installation -



<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 Swagger 2.0
- 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 the "Installation" section of the Gradle website. At the top, there's a navigation bar with links for "We're Hiring!", "Docs", "About", "Training", "News", "Services", and "Gradle Enterprise". A banner at the top of the page says, "New to using Gradle Build Tool? Join our next FREE introduction training session on July 25!". Below the banner, the "Installation" section header is visible. A sub-section titled "Prerequisites" lists several items: "Prerequisites", "Additional resources", "Installing with a package manager", "Installing manually", "Upgrade with the Gradle Wrapper", "Older Releases", and "Command-Line Completion". A code block shows the command `$ java -version` with the output `java version "1.8.0_121"`. At the bottom of the section, there's a link to "Additional resources".

For details on the level Gradle required for the z/OS Connect Gradle plug-in, see URL
<https://www.ibm.com/docs/en/zos-connect/zos-connect/3.0?topic=introduction-system-requirements>



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 window displaying the IBM z/OS Connect documentation. The title bar says "Building z/OS Connect APIs with Gradle". 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 content includes a sidebar with navigation links for "IBM z/OS Connect" and "Building z/OS Connect APIs with Gradle". The main content area has a heading "Building z/OS Connect APIs with Gradle", a last updated date of "2023-05-31", and a section titled "What is Gradle?" with a description of its purpose.

IBM Documentation

Search in IBM z/OS Connect (OpenAPI 3)

IBM z/OS Connect

All products / IBM z/OS Connect / IBM z/OS Connect (OpenAPI 3) / 3.0 /

Was this topic helpful? Yes No

Show full table of contents

Filter on titles

Building z/OS Connect APIs with Gradle

Last Updated: 2023-05-31

z/OS Connect uses Gradle to build API projects into API WAR files.

Contents:

- [What is Gradle?](#)
- [Installing Gradle](#)
- [How Gradle is used in IBM z/OS Connect](#)
- [How z/OS Connect Designer uses Gradle for API provider](#)
- [Running Gradle on z/OS](#)

What is Gradle?

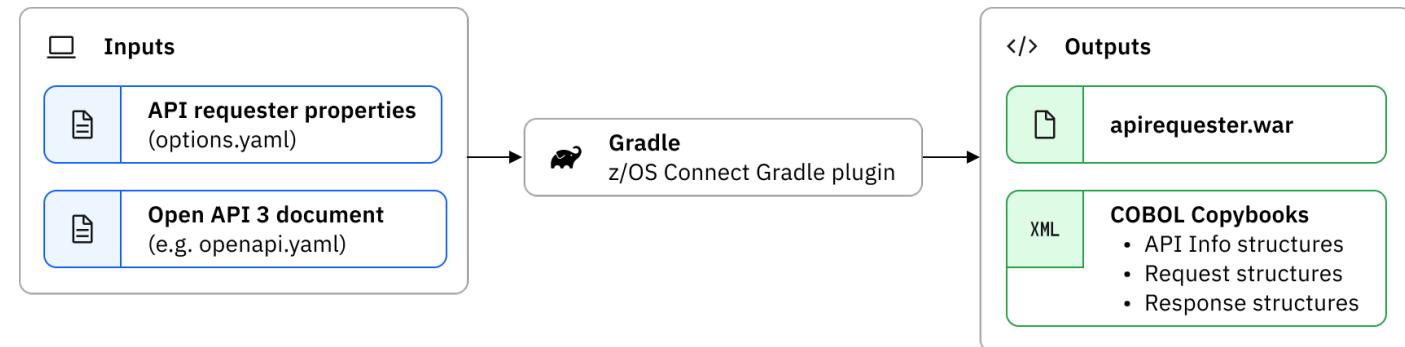
Gradle is an open source build automation tool that is flexible enough to build almost any type of software. For more information about Gradle, see [What is Gradle?](#).

Installing Gradle



Use the Gradle plug-in to generate the artifacts

```
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>



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

API endpoint connection

Only required property

Filter out the desired operations from the OpenAPI 3 document

additionalPropertiesSize
apiName
apiKeyMaxLength
apiKeyParamNameInCookie
apiKeyParamNameInHeader
apiKeyParamNameInQuery
characterMultiplier
characterVarying
characterVaryingLimit
characterWhiteSpace
connectionRef
dataTruncation
dateTimeFormat
defaultCharacterMaxLength
defaultRequestMediaType
defaultResponseMediaType
defaultFractionDigits
generatedCodePage
inlineMaxOccursLimit
language
nameTruncation
operations
requestMediaType
requesterPrefix
responseMediaType
runtimeCodePage
wideComp3

New for OpenAPI 3

At what point to use data areas

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

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



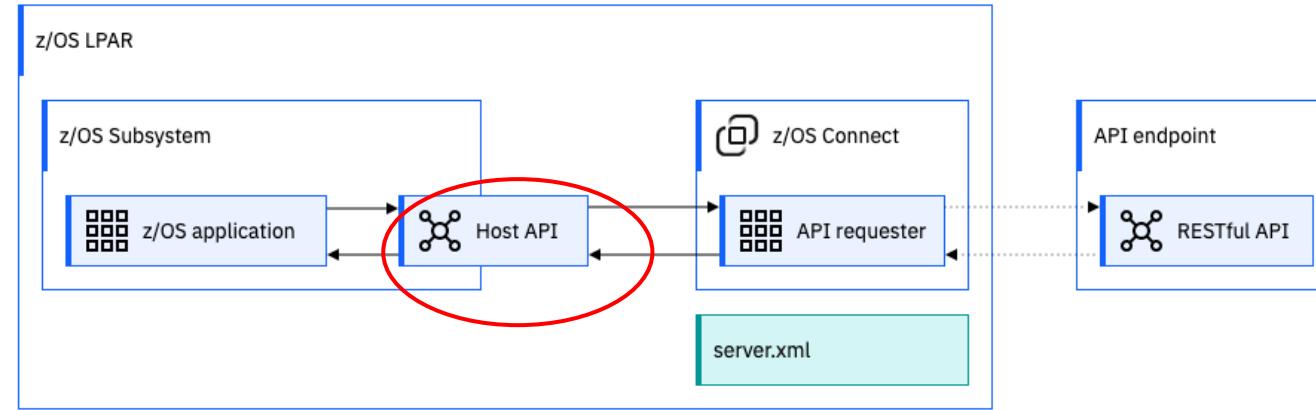
Summary of API requester enhancements for OpenAPI 3

- **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 enhancements 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 Swagger 2.0 communication stub. This change was implemented to address storage issues encountered with the Swagger 2.0 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**.

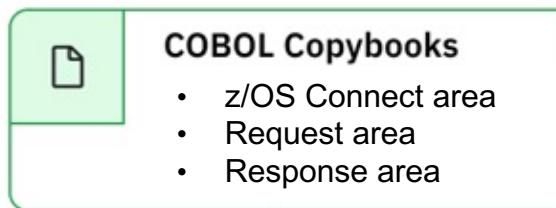
- New function added support for an application to program to be able to handle different response message payloads depending on whether the HTTP status code is 200 (OK), 201 (CREATED), or 404 (NOTFOUND) for example. Each of the response payloads documented in the API's OpenAPI 3.0 definition are mapped 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.



The OpenAPI3 COBOL copybooks

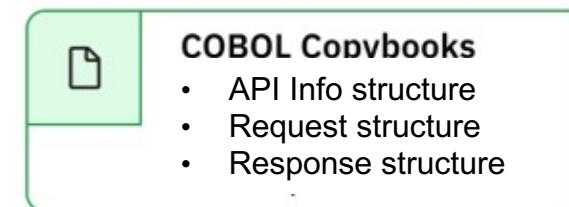
z/OS Connect provides two types of copybooks that are required to interface with the Host API and tooling to generate application specific copybooks

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.

```
File Edit Settings View Communication Actions Window Help  
File Edit Edit_Settings Menu Utilities Compilers Test Help  
EDIT ZCEE30.SBAQC0B(BAQHCONC) - 01.00 Columns 00001 00072  
Command ==> ***** **** 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 * Host API Request parameter names  
000021 77 BAQR-OAUTH-USERNAME PIC X(22)  
000022 VALUE 'BAQHAPI-oAuth-Username'.  
000023 77 BAQR-OAUTH-PASSWORD PIC X(22)  
000024 VALUE 'BAQHAPI-oAuth-Password'.  
000025 77 BAQR-OAUTH-SCOPE PIC X(19)  
000026 VALUE 'BAQHAPI-oAuth-Scope'.  
000027 77 BAQR-OAUTH-CLIENT-ID PIC X(22)  
000028 VALUE 'BAQHAPI-oAuth-ClientId'.  
000029 77 BAQR-OAUTH-CLIENT-SECRET PIC X(26)  
000030 VALUE 'BAQHAPI-oAuth-ClientSecret'.  
000031 * Host API ZCON parameter names  
000043 77 BAQZ-TRACE-VERBOSE PIC X(21)  
000044 VALUE 'BAQHAPI-Trace-Verbose'.  
000045 77 BAQZ-SERVER-URIMAP PIC X(21)  
000046 VALUE 'BAQHAPI-Server-URIMAP'.  
000047 77 BAQZ-SERVER-HOST PIC X(19)  
000048 VALUE 'BAQHAPI-Server-Host'.  
000049 77 BAQZ-SERVER-PORT PIC X(19)  
000050 VALUE 'BAQHAPI-Server-Port'.  
000051 77 BAQZ-SERVER-TIMEOUT PIC X(22)  
000052 VALUE 'BAQHAPI-Server-Timeout'.  
000053 77 BAQZ-SERVER-USERNAME PIC X(23)  
000054 VALUE 'BAQHAPI-Server-Username'.  
000055 77 BAQZ-SERVER-PASSWORD PIC X(23)  
000056 VALUE 'BAQHAPI-Server-Password'.  
***** Bottom_of_Data *****  
File Edit Settings View Communication Actions Window Help  
File Edit Edit_Settings Menu Utilities Compilers Test Help  
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

```
EDIT      ZCEE30.SBAQC0B(BAQHAREC) - 01.00          Columns 00001 00072
Command ==>-
***** **** Top of Data ****
000001   ****
000002   *
000003   * PID 5655-CE5
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 03 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.SBAQCOB(BAQHAREC) - 01.00 Columns 00001 00072
Command ==> - Scroll ==> PAGE
***** **** 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
***** **** Bottom of Data ****

```

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
***** **** 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 ...
***** **** 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

BAO-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

The screenshot shows the WG31 - 3270 application interface. At the top, there are two menu bars: 'File Edit Settings View Communication Actions Window Help' and 'File Edit Edit_Settings Menu Utilities Compilers Test Help'. Below the menu bars are two windows displaying COBOL source code.

The left window displays the following COBOL source code:

```
EDIT      ZCEE30.SBAQCOB(BAQHAREC) - 01.00
Command ==> - 01 BAQ-REQUEST-AREA.
000038      03 BAQ-REQ-AREA-EYE
000039      03 BAQ-REQ-AREA-LENGTH
000040      03 BAQ-REQ-AREA-VERSION
000041      03 BAQ-REQ-RESERVED-01
000042      03 BAQ-REQ-BASE-ADDRESS
000043      03 BAQ-REQ-BASE-LENGTH
000044      03 BAQ-REQ-PARM-INIT
000045      05 BAQ-REQ-RESERVED-02
000046      03 BAQ-REQ-PARAMETERS
000047      05 BAQ-REQ-PARMS
000048          07 BAQ-REQ-PARM-NAME
000049          07 BAQ-REQ-PARM-ADDRESS
000050          07 BAQ-REQ-PARM-LENGTH
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  PIC 9(8) COMP-5 VALUE 0.
000062          03 BAQ-RESP-STATUS-MESSAGE  PIC X(1024).
*****
***** Bottom of Data *****
```

The right window displays the same COBOL source code with some fields highlighted in red and green. Blue arrows point from the left window to the right window, indicating specific fields or sections of interest.

