



ZCREQUEST - IBM z/OS Connect

An introduction to API Requesters

API Requester Code and Security Considerations

Mitch Johnson

mitchj@us.ibm.com

Washington System Center



Notes and Disclaimers



- The information in this presentation was derived from various product documentation web sites.
- Additional information included in this presentation was distilled from years of experience implementing security using RACF with z/OS products like CICS, IMS, Db2, MQ, etc. as well as Java runtimes environments like WebSphere Application Server and WebSphere Application Server Liberty which is commonly called Liberty.
- There will be additional information on slides that will be designated as Tech/Tips. These contain information that at perhaps at least interesting and hopefully, useful to the reader.
- The examples, tips, etc. present in this material are based on firsthand experiences and are not necessarily sanctioned by Liberty or z/OS Connect product areas.

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



ZCINTRO - IBM z/OS Connect An introduction and overview

Mitch Johnson

mitchj@us.ibm.com

Washington System Center

mitchj@us.ibm.com



© 2017



ZCADMIN – IBM z/OS Connect Administration

WebSphere Liberty Profile with
IBM z/OS Connect (OpenAPI 2) and/or
IBM z/OS Connect (OpenAPI 3)
Administration



Mitch Johnson

mitchj@us.ibm.com

Washington Systems Center

mitchj@us.ibm.com

© 2017, 2022 IBM Corporation
Slide 1

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

mitchj@us.ibm.com

© 2018, 2022 IBM Corporation

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



The image displays three GitHub repository pages side-by-side, each featuring a red oval highlighting specific file uploads.

Left Repository: <https://github.com/ibm-wsc/zCONNEE-Wildfire-Workshop>

- Commits:** emitchj Add files via upload (8e503b9)
- Files:**
 - AdminSecurity Delete ZC2OMVS2.jcl
 - OpenAPI2 Delete Developing
 - cobol Add files via upload
 - xml Add files via upload
 - README.md Update README.md
 - ZCADMIN - zOS Connect Administrat... Add files via upload
 - ZCESEC - zOS Connect Security.pdf Add files via upload
 - ZCINTRO - Introduction to zOS Conn... Add files via upload
 - ZCREQUEST - Introduction to zOS Co... Add files via upload** (highlighted by a red oval)
 - zOS Connect EE V3 Advanced Topics ... Add files via upload
 - zOS Connect EE V3 Getting Started.pdf Add files via upload
- README.md**
- Notes:** This repository contains material from the z/OS Connect EE Wildfire workshops run by the IBM Center. It is should be referenced frequently for updates to the presentations, exercises, sam...

Middle Repository: <https://github.com/ibm-wsc/zCONNEE-Wildfire-Workshop>

- Commits:** emitchj Delete Developing (d880029 on Apr 23)
- Files:**
 - Developing CICS API Requester Applications.pdf Add files via upload (2 months ago)
 - Developing IMS API Requester Applications.pdf Add files via upload (2 months ago)
 - Developing MVS Batch API Requester Applications.pdf Add files via upload (2 months ago)

Bottom Repository: <https://github.com/ibm-wsc/zCONNEE-Wildfire-Workshop>

- Commits:** emitchj Add files via upload (428fc6c 5 days ago)
- Files:**
 - 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)

- **Contact your IBM representative to schedule access to these exercises**

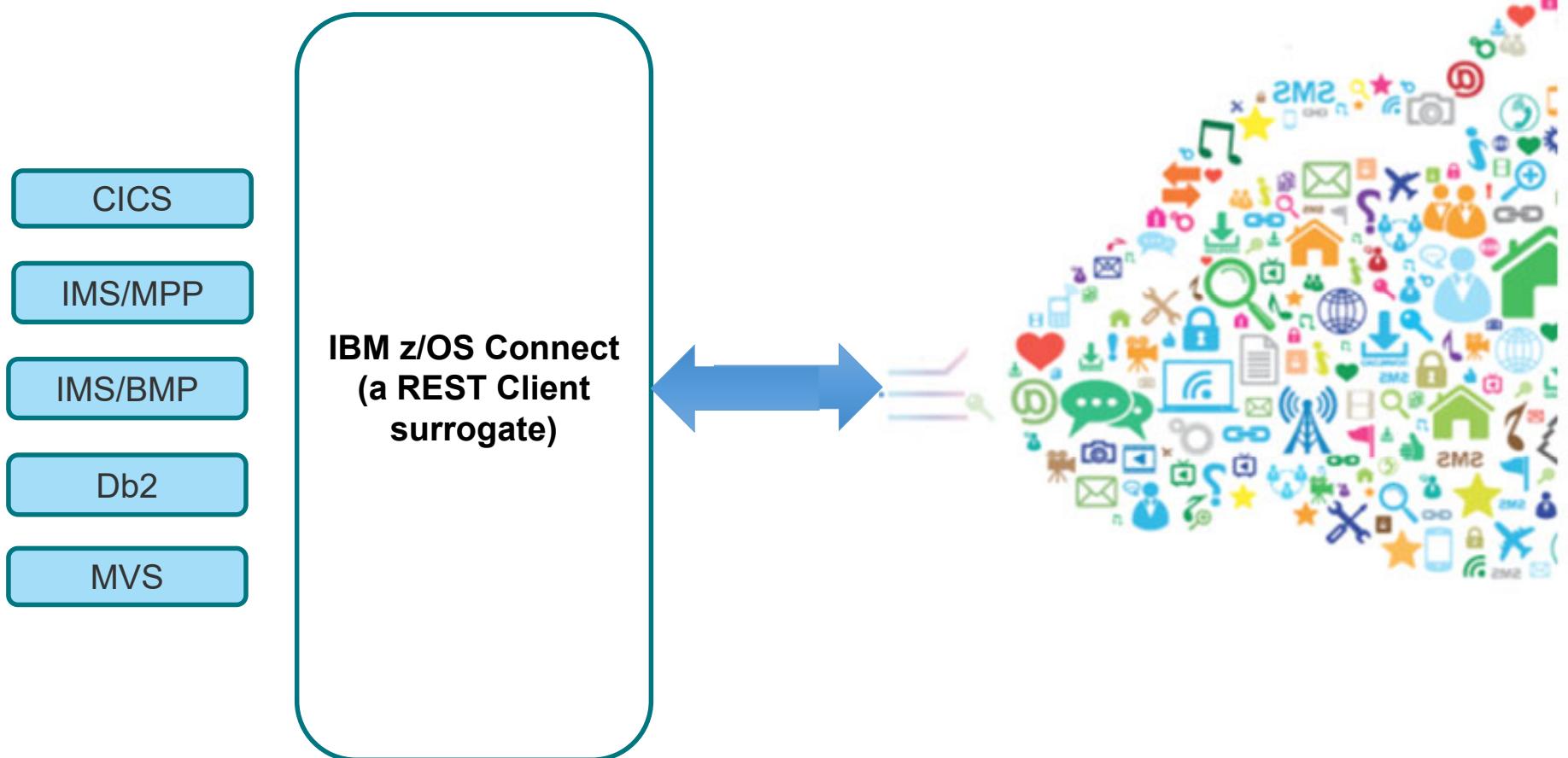
z/OS Connect EE exposes z/OS resources to the “cloud” via RESTful APIs



+ HCL and Rocket Software

*Other Vendors or your own implementation

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



/but_first, what_is_REST?

What makes an API “RESTful”?



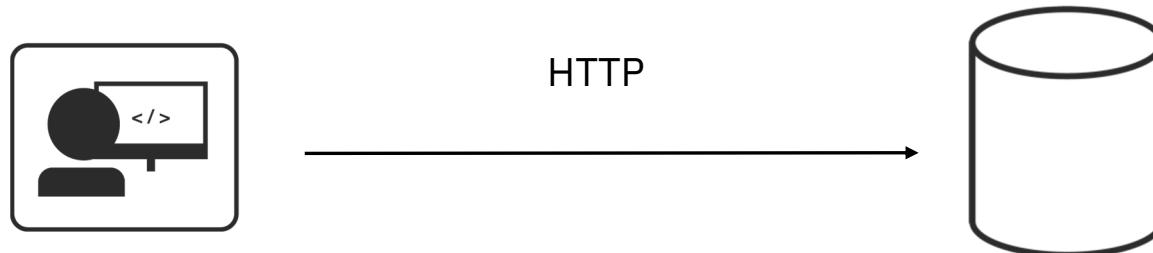
REST is architectural programming style

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

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

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

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



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



Key Principles of REST

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

POST
GET
PUT
DELETE

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

Use Path and Query parameters to refine the request

URI path identifies a resource (or lists of resources)

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

Request/Response Body is used to represent the data object

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



REST vs RESTful

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

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

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

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



RESTful Examples

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

GET /account?number=1234

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

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

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

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

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

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



Not every REST API is a RESTful API

- (How to know if an API is not RESTful)

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

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

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

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

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



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

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

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



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



Strive to design API to use RESTful properties

1. Use the same URIs for the same resource with the appropriate method for operations

```
GET http://www.acme.com/customers/12345
```

```
PUT http://www.acme.com/customers/12345?address=10%20New%20Street
```

2. Use the same JSON property names between request and response messages

```
POST http://www.acme.com/customers/12345  
BODY { "name": "Joe Bloggs",  
       "address": "10 Old Street",  
       "phoneNo": "01234 0123456" }
```



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

3. Use JSON name/value pairs

```
PUT http://www.acme.com/customers/12345  
BODY { "address" : "10 New Street" }
```



```
RESPONSE HTTP 200 OK
```



Why is REST popular?

Ubiquitous Foundation	It's based on HTTP, which operates on TCP/IP, which is a ubiquitous networking topology.
Relatively Lightweight	Compared to other technologies (for example, SOAP/WSDL), the REST/JSON pattern is relatively light protocol and data model, which maps well to resource-limited devices.
Relatively Easy Development	Since the REST interface is so simple, developing the client involves very few things: an understanding of the URI requirements (path, parameters) and any JSON data schema.
Increasingly Common	REST/JSON is becoming more and more a de facto "standard" for exposing APIs and Microservices. As more adopt the integration pattern, the more others become interested.
Stateless	REST is by definition a stateless protocol, which implies greater simplicity in topology design. There's no need to maintain, replicate or route based on state.

How do we describe a REST API?



/oai/open_api_initiative

The industry standard framework for describing REST APIs

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



Why use OpenAPI?

- It is more than just an API framework



There are a number of tools available to aid consumption:

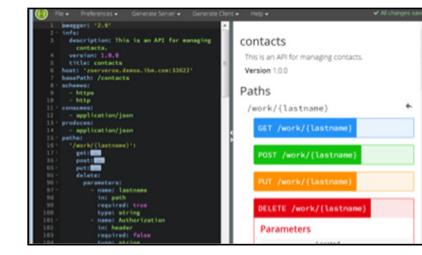
Code Generation* - create stub code to consume APIs from various languages



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



Editors - allows API developers to design their OpenAPI documents.



* z/OS Connect API Requester

+z/OS Connect, MQ REST support, Zowe

Important - You may have used or heard of the term Swagger with the use of APIs. As the use of APIs has grown this term has become in some respects misleading. To be more precise, OpenAPI refers to the API specifications (OpenAPI 2 and OpenAPI3) where Swagger refers to the tooling used to implement the specifications.