Below the windows, a PDF viewer window is open, showing a document titled '04/015 OF 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!

WG31 - 3270

File Edit Settings View Communication Actions Window Help

File Edit Edit_Settings Menu Utilities Compilers Test Help

EDIT JOHNSON.RBK02I01.CPY Columns 00001 00072
Command ==> _ Scroll ==> PAGE

```
***** **** Top of Data ****
000001 * ++++++ This file contains the generated API information structure
000002 * which is passed to the Host API via the BAQEXEC call.
000003 * ++++++
000004 * ++++++
000005 01 BAQ-API-INFO-RBK02I01.
000006 03 BAQ-API-INFO-EYE          PIC X(4)
000007      VALUE 'BAQA'.
000008 03 BAQ-API-INFO-LENGTH       PIC 9(9) COMP-5 SYNC
000009      VALUE 1052.
000010 03 BAQ-API-INFO-VERSION      PIC 9(9) COMP-5 SYNC
000011      VALUE 1.
000012 03 BAQ-API-INFO-RESERVED01   PIC 9(9) COMP-5 SYNC
000013      VALUE 0.
000014 03 BAQ-API-NAME            PIC X(255)
000015      VALUE 'RedbookApi'.
000016 03 BAQ-API-NAME-LEN         PIC 9(9) COMP-5 SYNC
000017      VALUE 10.
000018 03 BAQ-API-PATH            PIC X(255)
000019      VALUE '%2Fredbooks'.
000020 03 BAQ-API-PATH-LEN         PIC 9(9) COMP-5 SYNC
000021      VALUE 11.
000022 03 BAQ-API-METHOD          PIC X(255)
000023      VALUE 'GET'.
000024 03 BAQ-API-METHOD-LEN       PIC 9(9) COMP-5 SYNC
000025      VALUE 3.
000026 03 BAQ-API-OPERATION        PIC X(255)
000027      VALUE 'getHtlRedbooks'.
000028 03 BAQ-API-OPERATION-LEN     PIC 9(9) COMP-5 SYNC
000029      VALUE 14.
000030

***** Bottom of Data ****
```

MA A 04/015

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

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 Columns 00001 00072
Command ==> Scroll ==> PAGE
000001 * ++++++*+++++*+++++*+++++*+++++*+++++*+++++*+
000002 * This file contains the generated language structure(s) for
000003 * request JSON schema 'getAllRedbooks_request.json'.
000004 * This structure was generated using 'DFHJS2LS' at mapping level
000005 * '5.0'.
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.
```

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



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

```
WG31 - 3270
File Edit Settings View Communication Actions Window Help
File Edit Edit_Settings Menu Utilities Compilers Test Help
EDIT          JOHNSON.RBK02P01.CPY
Command ===> -
000343
000344      01 BAQBASE-RBK02P01 .
000345
000346      03 responseCode200-num
000347      03 responseCode200-dataarea
000348
000349
000350      03 responseCode404-existence
000351      03 responseCode404-dataarea
000352
000353
000354      01 RBK02P01-responseCode200 .
000355      03 responseCode200
000356          06 Xtitle-length
000357          06 Xtitle
000358
000359          06 authors-num
000360          06 authors-dataarea
000361
000362          06 Xstatus-length
000363          06 Xstatus
000364          06 formNumber
000365
000366          06 publicationDate-existence
000367
000368          06 publicationDate
000369          09 publicationDate2-length
000370          09 publicationDate2
000371
000372          06 documentType-existence
000373
000374          06 documentType
000375          09 documentType2-length
000376          09 documentType2
000377
000378          06 sizeMB-existence
000379
000380          06 sizeMB
000381

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



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

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

BAQEXEC

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

BAQGETN

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

BAQFREE

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

BAQTERM

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



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.
```

```
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
```

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

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          PIC S9999 COMP-5 SYNC.  
    06 Xtitle                PIC X(80).  
  
    06 authors-num           PIC S9(9) COMP-5 SYNC.  
    06 authors-dataarea      PIC X(16).  
  
    06 Xstatus-length        PIC S9999 COMP-5 SYNC.  
    06 Xstatus               PIC X(9).  
    06 formNumber            PIC X(12).
```

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 and required security details.

The screenshot shows a JSON editor displaying a Swagger API definition. The 'host' field is set to 'localhost:8080'. Below it, a COBOL dump window titled 'CSC02101' shows a record structure with fields 'BAQ-APINAME' and 'BAQ-APIPATH'. A red arrow points from the 'host' field in the JSON editor to the 'BAQ-APIPATH' field in the COBOL dump, indicating they are connected.

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

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.	

The screenshot shows a 'Server Config' interface with an XML configuration file named 'apiRequesterHTTPS.xml'. The 'zosconnect_endpointConnection' element has its 'host' attribute set to 'https://dvipa.washington.ibm.com'. Above the XML, a properties file 'cscvinc.properties' contains the line 'connectionRef=cscvincAPI'. A red arrow points from the 'connectionRef' line to the 'zosconnect_endpointConnection' element, and another red arrow points from the 'host' attribute to the resulting URL at the bottom.

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" />
```

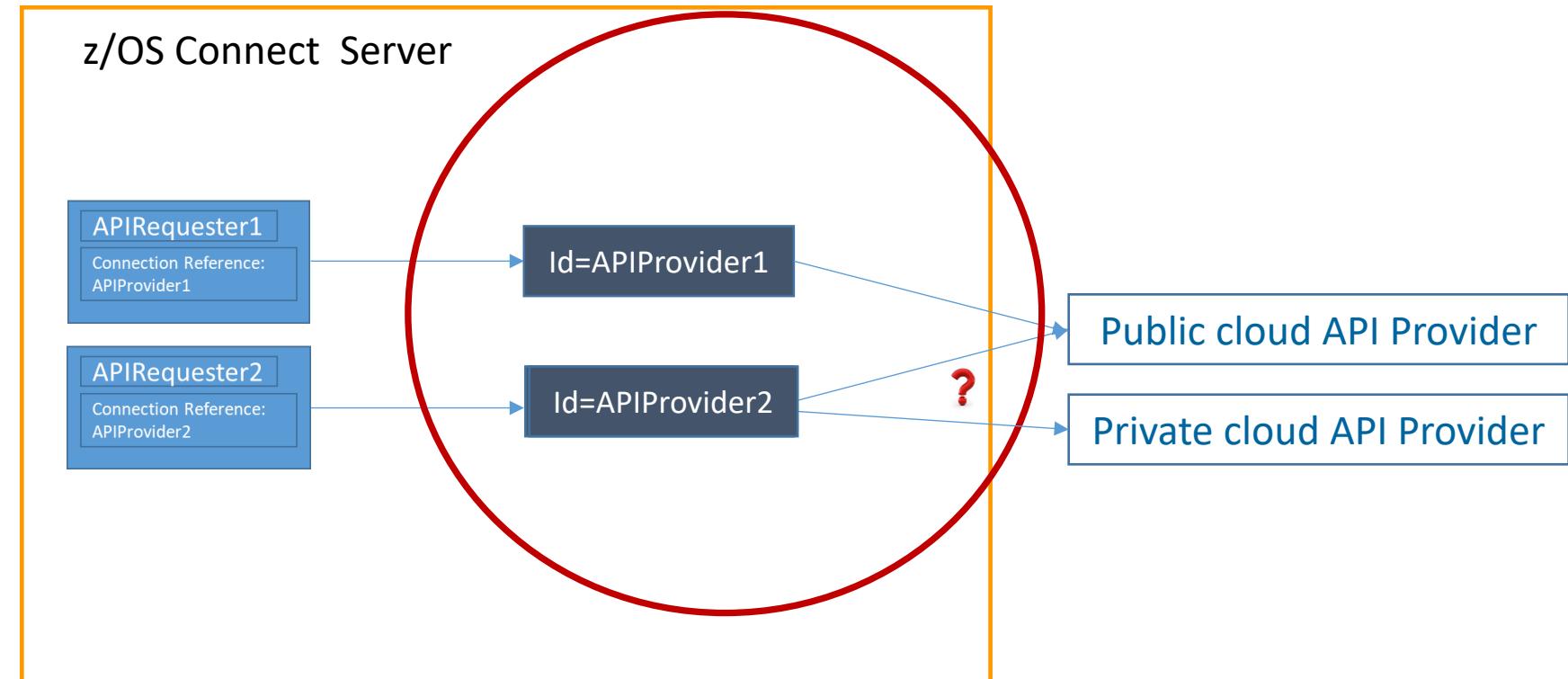
```
<zosconnect_authData id="mySAFAuth"
  user="USER1"
  password="user1" />
</server>
```

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



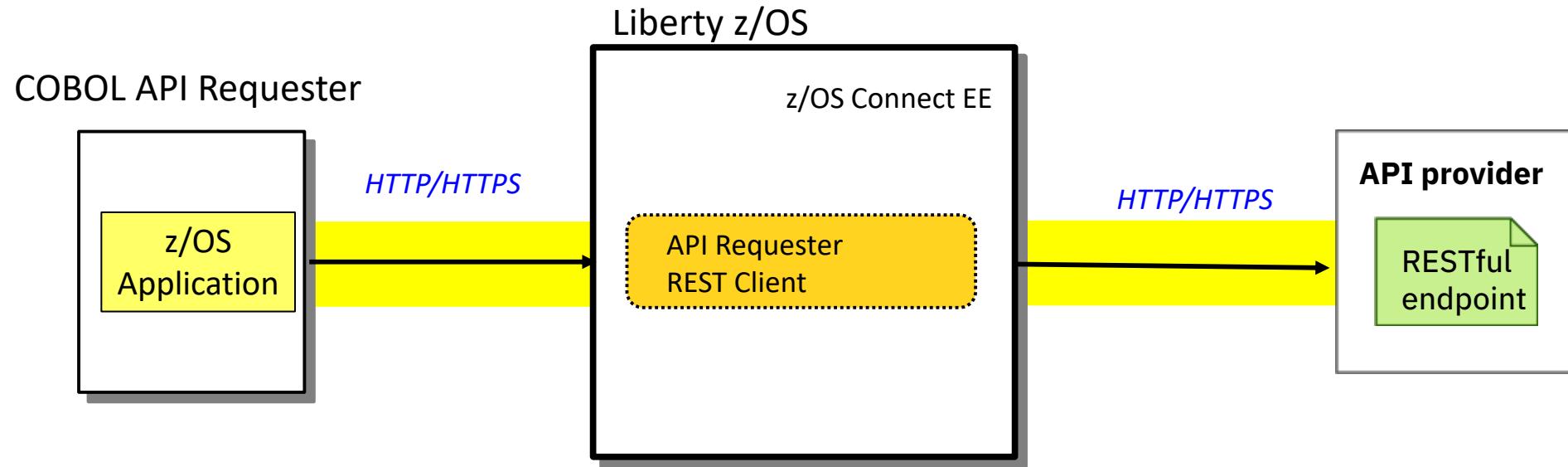
```
<zosconnect_apiRequesters>
  requireAuth="true|false"
  <apiRequester name="cscvincapi_1.0.0"
    connectionRef="APIProvider2"
  </zosconnect_apiRequesters>
```

```
<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*

The screenshot shows a CICS terminal window titled "WG31 - 3270". The terminal displays a COBOL-like code snippet for a "URIMAP" resource. Several lines of code are circled in red: "Urimap(BAQURIMP)", "Usage(Client)", "Scheme(Http)", and "Host(wg31.washington.ibm.com:9120)". The code defines various parameters for the URI map, including host, port, and scheme. The bottom of the screen shows standard CICS control codes (PF 1-12, VAR, SBH, SFH, SB, SF) and system information (SYSID=CICS, APPLID=CICS53Z, TIME: 10.38.37, DATE: 02/14/22). A status bar at the bottom indicates a connection to "wg31a" using TCP port 23.

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.

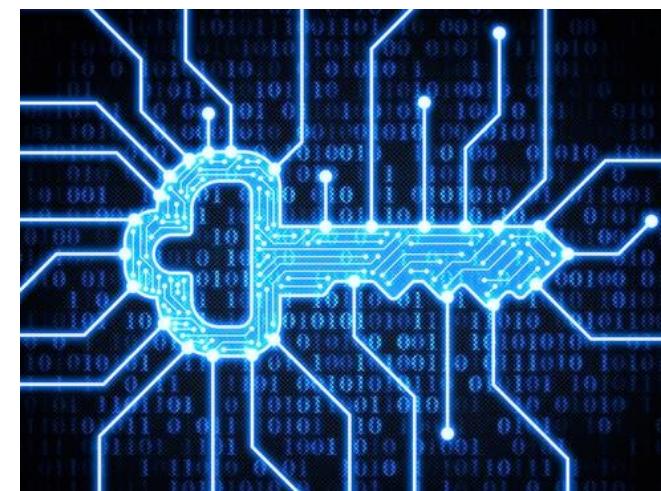
BAQUSERNAME - Specifies the username for connections if basic authentication is used.

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

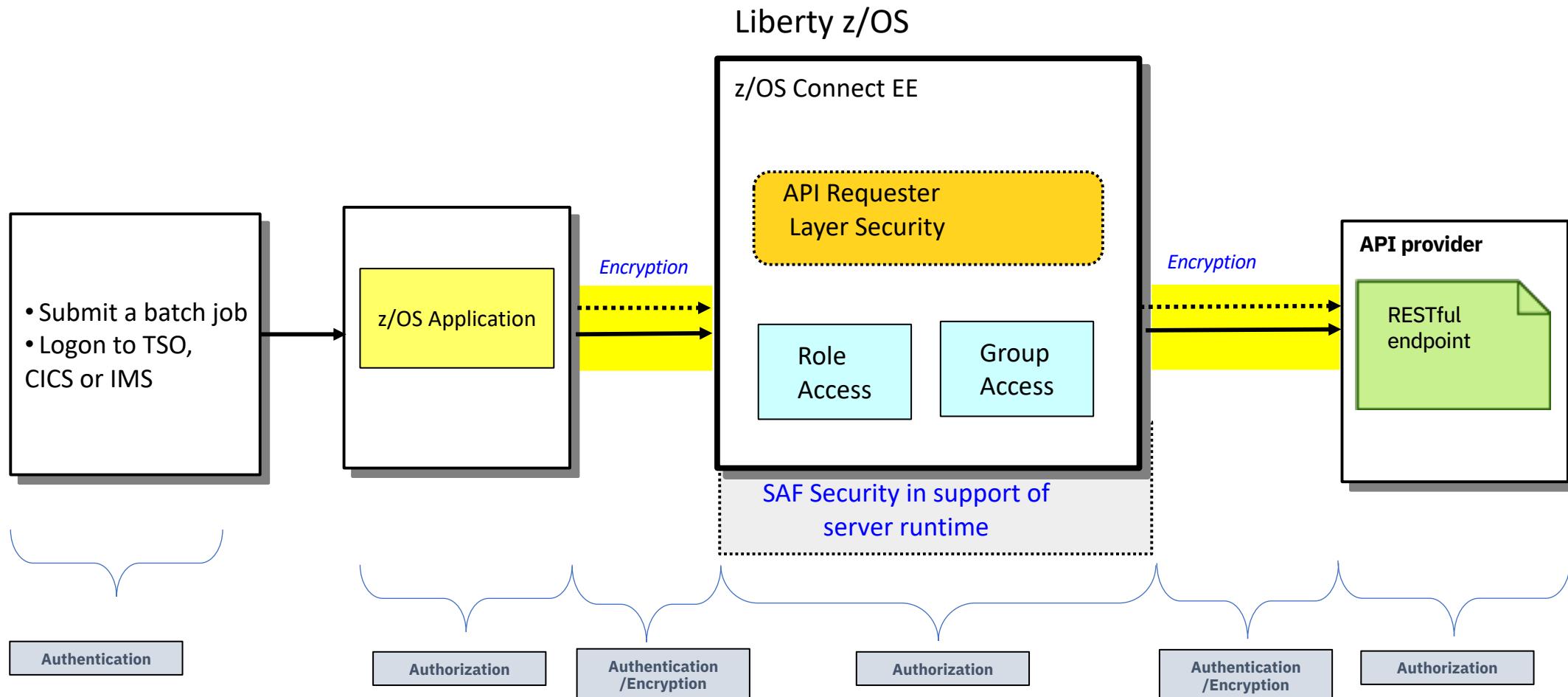
Security involves

- 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.



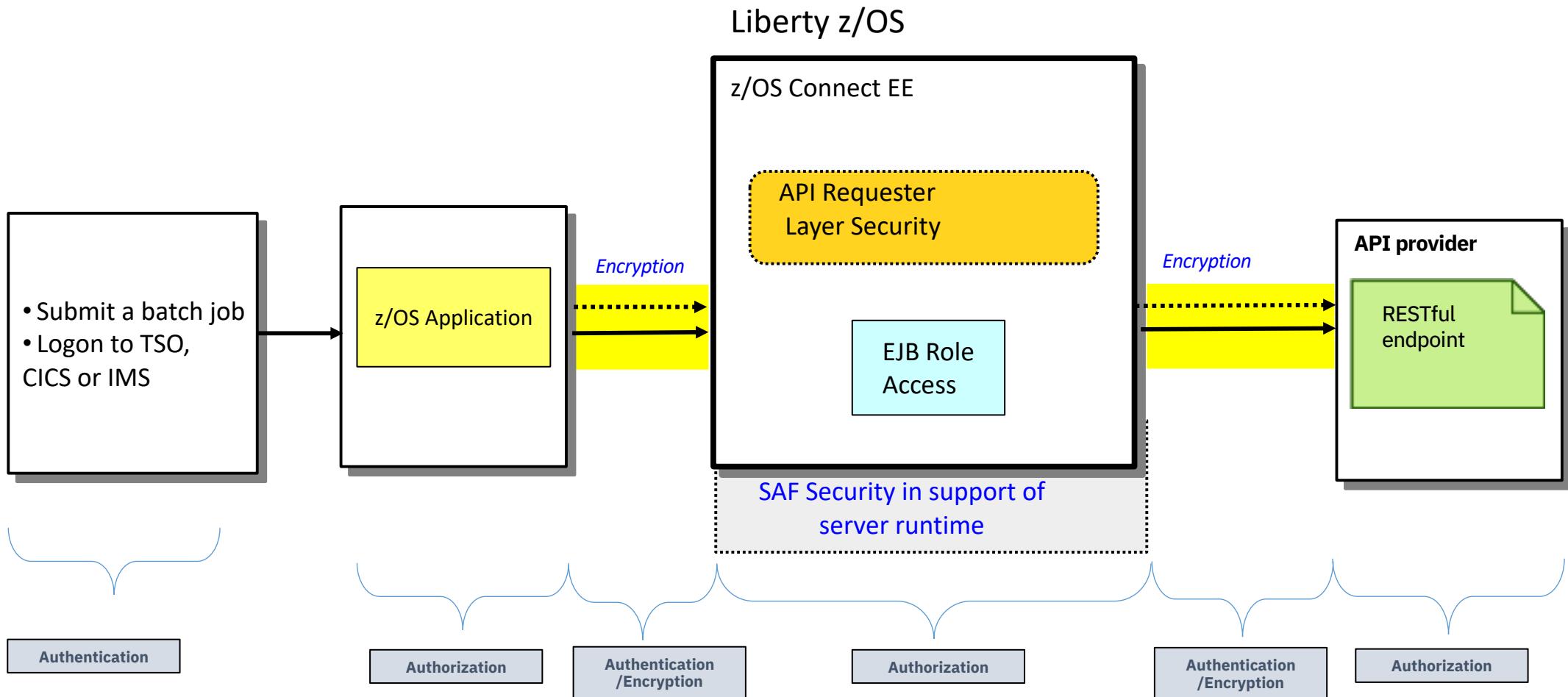


Outbound Authentication versus Authorization (Swagger 2.0)



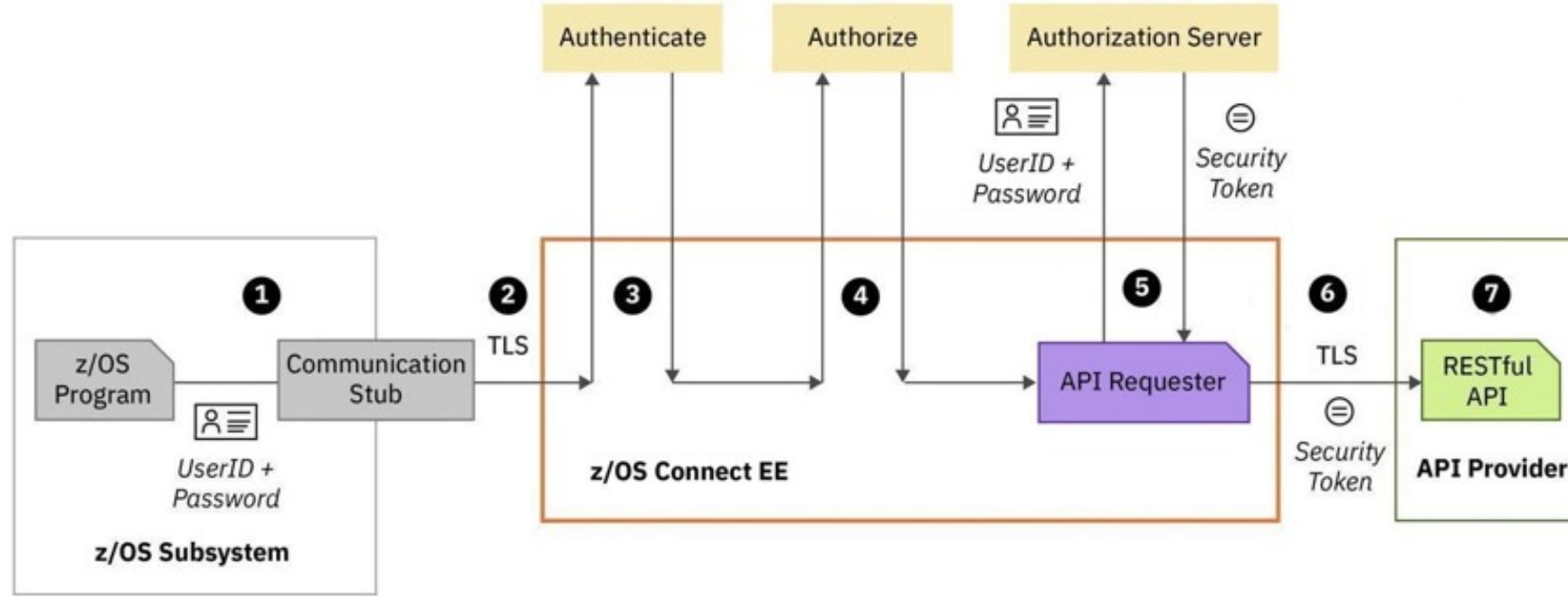


Outbound Authentication versus Authorization (OpenAPI 3)