An OpenAPI 2 versus an OpenAPI 3 specification document

JSON
format

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

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

```

YAML
Format*



Tech-Tip: Swagger (OpenAPI 2) Specification Example

The image displays two side-by-side browser windows, each showing a JSON-based Swagger/OpenAPI 2 specification. The left window shows the full API definition, while the right window provides a detailed view of a specific request schema.

Left Window (Full API Definition):

```
swagger: "2.0"
info:
  description: ""
  version: "1.0.0"
  title: "miniloan"
  basePath: "/miniloan"
schemes:
  0: "https"
  1: "http"
consumes:
  0: "application/json"
produces:
  0: "application/json"
path:
  /loan:
    post:
      tags:
        0:
          operationId: "miniloan"
          postMiniloanService"
      parameters:
        0:
          name: "Authorization"
          in: "header"
          required: false
          type: "string"
        1:
          in: "body"
          name: "postMiniloanService_request"
          description: "request body"
          required: true
          schema:
            $ref: "#/definitions/postMiniloanService_request"
      responses:
        200:
          description: "OK"
          schema:
```

Right Window (Detailed Request Schema):

```
schema:
  $ref: "#/definitions/postMiniloanService_response_200"
definitions:
  postMiniloanService_request:
    type: "object"
    properties:
      MINILOAN_COMMAREA:
        type: "object"
        properties:
          name:
            type: "string"
            maxLength: 20
          creditscore:
            type: "integer"
            minimum: 0
            maximum: 1000000000000000000
          yearlyIncome:
            type: "integer"
            minimum: 0
            maximum: 1000000000000000000
          age:
            type: "integer"
            minimum: 0
            maximum: 9999999999
          amount:
            type: "integer"
            minimum: 0
            maximum: 1000000000000000000
          effectiveDate:
            type: "string"
            maxLength: 8
          yearlyRepayment:
            type: "integer"
            minimum: 0
            maximum: 1000000000000000000
    postMiniloanService_response_200:
      type: "object"
```

mitchj@us.ibm.com

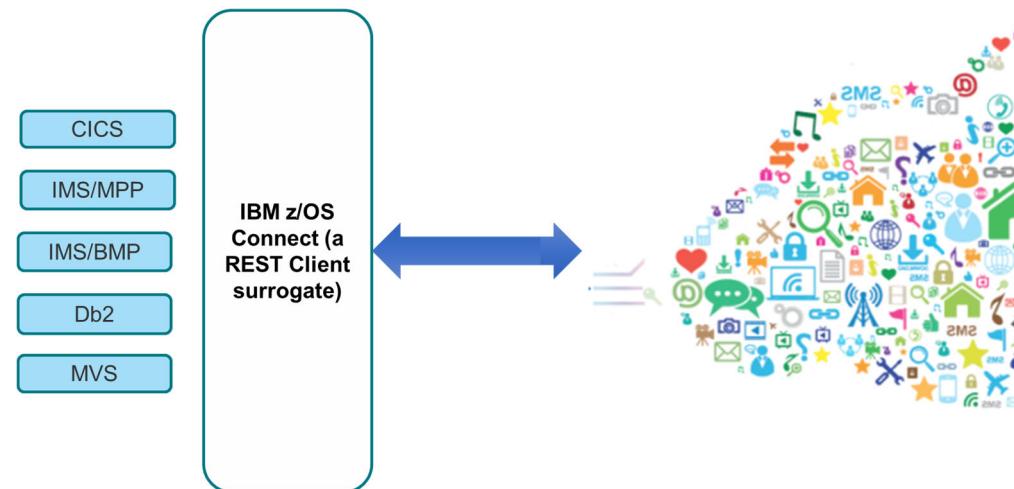
© 2018, 2022 IBM Corporation

Page 19



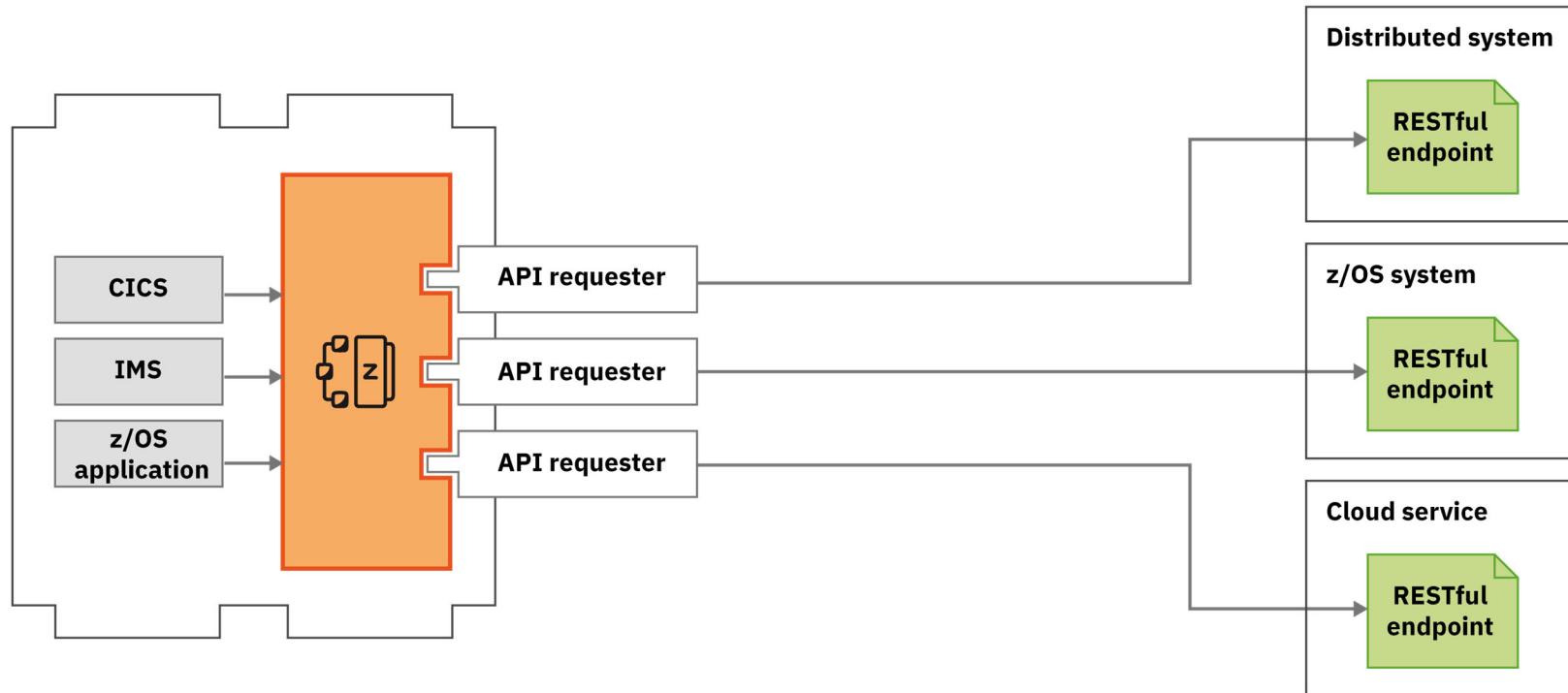
/api_toolkit/apiRequesters

Quick and easy API mapping.





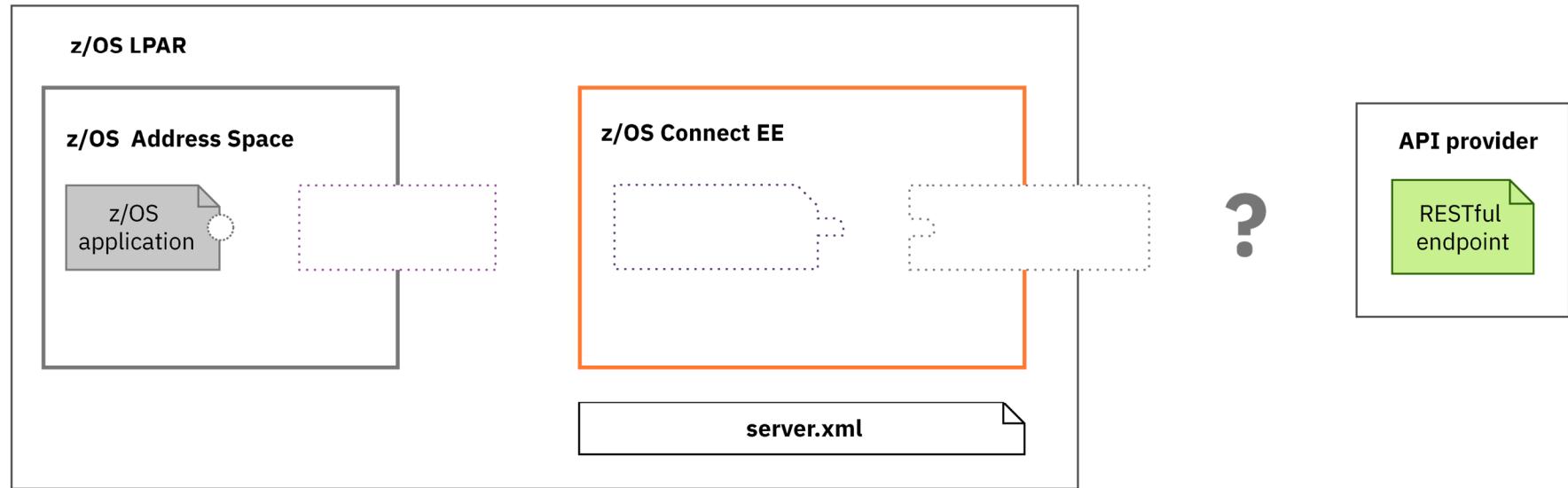
Use API requester to call external APIs from z/OS assets





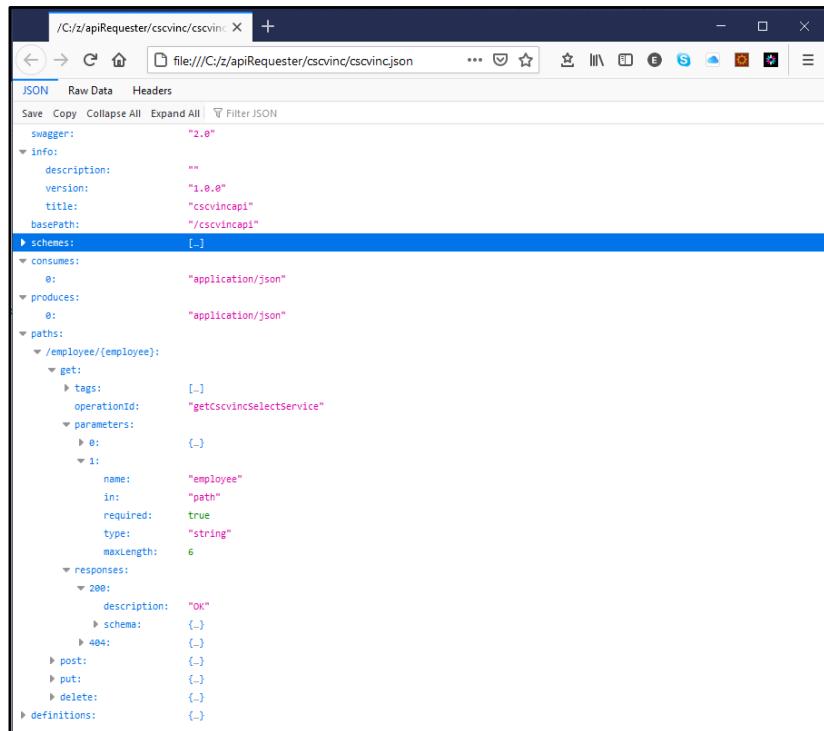
Steps to calling an external API

Starting point



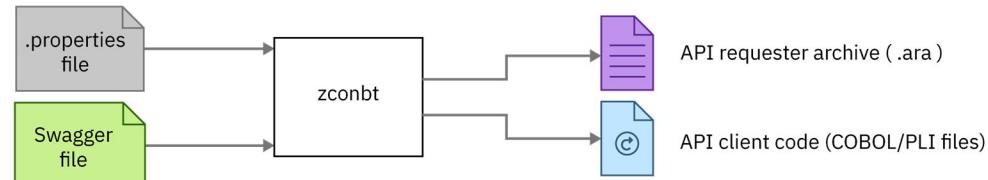
Calling an external API requires the API's specification document

Start with the API's specification and generate the API requester archive file and API client code



The screenshot shows a JSON editor window displaying a Swagger JSON file. The file defines an API endpoint for employees, including parameters, responses, and definitions. Key sections include:

- swagger:** "2.0"
- info:** description: "", version: "1.0.0", title: "cscvincapi", basePath: "/cscvincapi"
- schemes:** []
- consumes:** @: "application/json"
- produces:** @: "application/json"
- paths:** /employee/{employee}:
 - get:** tags: [], operationId: "getCscvincSelectService", parameters: @: {}, 1:
 - name: "employee", in: "path", required: true, type: "string", maxLength: 6
 - responses:** 200:
 - description: "OK", schema: {}
 - 404: @: {}
- post:** @: {}
- put:** @: {}
- delete:** @: {}
- definitions:** @: {}



properties file#

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

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



COBOL working storage implications

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

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

mitchj@us.ibm.com

```
File Edit Format View Help
* ++++++
06 RespBody.

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

09 memberContactsResponse OCCURS 255.

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

12 umi.
15 umi2-length
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).
```

© 2018, 2022 IBM Corporation

Page 24



Consider adding constraints to the properties

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

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

```
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 middleName2
```



There are API Requester generation properties are available to help

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

defaultArrayMaxItems - Specify the maximum array boundary to apply when no maximum occurrence information (maxItems) is implied in the Swagger. The value of this parameter can be a positive integer in the range 1 - 32767. By default, **defaultArrayMaxItems** is set to 255.

defaultCharacterMaxLength - Specify the default array length of character data in characters for mappings where no length is implied in the JSON schema document. When **characterVarying** is set to YES, the value of this parameter can be a positive integer in the range of 1 to 32767. When **characterVarying** is set to NO or NULL the value of this parameter can be a positive integer in the range of 1 to 16777214. By default, **defaultCharacterMaxLength** is set to 255.

characterVarying - Specifies how variable-length character data is mapped to the language structure.

- NO - Variable-length character data is mapped as fixed-length strings.
- NULL - Variable-length character data is mapped to null-terminated strings (defaultCharacterMaxLength + 1)
- YES - Variable-length character data is mapped to a CHAR VARYING data type in PL/I. In COBOL variable-length character data is mapped to an equivalent representation that consists of two related elements: the **data-length** and the **data**. By default, **characterVarying** is set to YES.

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

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

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

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



The number of specific entries can be ambiguous

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

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

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

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

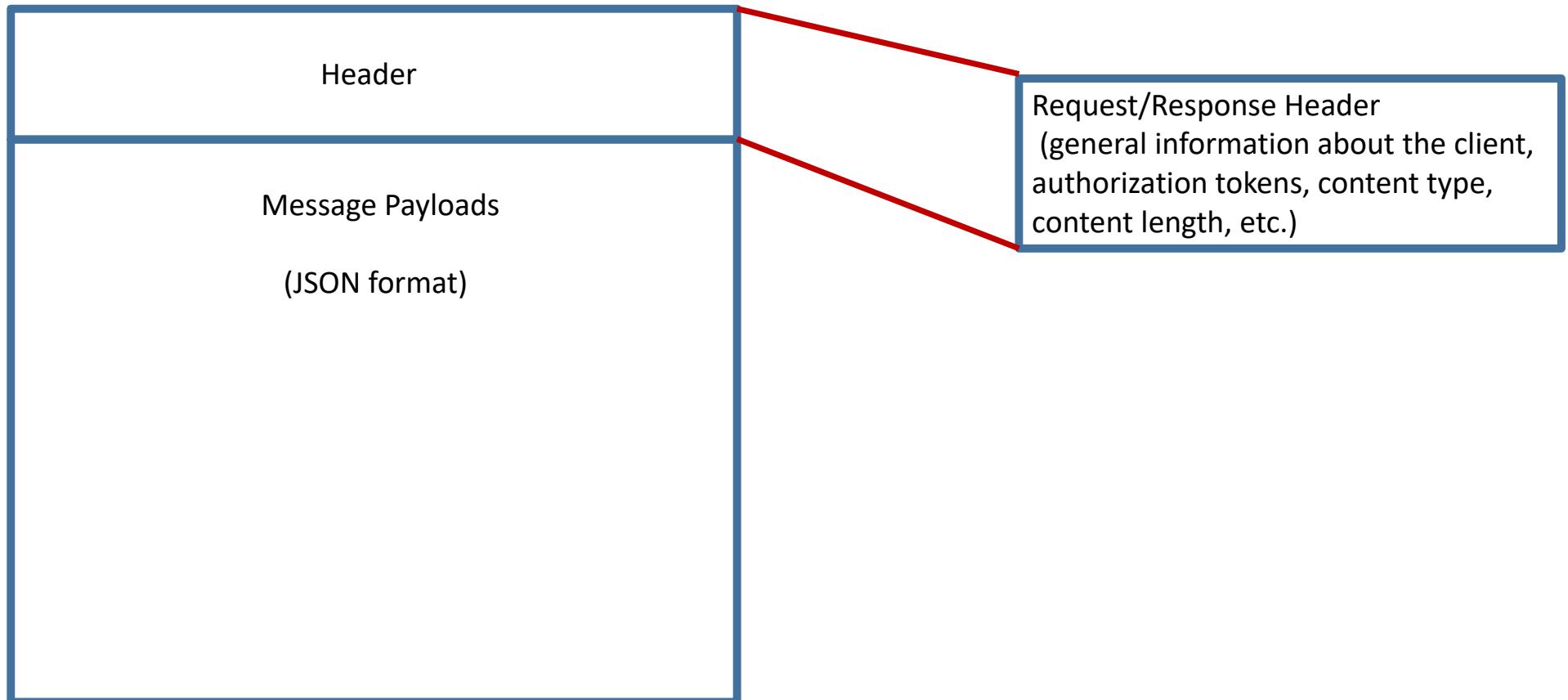
  MOVE num of PARM-DATA TO employee IN PUT-REQUEST.
  MOVE LENGTH of employee in PUT-REQUEST to
    employee-length IN PUT-REQUEST.

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

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



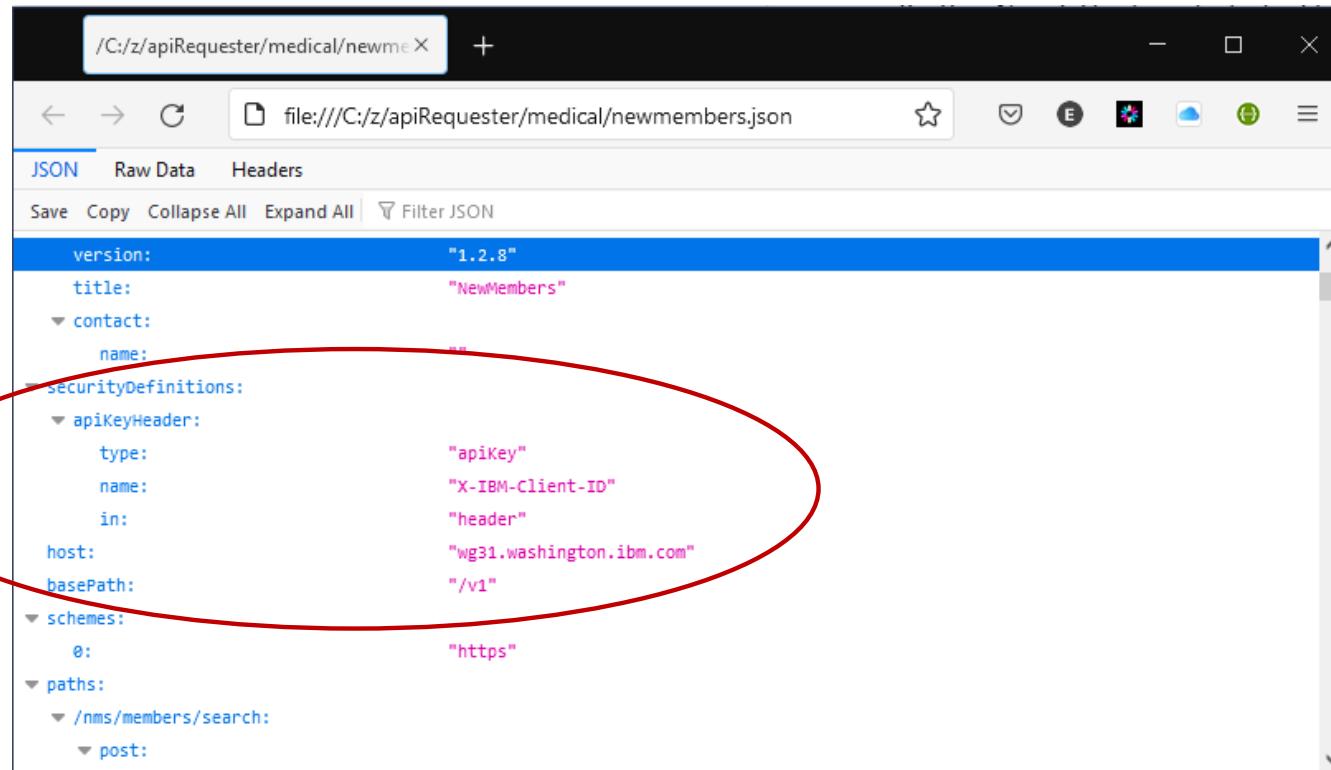
Request and Response Message Layout



An API key(aka password) may be required for accessing an API



The details can 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 a "apiKeyHeader" object. This object has "type": "apiKey", "name": "X-IBM-Client-ID", and "in": "header". Below it is a "host" field with the value "wg31.washington.ibm.com" and a "basePath" field with the value "/v1". The "securityDefinitions" field is part of a larger object that also includes "version": "1.2.8", "title": "NewMembers", and a "contact" object with a "name" field.

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

Via a HTTP header

GET /something HTTP/1.1

X-API-Key: abcdef12345

Or via a query parameter

GET /something?api_key=abcdef12345



Or by including generation properties related to API keys

Use these generation properties to add API key information to the request message when not defined in specification document

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

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

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

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

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



An application provides the API key to the request header

Either way, the generated copy book includes a ReqHeaders structure which can be used to provide values for header properties

The image displays two windows side-by-side. The left window is titled "request.cpy - Notepad" and contains a COBOL copybook definition. It includes several fields and a section for "ReqHeaders". The "ReqHeaders" section is highlighted with a red oval, containing fields like "X-IBM-Client-ID-length" and "X-IBM-Client-ID". The right window is titled "mpz3" and shows an AS/400 edit session of a source program named "USER1.ZCEE.SOURCE(GETAPIEN) - 01.01". The code is written in green and red. A specific section of the code, which initializes the "X-IBM-Client-ID" header, is highlighted with a red oval. The code snippet is as follows:

```

      *-----+
      * Common code
      *-----+
      * initialize working storage variables
      *-----+
      *-----+
      * Set up the data for the API Requester call
      *-----+
      MOVE "abcdef12345" to X-IBM-Client-ID
      MOVE 11 to X-IBM-Client-ID-length
      *-----+
      *-----+
      * 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 GET-REQUEST.
      MOVE LENGTH OF GET-REQUEST TO BAQ-REQUEST-LEN.
      SET BAQ-RESPONSE-PTR TO ADDRESS OF GET-RESPONSE.
      MOVE LENGTH OF GET-RESPONSE TO BAQ-RESPONSE-LEN.
      *-----+

```



Other header properties can be required when accessing an API

The application can also set the values for other header properties that may be required by the API

```
/C:/apiRequester/ATS/decision-se X +  
file:///C:/apiRequester/ATS/decision-service-swagg  
JSON Raw Data Headers  
Save Copy Collapse All Expand All Filter JSON  
parameters:  
  0:  
    name: "ConsumerID"  
    in: "header"  
    description: "Consumer ID"  
    required: true  
    type: "string"  
    default: "ROMEDSv2.0"  
  1:  
    name: "ContextID"  
    in: "header"  
    description: "Context ID"  
    required: true  
    type: "string"  
    default: "RDSv02.0"  
  2:  
    in: "body"  
    name: "body"  
    schema:  
      properties:  
        ClaimInformation: (...)  
      required:  
        0: "ClaimInformation"  
    produces:
```

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



Steps to calling an external API

Use the z/OS Connect build toolkit, e.g., `zconbt`, to generate an API requester archive file and at most 3 copy books per method found in the Swagger

```
zconbt.bat -p=./cscvinc.properties -f=./cscvinc.ara
BAQB0000I: z/OS Connect Enterprise Edition 3.0 Build Toolkit Version 1.5 (20210816-0926).
BAQB0008I: Creating API requester archive from configuration file ./cscvinc.properties.
BAQB0040I: The generated API requester is automatically named cscvincapi_1.0.0 based on the title cscvincapi and version 1.0.0 of the API to be called.
.
.
.
Total 4 operation(s) (success: 4, ignored: 0) defined in api description file: ./cscvinc.json
----- Successfully processed operation(s) -----
operationId: getCsvincSelectService, basePath: /cscvincapi, relativePath: /employee/{employee}, method: GET
- request data structure : CSC00Q01
- response data structure : CSC00P01
- api info file : CSC00I01

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

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

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

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

BTW, the z/OS Connect Build Toolkit can be executed on z/OS



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

cscvinc.properties

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

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



Tech-Tip: Copy books naming convention

- Up to three copy books are generated for each method of each API found in the Swagger document.
 - In the previous example, there were 4 APIs with each having 1 method for a total of 12 copy books.
 - If there is no request message or no response message, then no copy book will be generated. But the messages stills need to have addressable storage in the application's working area.

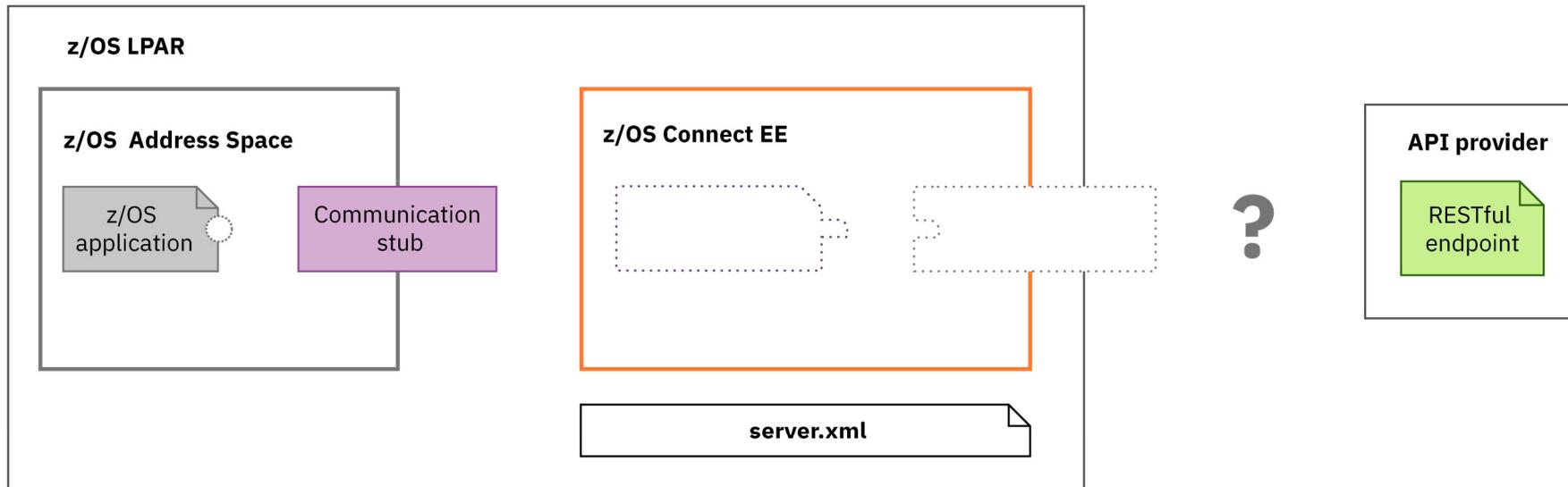
```
* Request and Response
 01 GET-REQUEST.
   10 FILLER                      PIC X(1).
 01 GET-RESPONSE.
   COPY MQ000P01 SUPPRESS.
* Structure with the API information
 01 GET-INFO-OPER1.
   COPY MQ000I01 SUPPRESS.
```

- The naming convention for the generated COBOL copy books is based on the *requesterPrefix* value specified in the properties file. That value was set to CSC in this case, e.g., CSC#####. The next 2 characters in the name are assigned sequentially as each API and method is processed, e.g., CSC00### and CSC01###, and CSC02###.
- The next character will be either a Q, P or an I. A “Q” for a **req**uest copy book, the “P” for a **res**ponse copy book and the “I” for the copy book which contains **in**formation, e.g., method, path name etc. derived from the Swagger document



Steps for involving an external API from a COBOL program

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



Configure a communication stub.