Typical z/OS Connect EE API Requester security flow

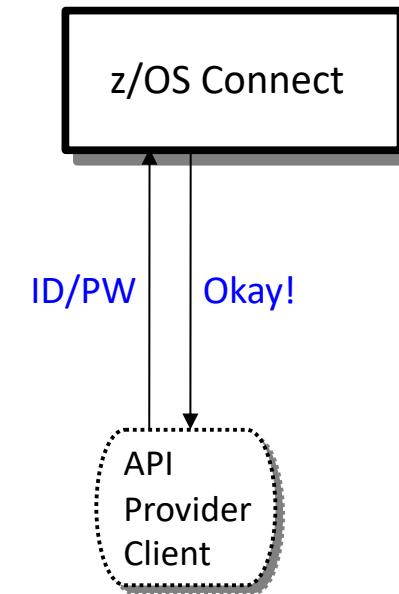


1. A user ID and password can be used for basic authentication by the z/OS Connect EE server
2. Connection between the CICS, IMS, or z/OS application and the z/OS Connect EE server can use TLS
3. Authenticate the CICS, IMS, or z/OS application.
4. Authorize the authenticated user ID to connect to z/OS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

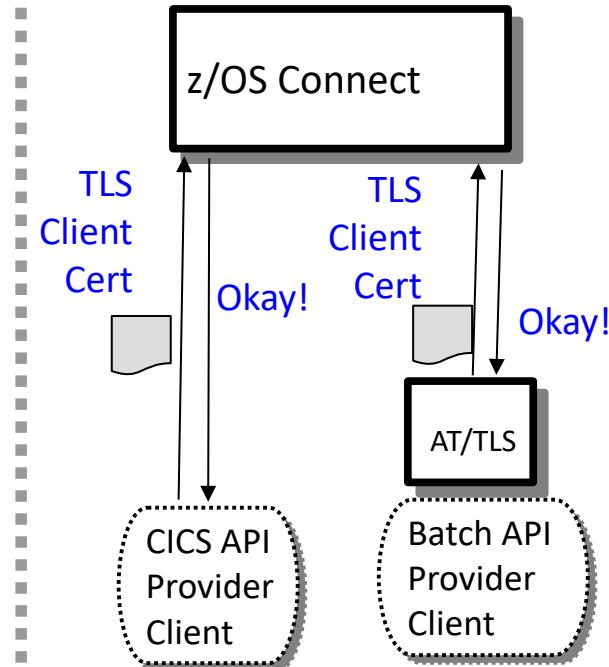
Two options for providing credentials for authentication

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  
CEEXOPT POSIX=((ON),OVR),  
      ENVAR=(('BAQURI=wg31.washington.ibm.com',  
      'BAQPORT=9120',  
      'BAQUSERNAME=USER1',  
      'BAQPASSWORD=USER1'),OVR),  
      RPTOPTS=((ON),OVR)  
END
```

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



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

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

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



Tech/Tip: Generating PassTickets on z/OS

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

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

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

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

*-----*
MAINLINE SECTION.

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

JOHNSON.PASSTCKT.SOURCE(ATSPKTTC)

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

Tech/Tip: API Requester - HTTP v HTTPS



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

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

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

CICS URIMAPS

```
WG31
File Edit Settings View Communication Actions Window Help
OVERTYPE TO MODIFY
CEDA ALter UriMap( BAQRIMP )
UriMap      : BAQRIMP
Group       : SYSGRP
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
PF 1 HELP 2 COM 3 END      6 CRSR 7 SBH 8 SFH 9 MSG 10 SB 11
MA  C
Connected to remote server/host wg31 using lu/pool TCP00133 and port 23

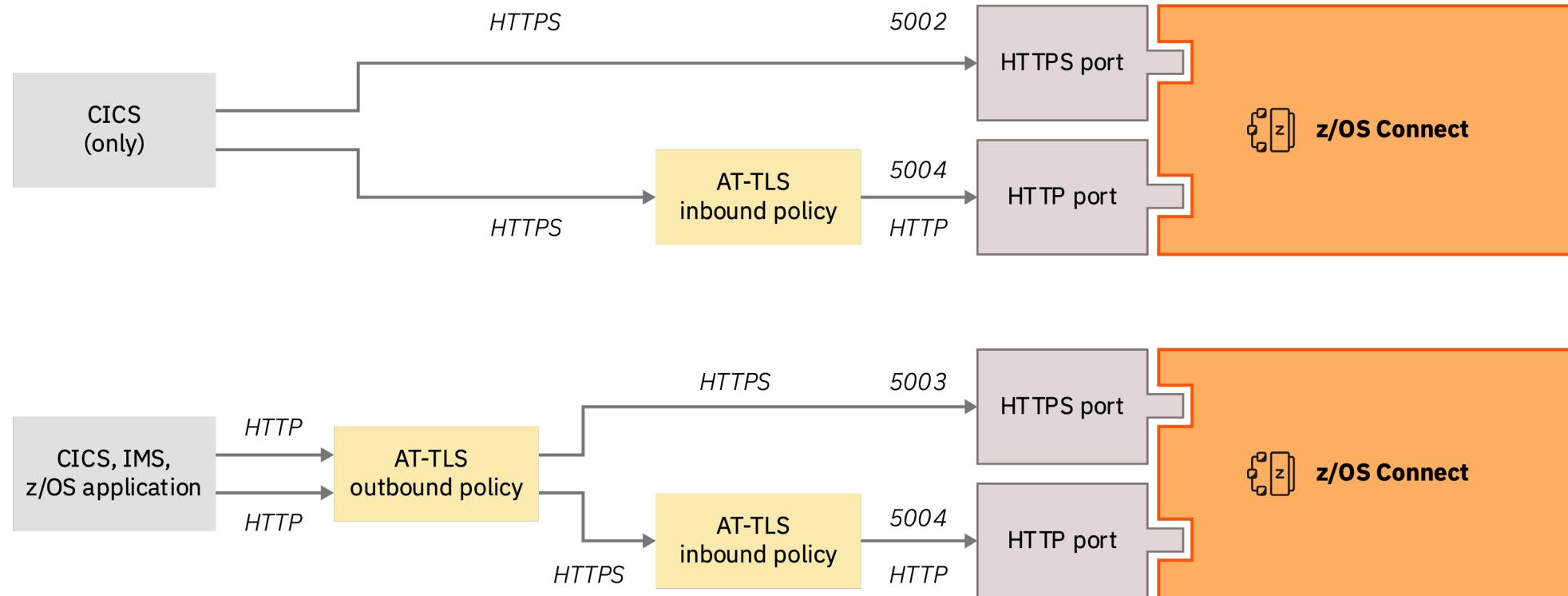
CICS RELEASE
File Edit Settings View Communication Actions Window Help
OVERTYPE TO MODIFY
CICS RELEASE = 0710
CEDA ALter UriMap( BAQRIMP )
UriMap      : BAQRIMP
Group       : SYSGRP
DEscription ==> URIMAP for z/OS Connect EE server
Status      ==> Enabled      Enabled | Disabled
Usage       ==> Client       Server | Client | Pipeline | Atom
                         | Jvmserver
UNIVERSAL RESOURCE IDENTIFIER
SCHEME      ==> HTTPS       HTTP | HTTPS
PORT        ==> 09443       No | 1-65535
HOST        ==> wg31.washington.ibm.com
             ==>
PATH         ==> /
(Mixed Case) ==>
             ==>
             ==>
             ==>
+ OUTBOUND CONNECTION POOLING
SYSID=CICS APPLID=CICS53Z
PF 1 HELP 2 COM 3 END      6 CRSR 7 SBH 8 SFH 9 MSG 10 SB 11 SF 12 CNCL
MA  C
Connected to remote server/host wg31 using lu/pool TCP00133 and port 23
13/022
```

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

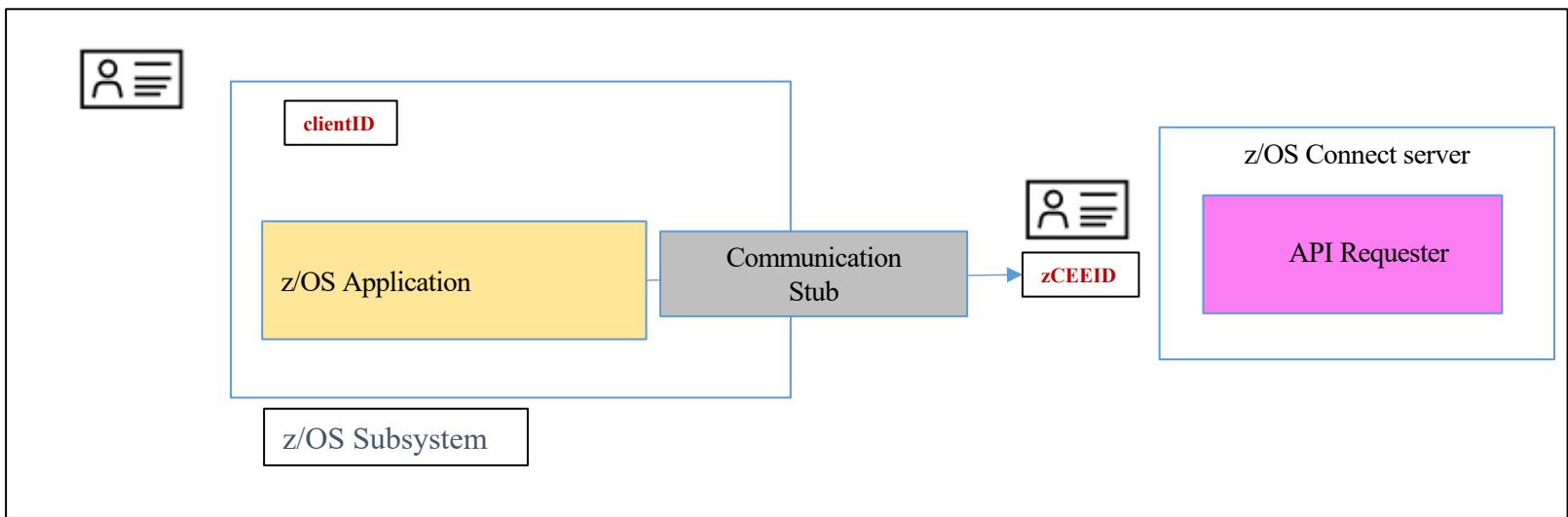
MOVE "URIMAP01" TO BAQ-ZCON-SERVER-URI.



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



API Requester - basic authentication and identity assertion (Swagger 2.0 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>
```

* Added in V3.0.45



API Requester - API Provider Authentication

Several different ways this can be accomplished:

Basic Authentication



ID/PW

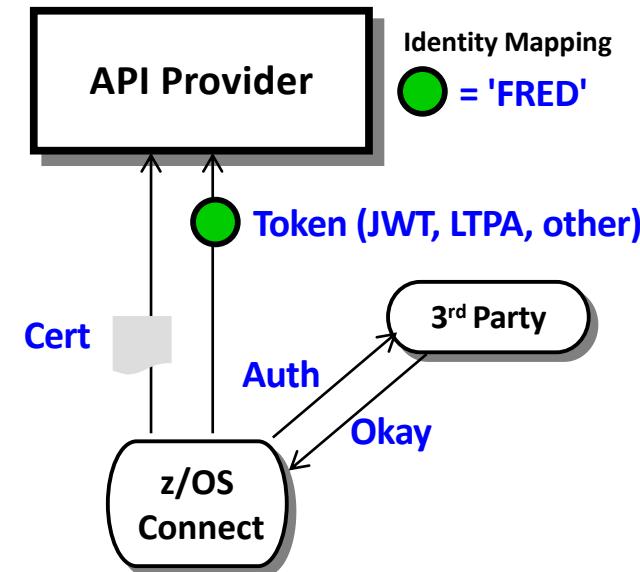
Client Certificate



TLS
Client
Cert

Could be
a trusted
server

Third Party Authentication



Cert

3rd Party

Okay

Identity Mapping
Green Circle = 'FRED'

zCEE supplies ID/PW or
ID/Passticket

Server prompts for certificate
zCEE supplies certificate

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



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

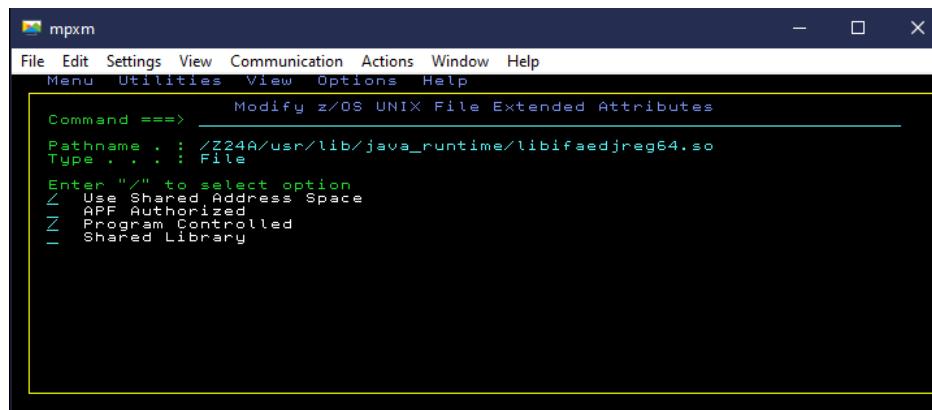
RDEFINE SURROGAT *.BAQASSRT UACC(NONE) OWNER(SYS1)

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
```





```
<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.

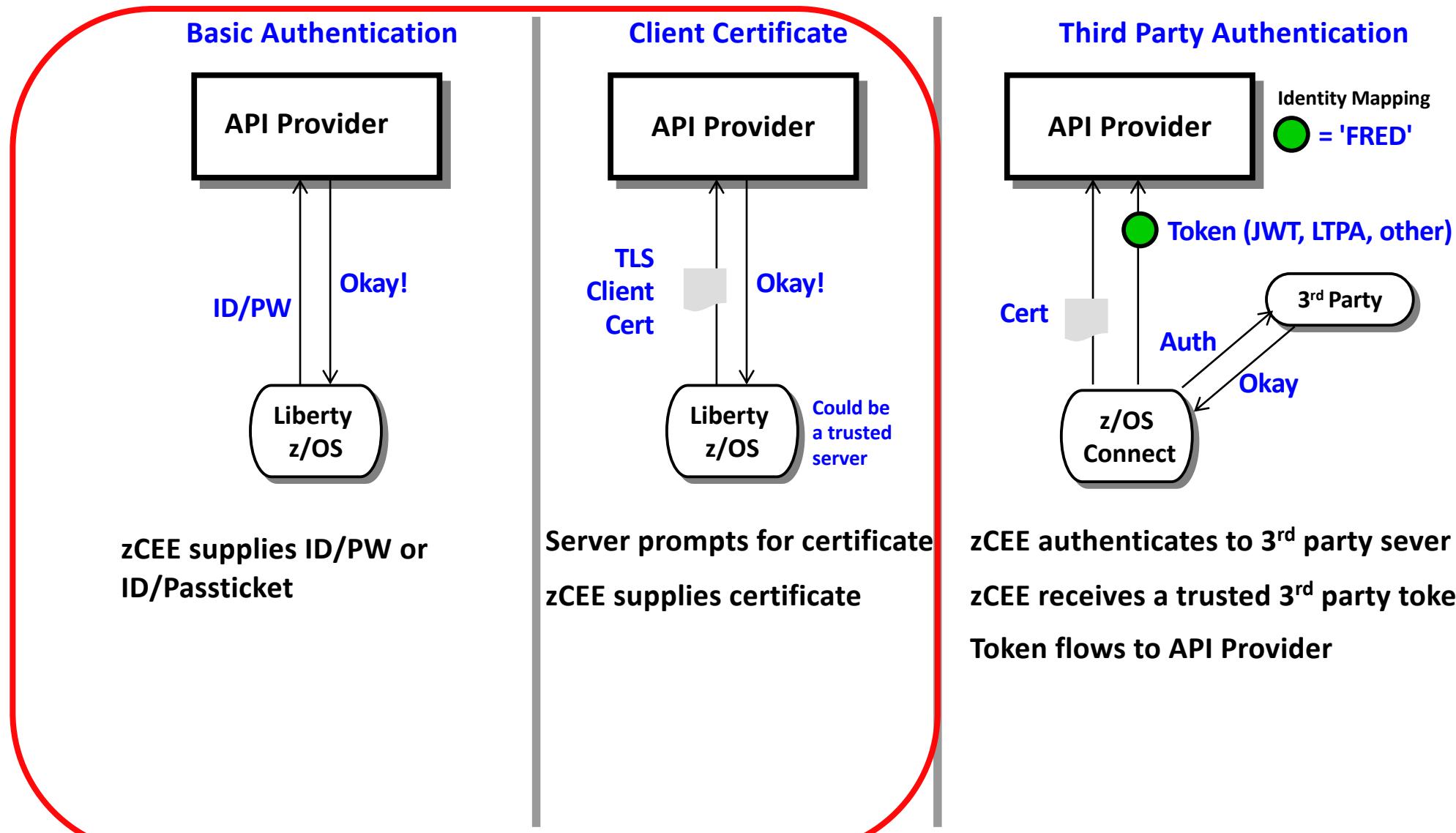
So, the required SAF EJB roles to be defined would be (*invoke* is the only role).

- *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.

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

Several different ways this can be accomplished:



Configuring Basic and/or TLS 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



ID/PW

Liberty
z/OS

zCEE supplies ID/PW or
ID/Passticket

Client Certificate



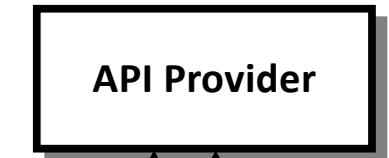
TLS
Client
Cert

Liberty
z/OS

Could be
a trusted
server

Server prompts for certificate
zCEE supplies certificate

Third Party Authentication



Cert

z/OS
Connect

Token (JWT, LTPA, other)

Identity Mapping
Green circle = 'FRED'

3rd Party

Okay

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



Third Party Authentication Examples

Screenshot of the UPS Sign Up page:

The page features a yellow banner at the top stating "UPS is open for business: Service impacts related to Coronavirus ...More". Below the banner is the UPS logo and navigation links for "Sign up / Log in" and "Search or Track".

Sign Up

Already have an ID? [Log in](#)

Use one of these sites.

[Google](#) [Facebook](#)
[Amazon](#) [Apple](#)
[Twitter](#)

Or enter your own information.

* Indicates required field

Name *

Email *

User ID *

Password * [Show](#)

Phone US +1

Screenshot of the myNCDMV Sign In page:

The page has a "Log In" tab selected, with a "Sign Up" link nearby. It features a background image of autumn foliage.

Log In to myNCDMV

Email Address:

Password: [Show Password](#)

Remember Me

Log In **Forgot Password**

Or

[Continue with Apple](#)
[Continue with Facebook](#)
[Continue with Google](#)

Continue as Guest

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.

powered by



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

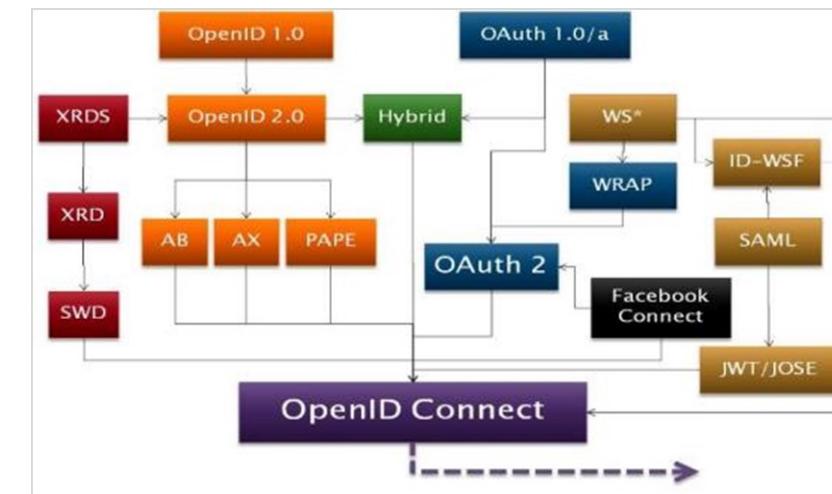
See the YouTube videos:

OAuth 2.0 and OpenID Connect (in plain English)

<https://www.youtube.com/watch?v=996OiexHze0>

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. At the top, it says "Encoded" and displays a long string of characters: eyJraWQi0iI0cWpYLWJrWE9Vd19GX... The bottom right corner of this string has a red oval highlighting the timestamp "Mon Nov 02 2020 11:05:58 GMT-0500 (Eastern Standard Time)". To the right, under "Decoded", the "HEADER" section shows a JSON object with "kid": "4qjX-bkXOUw_F_uccjRMkB9ivMjXSQwj0RrkyRJq8DM" and "alg": "RS256". The "PAYLOAD" section shows a JSON object with "sub": "Fred", "token_type": "Bearer", "scope": ["openid", "profile", "email"], "azp": "rpSsl", "iss": "https://wg31.washington.ibm.com:26213/oidc/endpoint/OP", "aud": "myZcee", "exp": 1604333158, "iat": 1604333858, "realmName": "zCEERealm", and "uniqueSecurityName": "Fred".