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

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



Steps to calling an external API

Include the generated copy books in a COBOL program

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

  * Copy API Requester required copybook
  *-----*
  COPY BAQRINFO.

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

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

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

API-REQUEST

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

API-RESPONSE

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

API-INFO-OPER1

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



Steps to calling an external API

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

The diagram illustrates the steps to calling an external API. It shows the GETAPI program (left) interacting with the CSC00101 copy book (right) through several windows:

- GETAPI**: The main program window showing assembly code. Red boxes highlight sections:
 - * Set up the data for the API Requester call
 - * Initialize API Requester PTRs & LENs
 - * 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.
- CSC00101**: The main copy book window showing the structure of the API REQUEST. Red boxes highlight:
 - BAQ-APINAME: VALUE 'cscvincap1_1.0.0'.
 - BAQ-APINAME-LEN: VALUE 16.
 - BAQ-APIPATH: VALUE 'S2fcsvincap1%2Femployee%7D'.
 - BAQ-APIPATH-LEN: VALUE 41.
 - BAQ-APIMETHOD: VALUE 'GET'.
 - BAQ-APIMETHOD-LEN: VALUE 3.
- CSC00Q01**: A sub-copy book window showing the JSON schema for the employee array. Red boxes highlight:
 - Employee length: 09 employee-length PIC S9999 COMP-5 SYNC.
 - Employee: 09 employee PIC X(6).
- CSC00P01**: Another sub-copy book window showing the ReqPathParameters. Red boxes highlight:
 - ReqPathParameters: 09 employee-length PIC S9999 COMP-5 SYNC.
 - Employee: 09 employee PIC X(6).



Steps to calling an external API

Access the results

```
GETAPI X
BY REFERENCE BAQ-RESPONSE-LEN.
* The BAQ-RETURN-CODE field in 'BAQRINFO' indicates whether this
* API call is successful.

* When BAQ-RETURN-CODE is 'BAQ-SUCCESS', response is
* successfully returned and fields in RESPONSE copybook
* can be obtained. Display the translation result.
IF BAQ-SUCCESS THEN
    DISPLAY "NUMB: " Numb2 of API_RESPONSE
    DISPLAY "NAME: " name2 of API_RESPONSE
    DISPLAY "ADDRX: " addrx2 of API_RESPONSE
    DISPLAY "PHONE: " phone2 of API_RESPONSE
    DISPLAY "DATEX: " datex2 of API_RESPONSE
    DISPLAY "AMOUNT: " amount2 of API_RESPONSE
    MOVE CEIBRESP of API_RESPONSE to EIBRESP
    MOVE CEIBRESP2 of API_RESPONSE to EIBRESP2
    DISPLAY "EIBRESP: " EIBRESP
    DISPLAY "EIBRESP2: " EIBRESP2
    DISPLAY "HTTP CODE: " BAQ-STATUS-CODE

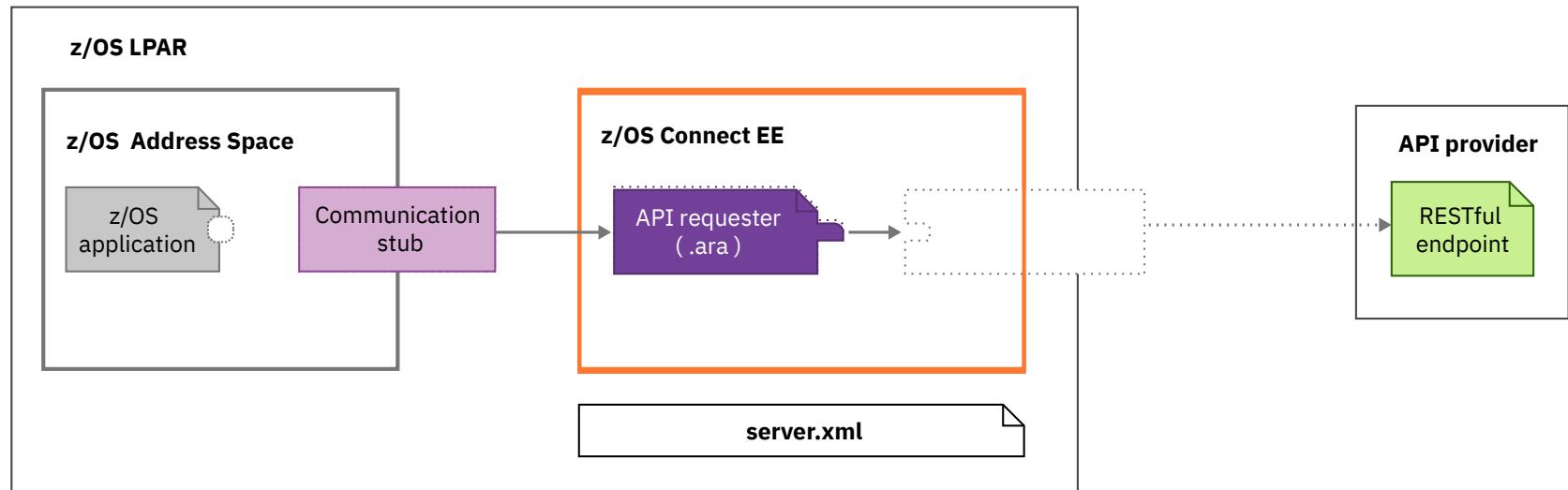
* Otherwise, some error happened in API, z/OS Connect EE server
* or communication stub. 'BAQ-STATUS-CODE' and
* 'BAQ-STATUS-MESSAGE' contain the detailed information
* of this error.
ELSE
    DISPLAY "Error code: " BAQ-STATUS-CODE
    DISPLAY "Error msg: " BAQ-STATUS-MESSAGE
    MOVE BAQ-STATUS-CODE TO EM-CODE
    MOVE BAQ-STATUS-MESSAGE TO EM-DETAIL
    EVALUATE TRUE
* When error happens in API, BAQ-RETURN-CODE is BAQ-ERROR-IN-API.
* BAQ-STATUS-CODE is the HTTP response code of API.
    LINES BAQ-ERROR-IN-API
```

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



Steps to calling an external API

Deploy the API requester (.ara) archive



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



Tech-Tip-Deploying API requester archive files

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

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

```
{"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"}
```

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.wash  
ington.ibm.com:9483/zosConnect/apiRequesters/filea_2.0  
.0", "connection": "fileaAPI"}
```



Steps to calling an external API

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

The screenshot shows a browser window displaying the Swagger JSON definition for the cscvinc API. The URL is /C:/apiRequester/cscvinc/swagger.json. The JSON structure includes information about the API version (2.0), info (host: localhost:8080, basePath: /cscvinc), schemes (https, http), consumes (application/json), produces (application/json), and paths (employee POST endpoint).

The screenshot shows the IBM i terminal window titled CSC02I01. It displays several system parameters related to the API connection:

- BAQ-APINAME: VALUE 'cscvinc_1.0.0'.
- BAQ-APINAME-LEN: VALUE 13.
- BAQ-APIPATH: VALUE '/cscvinc/employee/{numb}'.
- BAQ-APIPATH-LEN: VALUE 24.
- BAQ-APIMETHOD: VALUE 'GET'.
- BAQ-APIMETHOD-LEN: VALUE 3.

cscvinc.properties
connectionRef=cscvincAPI

The screenshot shows the Server Config interface for the file apiRequesterHTTPS.xml. The XML code defines an endpoint connection with the following properties:

```
<zosconnect_endpointConnection id="cscvincAPI"
    host="https://dvipa.washington.ibm.com"
    port="9443"
    authenticationConfigRef="mySAFAuth"
    connectionTimeout="10s"
    receiveTimeout="4s" />
```

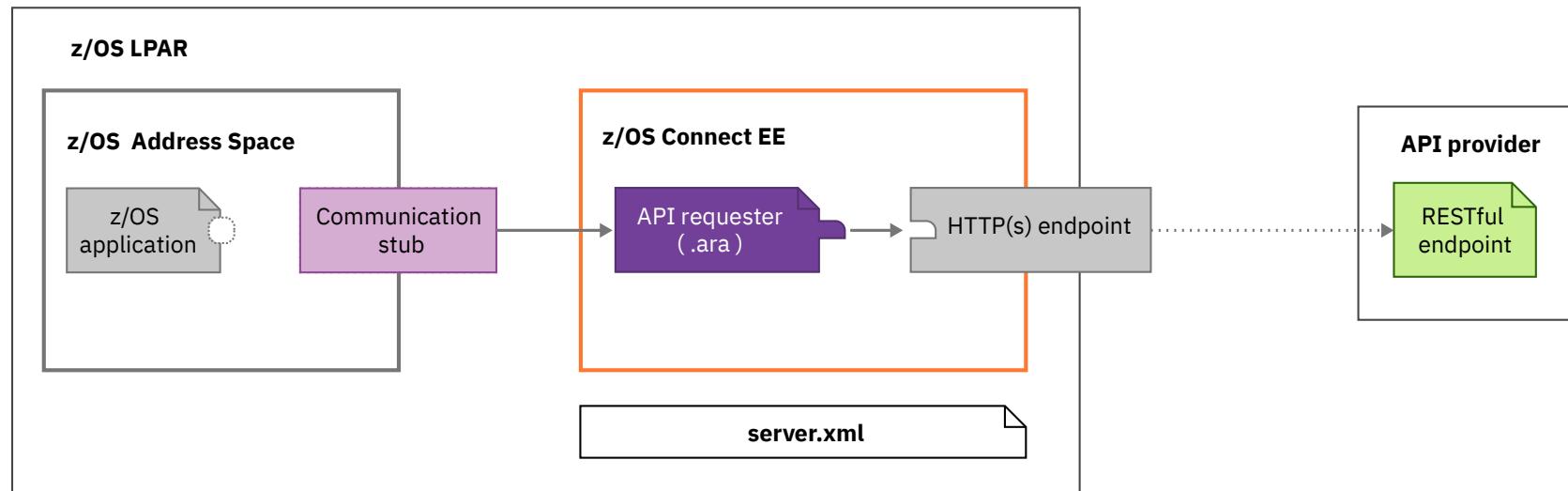
The connection is configured to use the mySAFAuth authentication configuration and to connect to https://dvipa.washington.ibm.com:9443.

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



Steps to calling an external API

Configure HTTP(S) endpoint configuration element



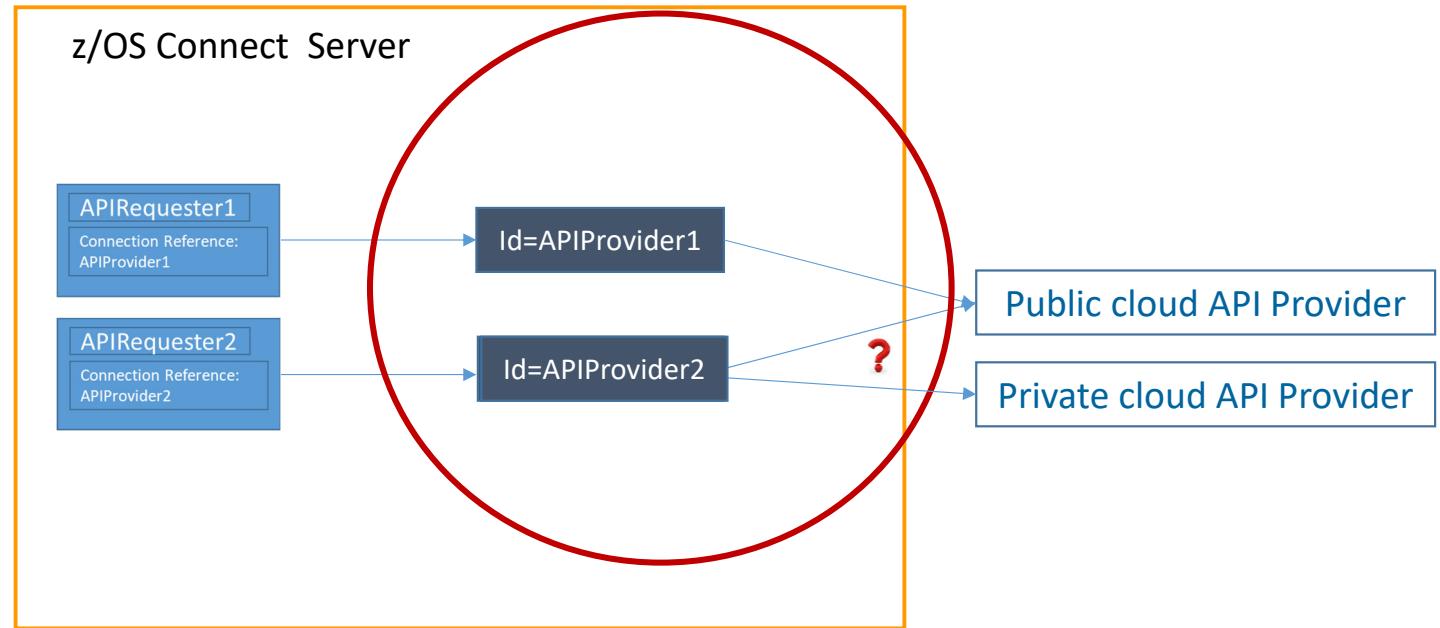
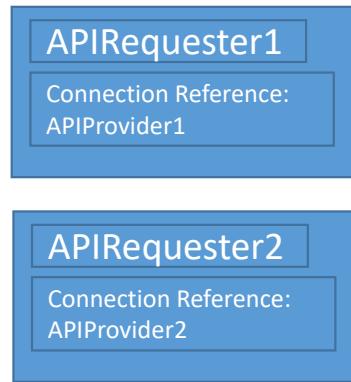
Configure the connection between z/OS Connect EE and the external API.

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



Use naming conventions for connection references

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

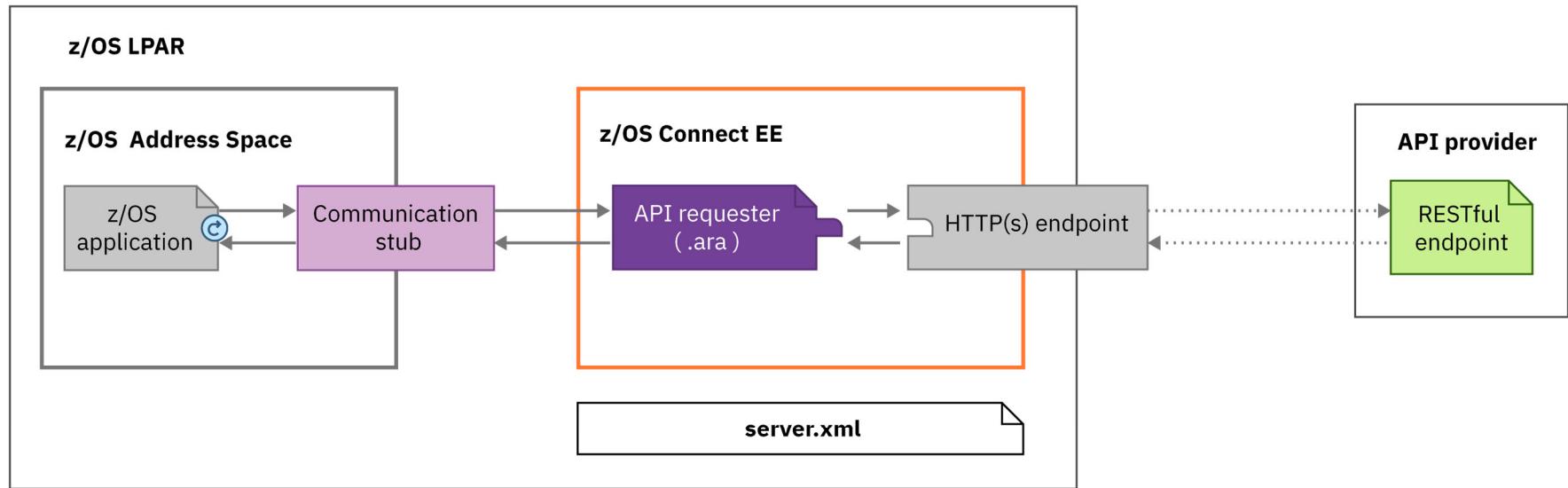


```
<zosconnect_apiRequesters>
  requireAuth="true|false"
  <apiRequester name="cscvincapi 1.0.0"
    connectionRef="APIProvider2"
  </zosconnect_apiRequesters>
```

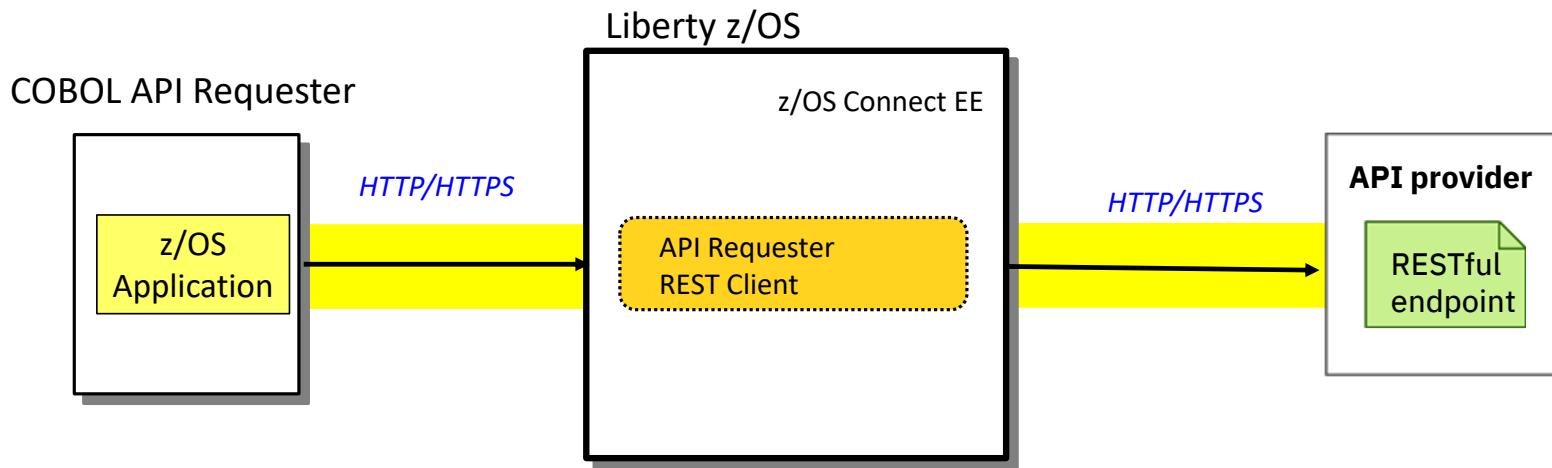


Steps to calling an external API

Done



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)



Configure connections to the z/OS API requester server

Default CICS URI MAP*

```

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

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

```

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

LE Environment Variables

```

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

```

```

mpz3
File Edit Settings View Communication Actions Window Help
BROWSE ZCEE30.SBAQCOB(BAQRINFO) Line 000000010 Col 001 080
Command ==> Scroll ==> PAGE
* (C) Copyright IBM Corp. 2017, 2021
* US Government Users Restricted Rights - Use, duplication or
* disclosure restricted by GSA ADP Schedule Contract with
* IBM Corp
*****
* This file contains the generated language structure(s) for
* Request and Response Info
*****
* BAQ-REQUEST-INFO-COMP-LEVEL permitted values
* VALUE
*   0  Base support
*   1  Added support for BAQ-OAUTH
*   2  Added support for BAQ-TOKEN (JWT)
*   3  Added support for setting z/OS Connect EE server URI
*   4  Added support for BAQ-OAUTH-EXT
*****
01 BAQ-REQUEST-INFO.
03 BAQ-REQUEST-INFO-COMP-LEVEL PIC S9(9) COMP-5 SYNC VALUE 4.
03 BAQ-REQUEST-INFO-USER
05 BAQ-OAUTH.
07 BAQ-OAUTH-USERNAME PIC X(256).
07 BAQ-OAUTH-USERNAME-LEN PIC S9(9) COMP-5 SYNC
07 BAQ-OAUTH-PASSWORD VALUE 0.
07 BAQ-OAUTH-PASSWORD-LEN PIC X(256).
PIC S9(9) COMP-5 SYNC
04/015
Connected to remote server/host mpz3 using lu/pool MPZ30044 and port 23

```



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.



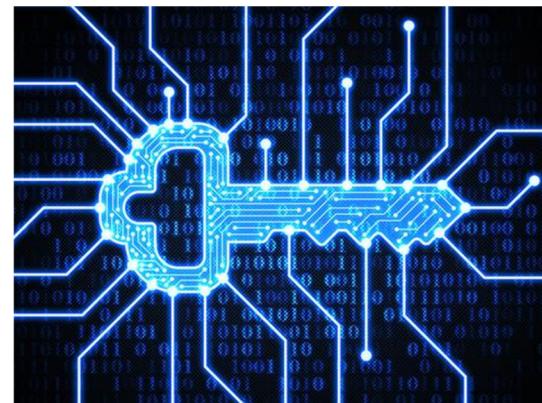
/security

How is security implemented?

General security terms or considerations

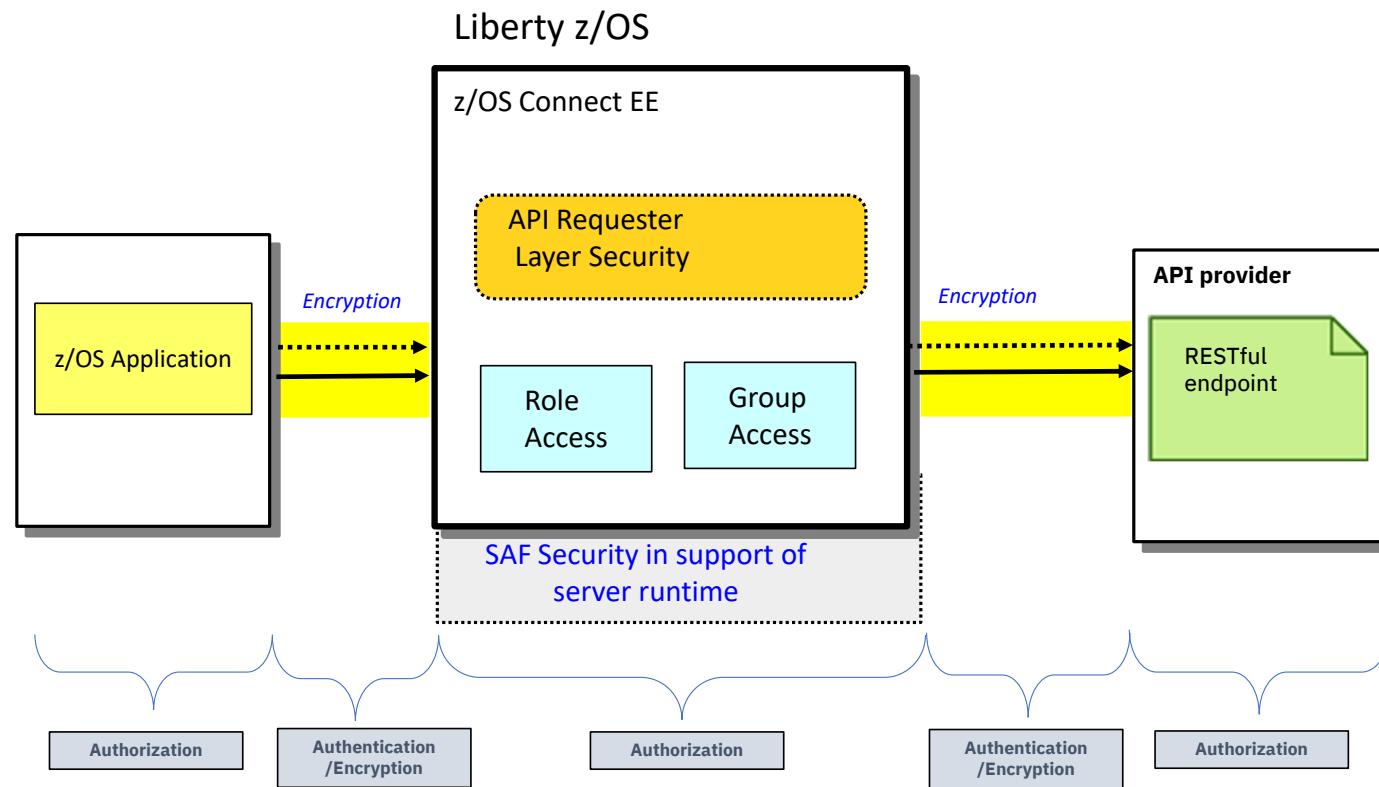
Security involves

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



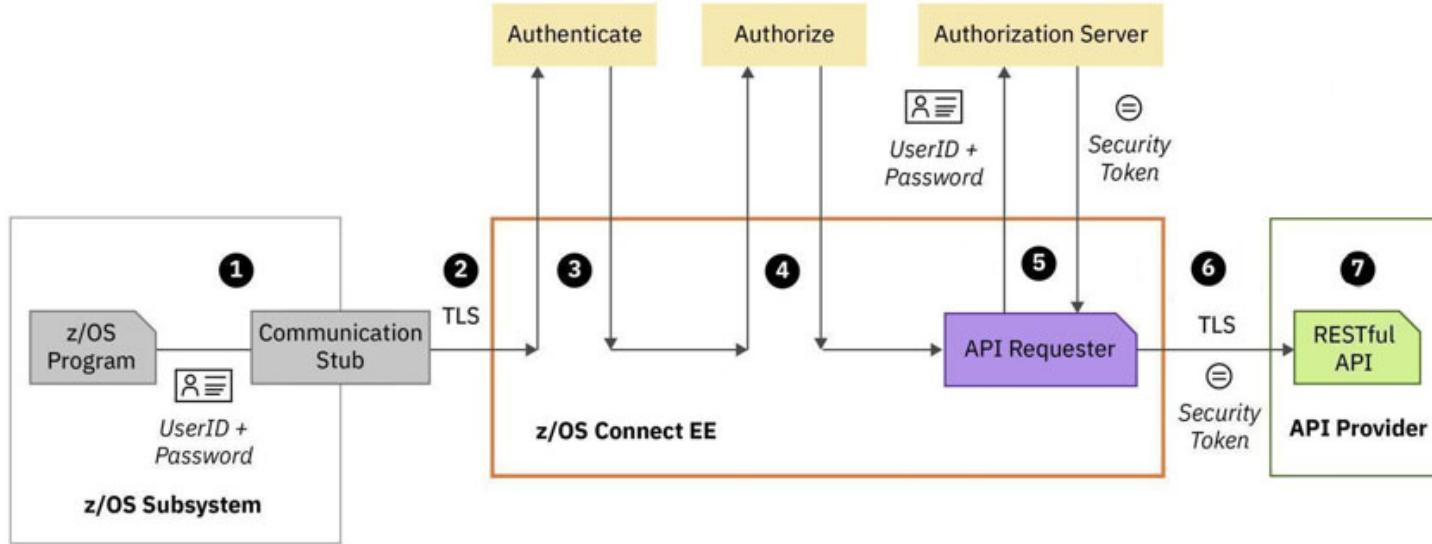


Outbound Authentication versus Authorization (OpenAPI 2)





Typical z/OS Connect EE API Requester security flow



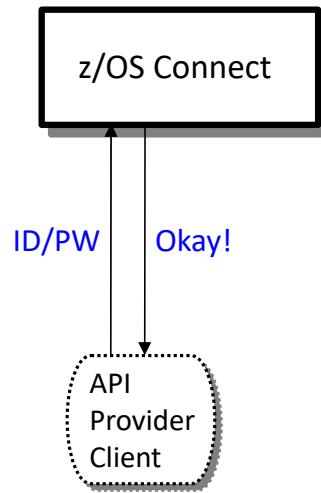
1. A user ID and password can be used for basic authentication by the z/OS Connect EE server
2. Connection between the CICS, IMS, or z/OS application and the z/OS Connect EE server can use TLS
3. Authenticate the CICS, IMS, or z/OS application.
4. Authorize the authenticated user ID to connect to z/OS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



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

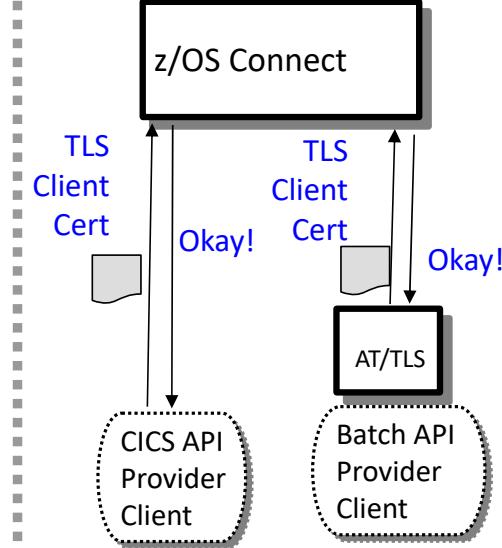
Two options for providing credentials for authentication

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 – COBOL API Requester

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

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

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

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

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

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



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

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



Tech/Tip: Generating PassTickets on z/OS

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

```
77 COMM-STUB-PGM-NAME          PIC X(8) VALUE 'BAQCSTUB'.
77 PTKT-STUB-PGM-NAME         PIC X(8) VALUE 'ATSPKTTC'.
*-----
***** L I N K A G E   S E C T I O N *****
LINKAGE SECTION.
***** P R O C E D U R E S *****
PROCEDURE DIVISION using PARM-BUFFER.

*-----*
MAINLINE SECTION.

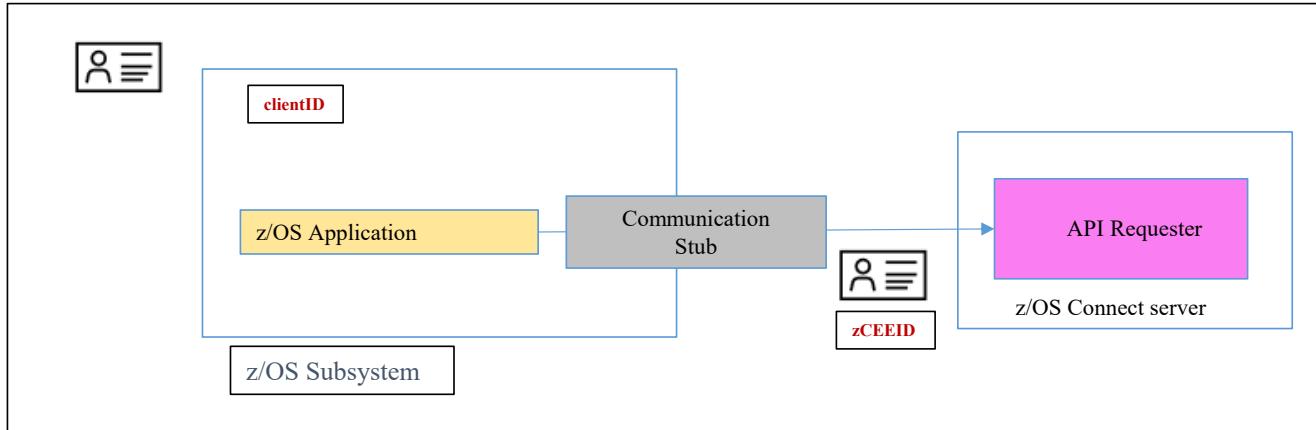
*-----*
* Common code *
*-----*
* initialize working storage variables
  INITIALIZE GET-REQUEST.
  INITIALIZE GET-RESPONSE.
  CALL PTKT-STUB-PGM-NAME.
```

JOHNSON. PASSTCKT. SOURCE (ATSPKTTC)

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



API Requester - basic authentication and identity assertion



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 USERID

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

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

```

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

<zosconnect_apiRequesters idAssertion="OFF">

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

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

</zosconnect_apiRequesters>

```



Identity assertion requires setting a program control extended attribute

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

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

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

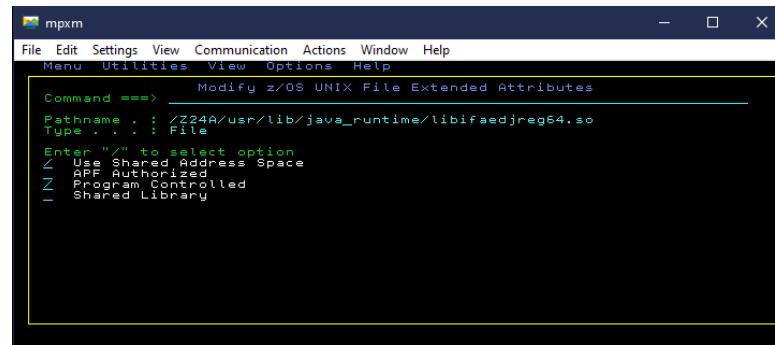
- *Define a SURROGAT profile for the asserted identity and permit access to connection identity*

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

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

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

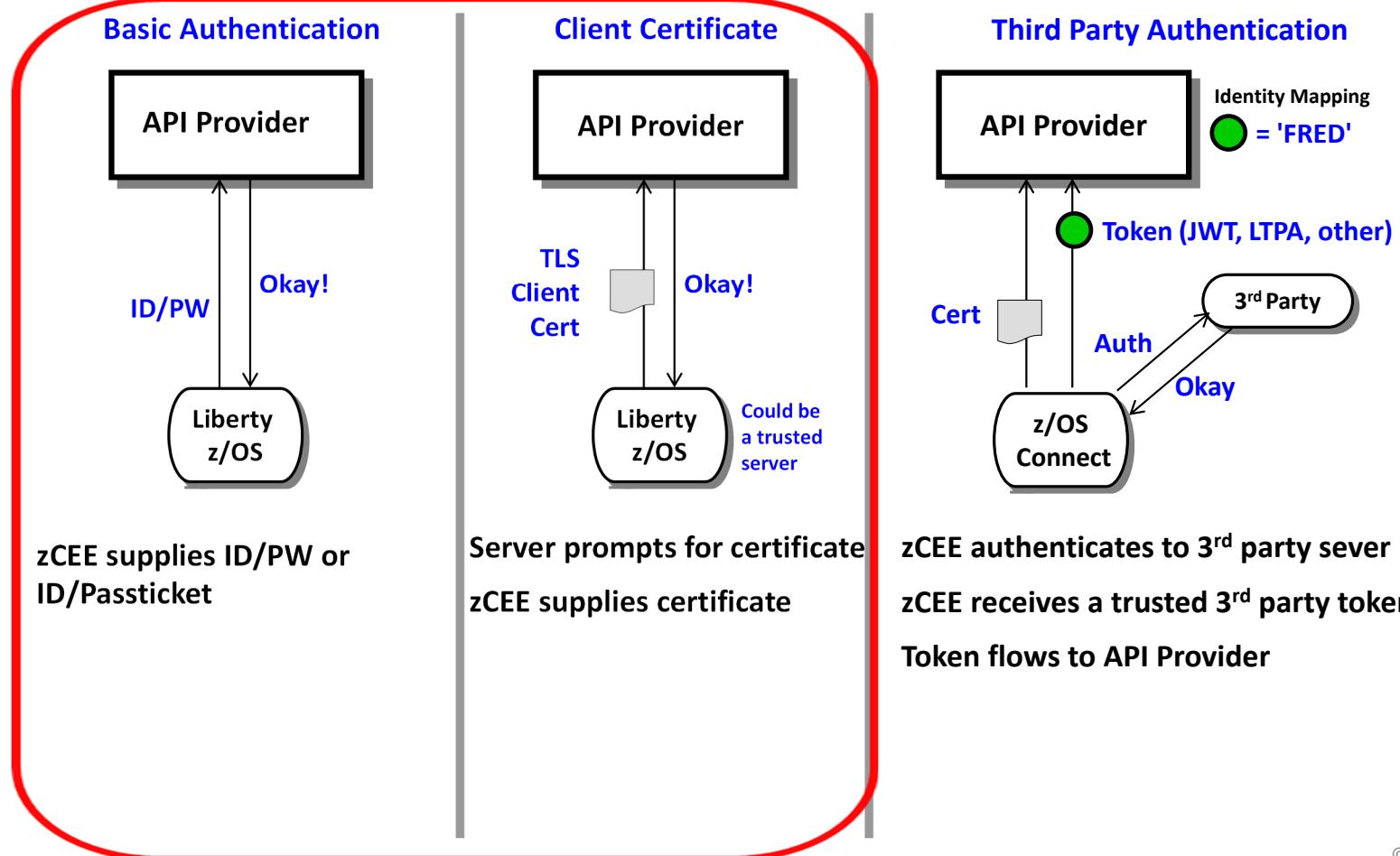
```
su  
cd /usr/lib/java_runtime  
extattr +p libifaedjreg64.so
```





API Requester - API Provider Authentication

Several different ways this can be accomplished:





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
CICS RELEASE = 0710
OVERTYPE TO MODIFY
CEDA ALTER Urimap( BAQURIMP )
Urimap      : BAQURIMP
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     ==> 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
MBI C
Connected to remote server/host wg31 using lu/pool TCP00133 and port 23
```