Values derived from the OAUTH configuration:

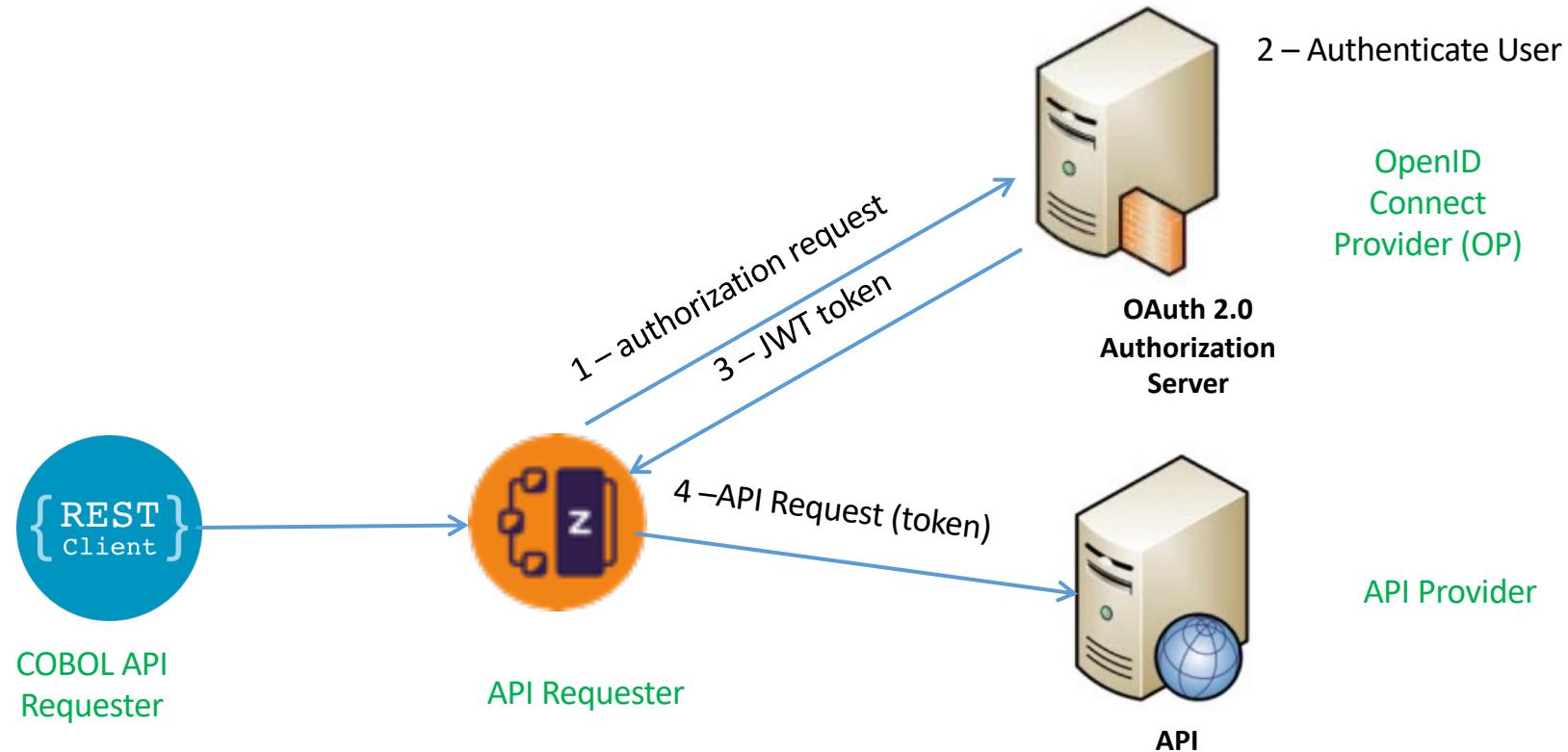
- signatureAlgorithm="RS256"
- accessTokenLifetime="300"
- resourceIds="myZcee"

<https://jwt.io>

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.

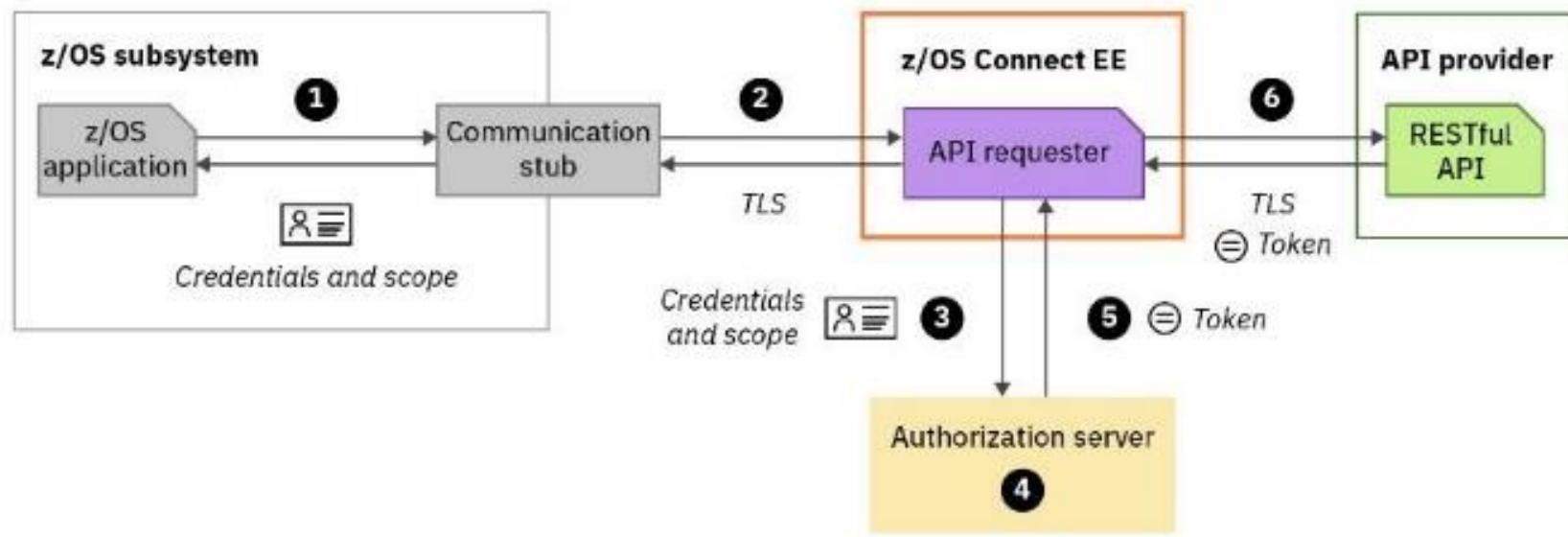
z/OS Connect OAuth Flow for API requester



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=996OjexHze0>

Configuring OAuth support – BAQRINFO copy book



```
File Edit Settings View Communication Actions Window Help
Menu Utilities Compilers Help
BROWSE ZCEE30.SBAQCOB(BAQRINFO) Line 0000000028 Col 001 080
Command ==> - Scroll ==> PAGE
01 BAQ-REQUEST-INFO.
 03 BAQ-REQUEST-INFO-COMP-LEVEL PIC S9(9) COMP-5 SYNC VALUE 4.
 03 BAQ-REQUEST-TINFO-USER
    05 BAQ-OAUTH.
      07 BAQ-OAUTH-USERNAME PIC X(256).
      07 BAQ-OAUTH-USERNAME-LEN PIC S9(9) COMP-5 SYNC VALUE 0.
      07 BAQ-OAUTH-PASSWORD PIC X(256).
      07 BAQ-OAUTH-PASSWORD-LEN PIC S9(9) COMP-5 SYNC VALUE 0.
    07 BAQ-OAUTH-CLIENTID PIC X(256).
    07 BAQ-OAUTH-CLIENTID-LEN PIC S9(9) COMP-5 SYNC VALUE 0.
    07 BAQ-OAUTH-CLIENT-SECRET PIC X(256).
    07 BAQ-OAUTH-CLIENT-SECRET-LEN PIC S9(9) COMP-5 SYNC VALUE 0.
    07 BAQ-OAUTH-SCOPE-PTR USAGE POINTER.
    07 BAQ-OAUTH-SCOPE-LEN PIC S9(9) COMP-5 SYNC 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
```

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

wg31 master

File Edit Settings View Communication Actions Window Help

Menu Utilities Compilers Help

BROWSE ZCEE30.SBAQCOB(BAQHCONC)

Line 000000020 Col 001 080
Scroll ==> PAGE

```
* 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)
```

MA A

Connected to remote server/host wg31z using lu/pool TCP00111 and port 23

wg31 master

File Edit Settings View Communication Actions Window Help

File Edit Edit_Settings Menu Utilities Compilers Test Help

EDIT USER1.ZCEE30.SOURCE(GETAPI) - 01.02 Columns 00001 00072
Command ==> Scroll ==> PAGE

***** **** 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
000024
000025
000026
000027
```

MA A

Connected to remote server/host wg31z using lu/pool TCP00111 and port 23

28/01/9

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
```

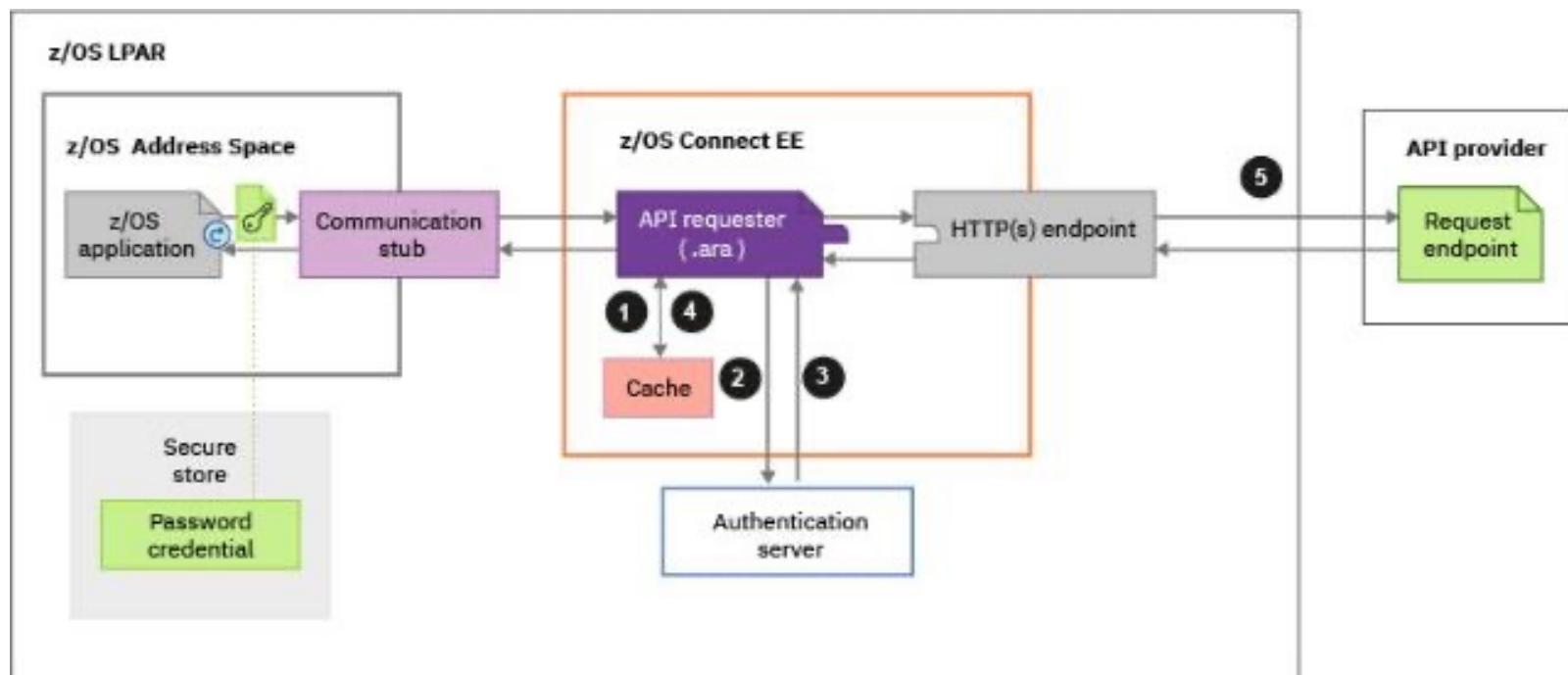
auser is not mapped to a valid RACF identity

```
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 )
```