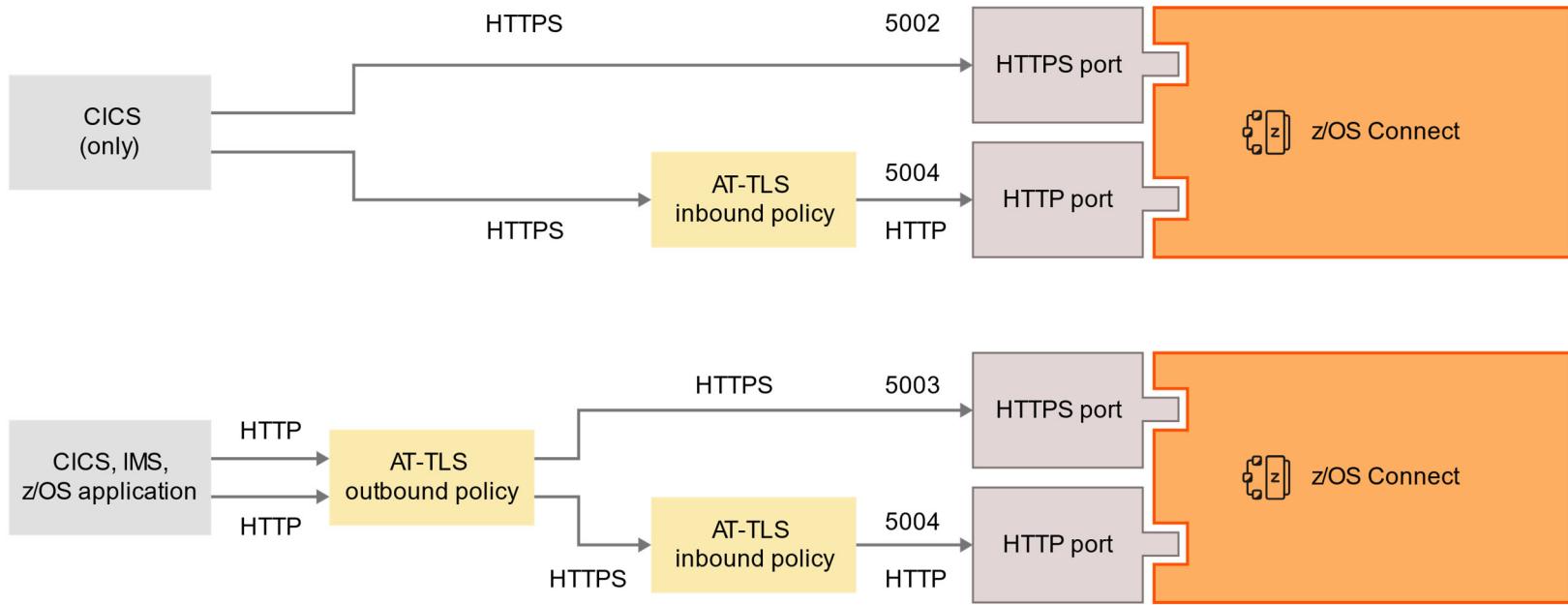
```
WG31
File Edit Settings View Communication Actions Window Help
CICS RELEASE = 0710
OVERTYPE TO MODIFY
CEDA ALTER Urimap( BAQURIMP )
Urimap      : BAQURIMP
Group       : SYSGRP
Description ==> URIMAP for z/OS Connect EE server
Status      ==> Enabled   Enabled | Disabled
Usage       ==> Client    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
MBI 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



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



Basic authentication with HTTP protocol

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

TLS with HTTPS protocol

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

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



Sample JCL - Executing the Liberty *securityUtility* command

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

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

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

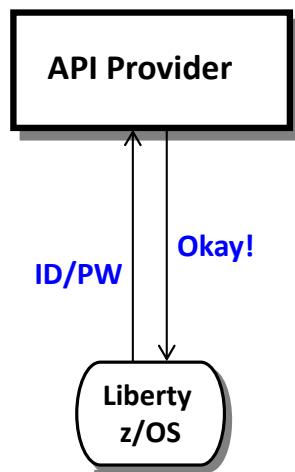
```
wlp.password.encryption.key=myEncryptionKey
```

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



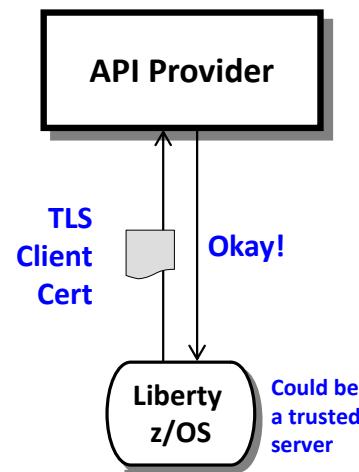
Several different ways this can be accomplished:

Basic Authentication



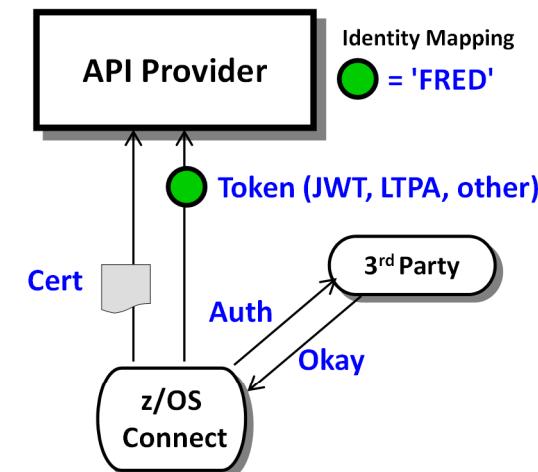
zCEE supplies ID/PW or
ID/Passticket

Client Certificate



Server prompts for certificate
zCEE supplies certificate

Third Party Authentication



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



Third Party Authentication Examples

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

Left Screenshot: UPS Sign Up

This screenshot shows the UPS Sign Up page. At the top, there's a banner stating "UPS is open for business: Service impacts related to Coronavirus ...More". Below the banner, the UPS logo is displayed. A "Sign Up / Log in" link and a "Search or Track" input field are visible. The main section is titled "Sign Up" and includes a link for users who already have an ID. It provides several social media and service provider options for sign-up: Google, Facebook, Amazon, Apple, and Twitter. Below these, fields for "Name *", "Email *", "User ID *", "Password *", and "Phone" are provided, along with a "Show" link for the password field.

Right Screenshot: myNCDMV Log In

This screenshot shows the myNCDMV Log In page. The background features a scenic view of autumn foliage. The page has "Log In" and "Sign Up" tabs at the top. The "Log In" tab is active. The log-in form requires "Email Address" and "Password", with a "Remember Me" checkbox. Below the form are links for "Forgot Password" and "Continue with Apple", "Facebook", and "Google". A "Continue as Guest" link is also present. A notice for public computer users is displayed at the bottom, followed by the "powered by payit" logo.



Open security standards

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

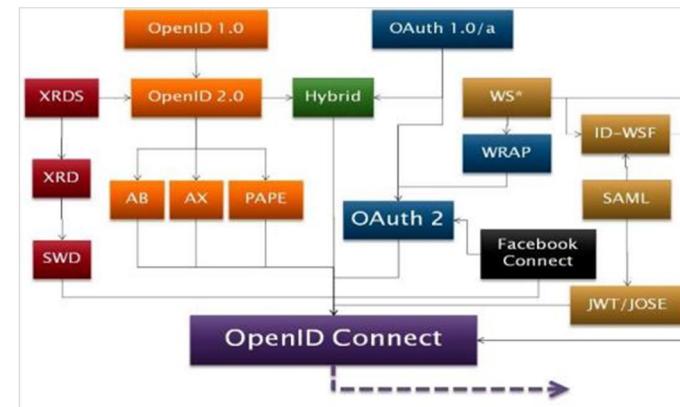
See the YouTube videos:

OAuth 2.0 and OpenID Connect (in plain English)

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

OpenID Connect on Liberty

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





What is a JWT (JSON Web Token) ?

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

The screenshot shows the jwt.io debugger interface. On the left, under 'Encoded', is a long string of characters: eyJraWQi0iI0cWpYLWJrWE9Vd19GX...vT_Ez0fD-vtHPxav4cBrGNRDR4ubRo-. In the center, under 'Decoded', are two JSON objects. The 'HEADER' object contains: { "kid": "4qjX-bkX0Uw_F_uccjRMkB9ivMjXSQwj0RrkYRJq8DM", "alg": "RS256" }. The 'PAYLOAD' object contains: { "sub": "Fred", "token_type": "Bearer", "scope": ["openid", "profile", "email"], "azp": "rpSsl", "iss": "https://wg31.washington.ibm.com:26213/oidc/endpoint/0P", "aud": "myZcee", "exp": 1604333158, "iat": 16043330858, "realmName": "zCEERealm", "uniqueSecurityName": "Fred" }. A red oval highlights the 'exp' field in the payload, which is annotated with 'Mon Nov 02 2020 11:05:58 GMT-0500 (Eastern Standard Time)'.

Values derived from the OAUTH configuration:

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

<https://jwt.io>

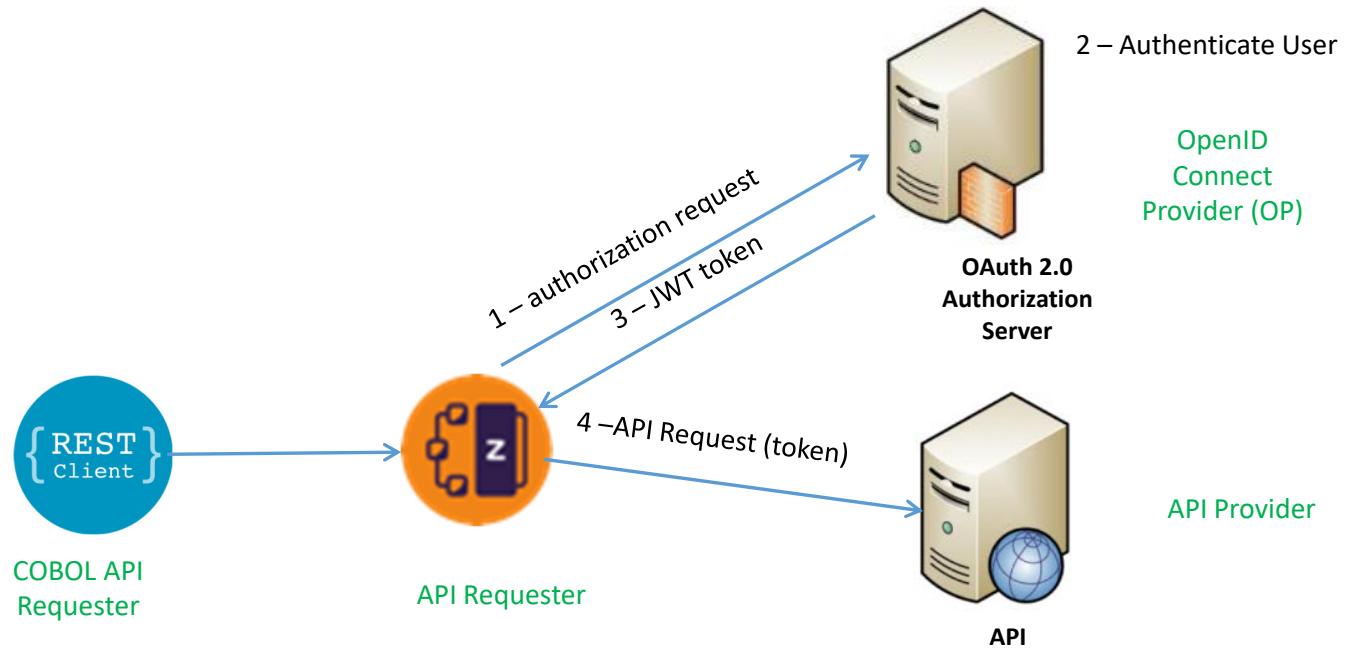
z/OS Connect API Requester - Token Support



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

1. Use the OAuth 2.0 support when the request is part of an OAuth 2.0 flow. With OAUTH 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.

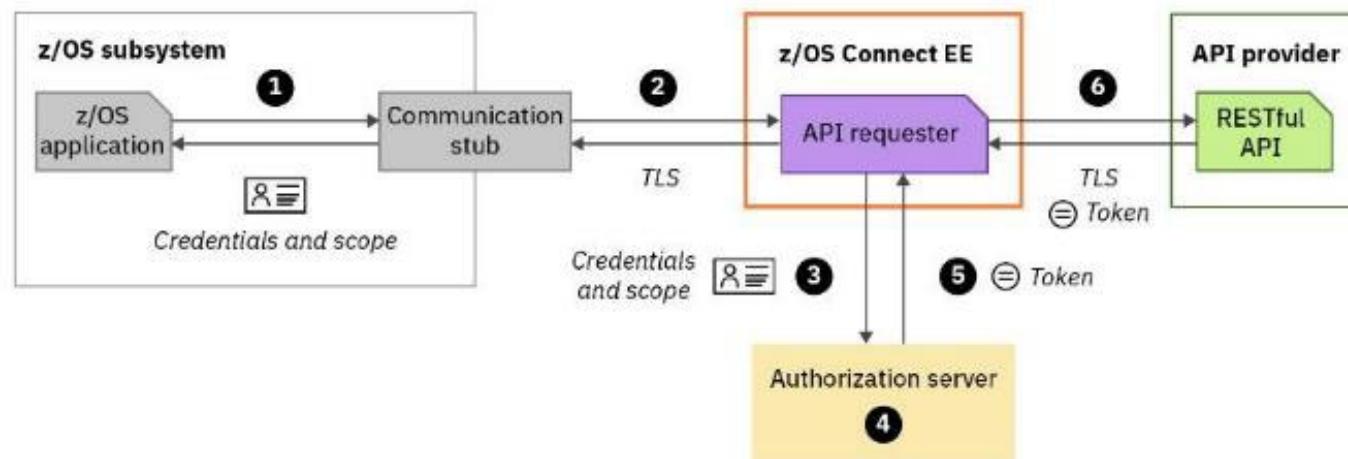
z/OS Connect OAuth Flow for API requester



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 application, 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 user of 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=996OixHze0>

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-DAUTH.
07 BAQ-DAUTH-USERNAME PIC X(256).
07 BAQ-DAUTH-USERNAME-LEN PIC S9(9) COMP-5 SYNC
    VALUE 0.
07 BAQ-DAUTH-PASSWORD PIC X(256).
07 BAQ-DAUTH-PASSWORD-LEN PIC S9(9) COMP-5 SYNC
    VALUE A.
07 BAQ-DAUTH-CLIENTID PIC X(256).
07 BAQ-DAUTH-CLIENTID-LEN PIC S9(9) COMP-5 SYNC
    VALUE 0.
07 BAQ-DAUTH-CLIENT-SECRET PIC X(256).
07 BAQ-DAUTH-CLIENT-SECRET-LEN PIC S9(9) COMP-5 SYNC
    VALUE A.
07 BAQ-DAUTH-SCOPE-PTR USAGE POINTER.
07 BAQ-DAUTH-SCOPE-LEN PIC S9(9) COMP-5 SYNC
    VALUE 0.
05 BAQ-AUTHTOKEN.
07 BAQ-TOKEN-USERNAME PIC X(256).
07 BAQ-TOKEN-USERNAME-LEN PIC S9(9) COMP-5 SYNC
    VALUE 0.
07 BAQ-TOKEN-PASSWORD PIC X(256).
07 BAQ-TOKEN-PASSWORD-LEN PIC S9(9) COMP-5 SYNC
    VALUE 0.
05 BAQ-ZCON-SERVER-URI PIC X(256)
    VALUE SPACES.
MA A 04/015
Connected to remote server/host wg31z using lu/pool TCP00145
```