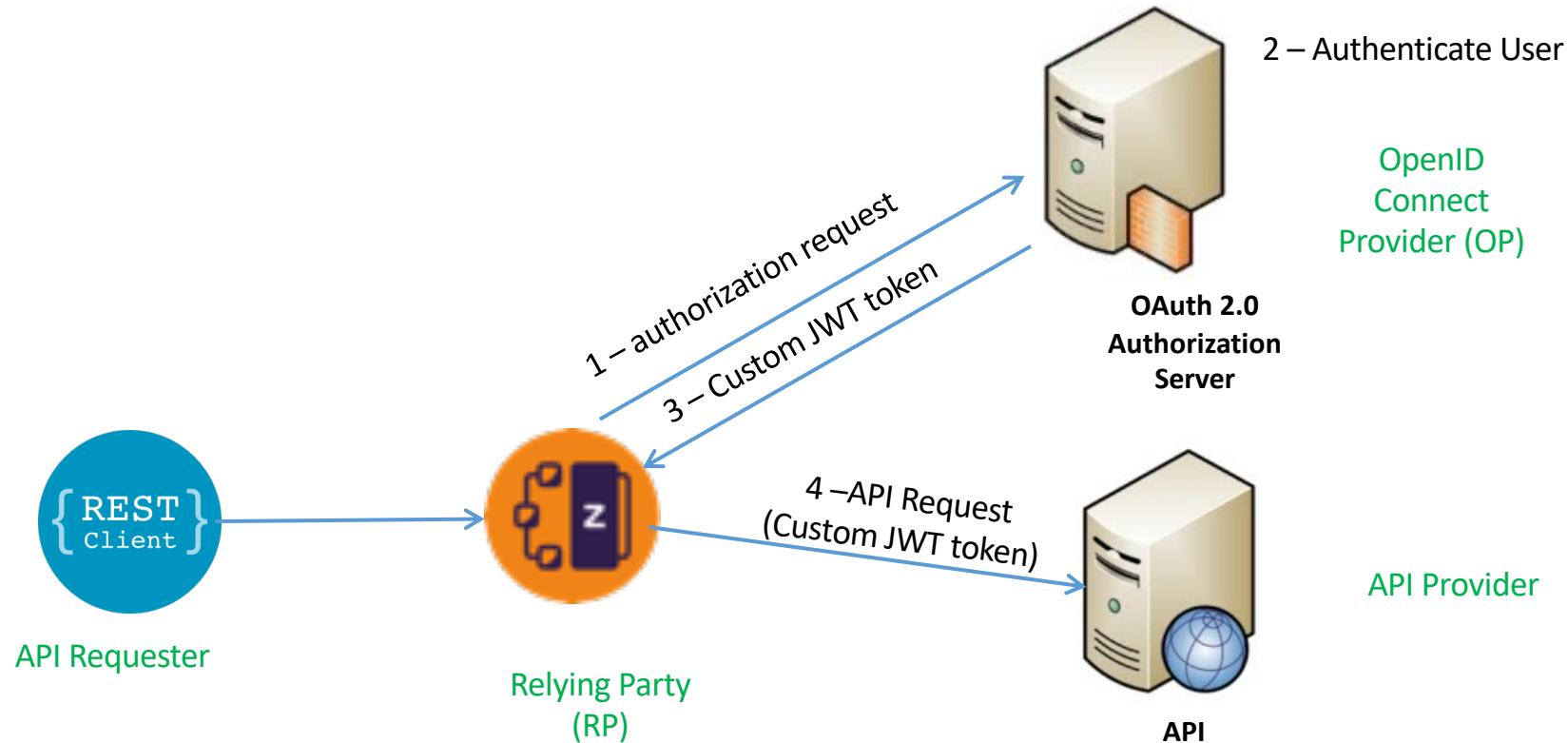
)
distuser2 is mapped to RACF identity USER2 which has no access to the EJBRole protecting z/OS Connect

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

BROWSE ZCEE30.SBAQCOB(BAQRINFO) Line 0000000028 Col 001 080
Command ==> - Scroll ==> PAGE

```
01 BAQ-REQUEST-INFO.  
 03 BAQ-REQUEST-INFO-COMP-LEVEL PIC S9(9) COMP-5 SYNC VALUE 4.  
 03 BAQ-REQUEST-INFO-USER.  
 05 BAQ-OAUTH.  
   07 BAQ-OAUTH-USERNAME PIC X(256).  
   07 BAQ-OAUTH-USERNAME-LEN PIC S9(9) COMP-5 SYNC  
      VALUE 0.  
   07 BAQ-OAUTH-PASSWORD PIC X(256).  
   07 BAQ-OAUTH-PASSWORD-LEN PIC S9(9) COMP-5 SYNC  
      VALUE 0.  
   07 BAQ-OAUTH-CLIENTID PIC X(256).  
   07 BAQ-OAUTH-CLIENTID-LEN PIC S9(9) COMP-5 SYNC  
      VALUE 0.  
   07 BAQ-OAUTH-CLIENT-SECRET PIC X(256).  
   07 BAQ-OAUTH-CLIENT-SECRET-LEN PIC S9(9) COMP-5 SYNC  
      VALUE 0.  
   07 BAQ-OAUTH-SCOPE-PTR USAGE POINTER.  
   07 BAQ-OAUTH-SCOPE-LEN PIC S9(9) COMP-5 SYNC  
      VALUE 0.  
05 BAQ-AUTHTOKEN.  
 07 BAQ-TOKEN-USERNAME PIC X(256).  
 07 BAQ-TOKEN-USERNAME-LEN PIC S9(9) COMP-5 SYNC  
    VALUE 0.  
 07 BAQ-TOKEN-PASSWORD PIC X(256).  
 07 BAQ-TOKEN-PASSWORD-LEN PIC S9(9) COMP-5 SYNC  
    VALUE 0.  
05 BAQ-ZCON-SERVER-URI PIC X(256)  
    VALUE SPACES.
```

MA A 04/015
Connected to remote server/host wg31z using lu/pool TCP00145

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.



API Requester – JWT Custom flow

WG31 - 3270

Menu Utilities Compilers Help

BROWSE ZCEE30.SBAQCOB(BAQHCONC)

Line 000000020 Col 001 080

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-TOKEN-USERNAME      PIC X(22)
    VALUE 'BAQHAPI-Token-Username'.
 77 BAQR-TOKEN-PASSWORD      PIC X(22)
    VALUE 'BAQHAPI-Token-Password'.
 77 BAQR-TOKEN-CUSTOM-PARMS  PIC X(25)
    VALUE 'BAQHAPI-Token-CustomParms'.
 77 BAQR-TOKEN-CUSTOM-HEADERS PIC X(27)
    VALUE 'BAQHAPI-Token-CustomHeaders'.

* 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)
    VALUE 'BAQHAPI-Server-Host'.
 77 BAQZ-SERVER-PORT         PIC X(19)
    VALUE 'BAQHAPI-Server-Port'.
 77 BAQZ-SERVER-TIMEOUT      PIC X(22)
    VALUE 'BAQHAPI-Server-Timeout'.
 77 BAQZ-SERVER-USERNAME     PIC X(28)
    VALUE 'BAQHAPI-Server-Username'.
```

MA A

Connected to remote server/host wg31 using lu/pool TCP00112 and port 23

Adobe PDF on

WG31 - 3270

File Edit Edit_Settings Menu Utilities Compilers Test Help

EDIT JOHNSON.ZCEE.SOURCE(BAQZUSER) - 01.01 Columns 00001 00072

Line 000000020 Col 001 080

Command ==>

```
***** **** Top of Data ****
==MSG> -CAUTION- Data contains invalid (non-display) characters. Use command
==MSG>      ==> FIND P'. to position cursor to these
000001 IDENTIFICATION DIVISION.
000002 PROGRAM-ID. HBRMINM.
000003 ENVIRONMENT DIVISION.
000004 DATA DIVISION.
000005 WORKING-STORAGE SECTION.
000006 01 MY-USER PIC {10} VALUE 'myUsername'.
000007 01 MY-PSWD PIC {10} VALUE 'myPassword'.
000008 ...
000009 ...
000010 ...
000011 ...
000012 ...
000013 ...
000014 MOVE BAQR-TOKEN-USERNAME TO
000015   BAQ-ZCON-PARM-NAME OF BAQ-ZCON-PARMS(1).
000016 SET BAQ-ZCON-PARM-ADDRESS OF BAQ-ZCON-PARMS(1) TO
000017   address of MY-USER.
000018 MOVE LENGTH OF MY-USER TO
000019   BAQ-ZCON-PARM-LENGTH(1) OF BAQ-ZCON-PARMS(1).
000020 ...
000021 MOVE BAQR-TOKEN-PASSWORD TO
000022   BAQ-ZCON-PARM-NAME OF BAQ-ZCON-PARMS(2).
000023 SET BAQ-ZCON-PARM-ADDRESS OF BAQ-ZCON-PARMS(2) TO
000024   ADDRESS OF MY-USER.
000025 MOVE LENGTH OF MY-USER TO
000026   BAQ-ZCON-PARM-LENGTH(1) OF BAQ-ZCON-PARMS(2).
***** **** Bottom of Data ****
```

MA A

Connected to remote server/host wg31 using lu/pool TCP00112 and port 23

Adobe PDF on Documents*.pdf

05/009



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_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" />
```

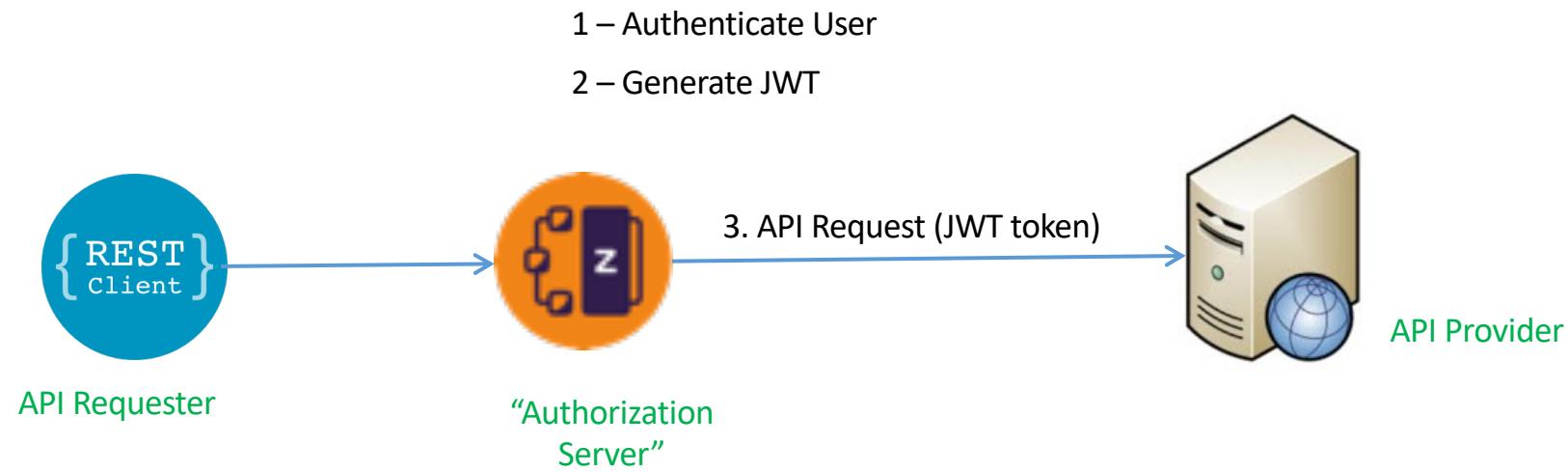
Response Token Example 2

```
<tokenRequest credentialLocation="body"  
    requestMethod="POST"  
    // Use XML escaped characters in requestBody  
    requestBody="'{&quot;apiuser&quot;:&quot;${userid}&quot;,&quot;apipassword&quot;:&quot;${password}&quot;}'" />
```

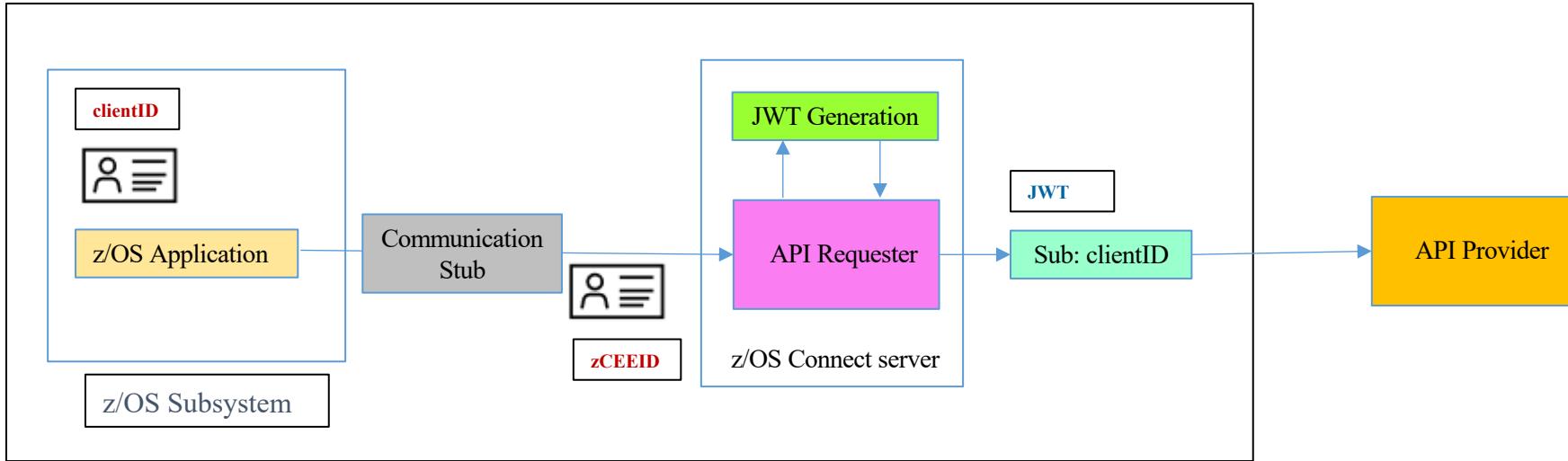
Response Token

```
<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*

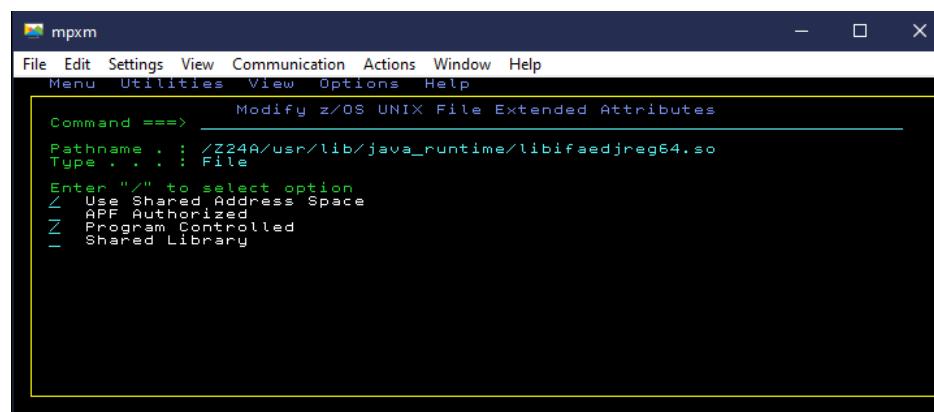
RDEFINE SURROGAT **clientID.BAQASSRT UACC(NONE) OWNER(SYS1)**
PERMIT clientID.BAQASSRT CLASS(SURROGAT) ACCESS(READ) ID(*zCEEID*)

OR

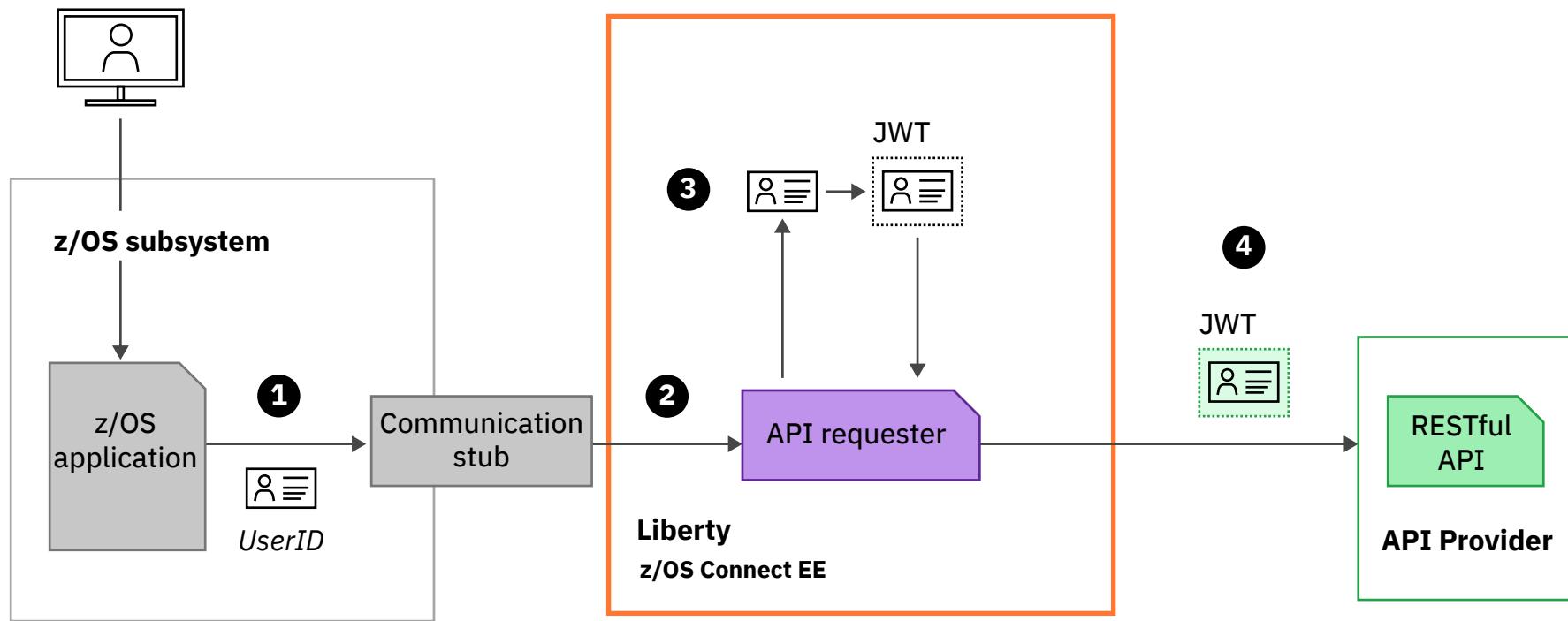
RDEFINE SURROGAT ***.BAQASSRT UACC(NONE) OWNER(SYS1)**
PERMIT *.BAQASSRT CLASS(SURROGAT) ACCESS(READ) ID(*zCEEID*)
SETROPTS RACLIST(SURROGAT) REFRESH

- *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.

Configuring JWT Generation support



```
<zosconnect_endpointConnection id="conn1"  
    host="http://api.server.com" port="8080"  
    authenticationConfigRef="jwtConfig" />  
<zosconnect_endpointConnection id="conn2"  
    host="http://api.server.com" port="8080"  
    authenticationConfigRef="jwtConfig" />  
<zosconnect_authTokenLocal id="jwtConfig"  
    tokenGeneratorRef="jwtBuilder"  
    header="Authorization" >  
    <claims>{"scope":"Scope1"}</claims>  
<zosconnect_authTokenLocal id="jwtConfig"  
    tokenGeneratorRef="jwtBuilder"  
    header="Authorization" >  
    <claims>{"scope":"Scope2"}</claims>  
<jwtBuilder id="jwtBuilder"  
    scope="scope"  
    audiences="myApp1"  
    jti="true"  
    signatureAlgorithm="RS256"  
    keyStoreRef="myKeyStore"  
    keyAlias="jwtsigner"  
    issuer="z/OS Connect EE Default"/>
```



server XML Configuration

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

Configure the Liberty jwtBuilder element in server.xml.

Configure the zosconnect_authTokenLocal element, specifying any additional private claims required and the name of the header used to send the JWT to the endpoint.

header default value is Authorization

Finally, reference the JWT configuration from the zosconnect_endpointConnection element.

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 Swagger 2.0 REST APIs
- Developing API requesters for OpenAPI 3 REST APIs
- Configuration and Security considerations for API requesters



Thank you for listening and your questions.



z/OS Connect Wildfire Github Site

<https://ibm.biz/BdPRGD>

The screenshot shows two GitHub repository pages side-by-side, illustrating the content of the Wildfire workshop.

Left Repository (ibm-wsc/zCONNEE-Wildfire-Workshop):

- Code:** master branch, 1 branch, 0 tags.
- Commits:** emitchj (8e503b9) - Add files via upload (3 days ago).
 - AdminSecurity - Delete ZC2OMVS2.jcl
 - OpenAPI2 - Delete Developing
 - cobol - Add files via upload
 - xml - Add files via upload
 - README.md - Update README.md
 - ZADMIN - 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
 - ZREQUEST - Introduction to zOS Co... - Add files via upload
 - ZOS Connect EE 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 IBM Washington Systems Center. It should be referenced frequently for updates to the presentations, exercises, samples and other material.

Right Repository (ibm-wsc/zCONNEE-Wildfire-Workshop):

- Code:** master branch, zCONNEE-Wildfire-Workshop / APIRequesters /.
- Commits:** emitchj (428fc6c) - Add files via upload (5 days ago).
 - Developing CICS API Requester Applications.pdf - Add files via upload (5 days ago)
 - Developing IMS API Requester Applications.pdf - Add files via upload (5 days ago)
 - Developing MVS Batch API Requester Applications.pdf - Add files via upload (5 days ago)
- Environments:** 1, github-pages (Active)
- Languages:** (not explicitly listed)

- Contact your IBM representative to schedule access to these exercises