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



Sample program and JCL

COBOL Application

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

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

Execution JCL

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



Tech/Tip: Accessing environment variables from COBOL application

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

```
01 functionCode PIC 9(9) BINARY.  
01 envVariableNameLength PIC 9(9) BINARY.  
01 envVariableName PIC X(255).  
01 valueLength PIC 9(9) BINARY.  
01 valuePointer POINTER.  
01 ws-length PIC 9(3).  
  
01 feedbackCode.  
02 CONDITION-TOKEN-VALUE.  
COPY CEEIGZCT.  
    03 CASE-1-CONDITION-ID.  
        04 SEVERITY      PIC S9(4) BINARY.  
        04 MSG-NO        PIC S9(4) BINARY.  
    03 CASE-SEV-CTL   PIC X.  
    03 FACILITY-ID   PIC XXX.  
    02 I-S-INFO       PIC S9(9) BINARY.  
01 VAL          PIC X(255).
```

```
CALL-CEEENV.  
    MOVE 1 TO functionCode.  
    MOVE ZERO TO ws-length.  
    INSPECT FUNCTION REVERSE (envVariableName)  
        TALLYING ws-length FOR LEADING SPACES.  
    COMPUTE envVariableNameLength =  
        LENGTH OF envVariableName - ws-length.  
    MOVE " " TO VAL.  
    MOVE 0 TO valueLength.  
    CALL "CEEENV" USING functionCode,  
        envVariableNameLength,  
        envVariableName,  
        valueLength,  
        valuePointer,  
        feedbackCode.  
  
    IF valueLength NOT = 0 THEN  
        SET ADDRESS OF VAR TO valuePointer .  
  
CALL-CEEENV-END.,
```

Configuring OAuth support – z/OS Connect API Requester



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

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

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

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

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

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

² These credentials can be specified by the application

Security Scenarios



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

distuser1 is mapped to RACF identity USER1 who has full access

```
BAQ-OAUTH-USERNAME: distuserx  
BAQ-OAUTH-PASSWORD: pwd  
Error code: 00000500
```

```
Error msg:{ "errorMessage": "BAQR1092E: Authentication or authorization failed for the z/OS Connect EE server." }
```

distuserx is unknown by the OAuth Provider

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

auser is not mapped to a valid RACF identity

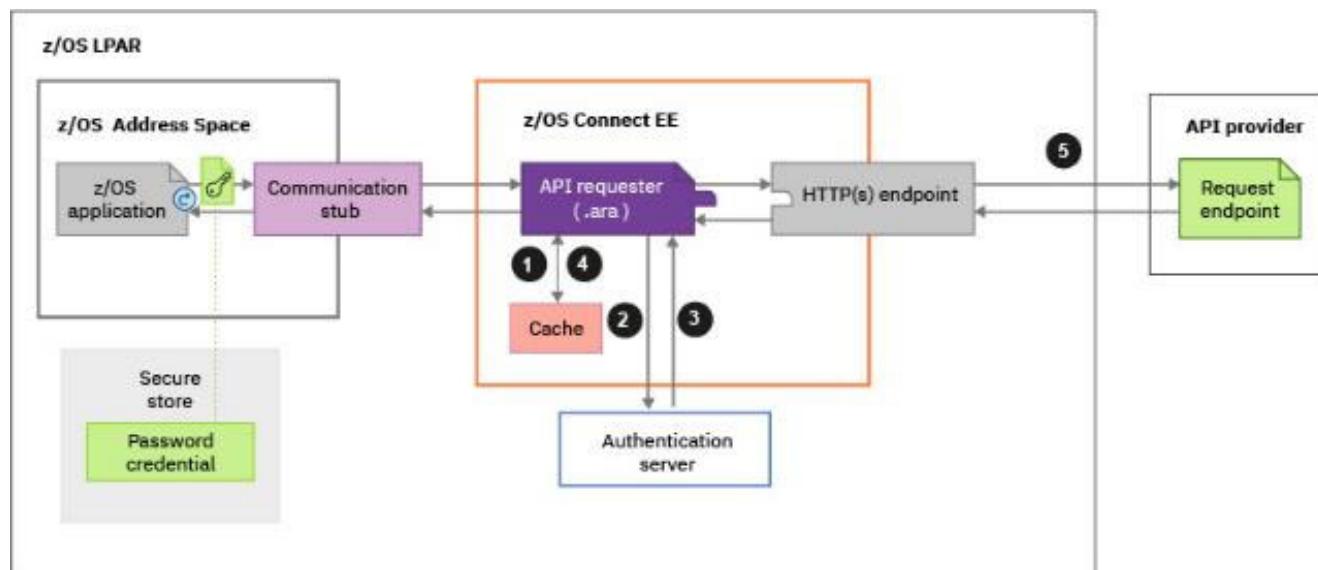
```
BAQ-OAUTH-USERNAME: distuser2  
BAQ-OAUTH-PASSWORD: pwd  
Error code: 0000000403  
Error msg:{ "errorMessage": "BAQR1144E: Authentication or authorization failed for the z/OS Connect EE server." }  
Syslog:  
ICH408I USER(USER2 ) GROUP(SYS1 ) NAME(WORKSHOP USER2  
ATSZDFLT.zos.connect.access.roles.zosConnectAccess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

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

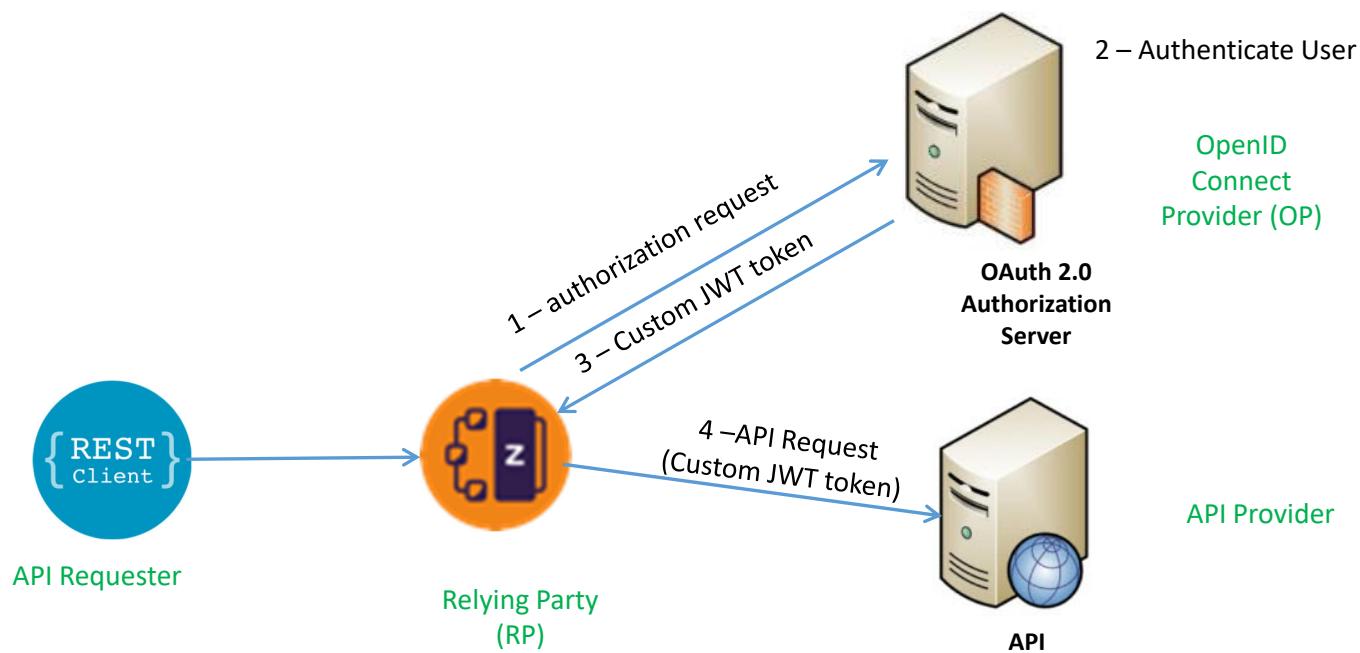


Calling an API with using a JWT custom flow

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



z/OS Connect OAuth Custom Flow





API Requester – JWT Custom flow

BAQRINFO copy book

```
BROWSE ZCEE30.SBAQC0B(BAQRINFO) Line 0000000028 Col 001 080
Command ==> Scroll ==> PAGE
01 BAQ-REQUEST-INFO.
  03 BAQ-REQUEST-INFO-COMP-LEVEL PIC S9(9) COMP-5 SYNC VALUE 4.
  03 BAQ-REQUEST-INFO-USER.
    05 BAQ-DAUTH.
      07 BAQ-DAUTH-USERNAME PIC X(256).
      07 BAQ-DAUTH-USERNAME-LEN PIC S9(9) COMP-5 SYNC
        VALUE 0.
      07 BAQ-DAUTH-PASSWORD PIC X(256).
      07 BAQ-DAUTH-PASSWORD-LEN PIC S9(9) COMP-5 SYNC
        VALUE 0.
      07 BAQ-DAUTH-CLIENTID PIC X(256).
      07 BAQ-DAUTH-CLIENTID-LEN PIC S9(9) COMP-5 SYNC
        VALUE 0.
      07 BAQ-DAUTH-CLIENT-SECRET PIC X(256).
      07 BAQ-DAUTH-CLIENT-SECRET-LEN PIC S9(9) COMP-5 SYNC
        VALUE 0.
      07 BAQ-DAUTH-SCOPE-PTR USAGE POINTER.
      07 BAQ-DAUTH-SCOPE-LEN PIC S9(9) COMP-5 SYNC
        VALUE 0.
  05 BAQ-AUTHTOKEN.
    07 BAQ-TOKEN-USERNAME PIC X(256).
    07 BAQ-TOKEN-USERNAME-LEN PIC S9(9) COMP-5 SYNC
      VALUE 0.
    07 BAQ-TOKEN-PASSWORD PIC X(256).
    07 BAQ-TOKEN-PASSWORD-LEN PIC S9(9) COMP-5 SYNC
      VALUE 0.
  05 BAQ-ZCON-SERVER-URI PIC X(256)
    VALUE SPACES.
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.



Configuring JWT Custom flow

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

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

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

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

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

² These credentials can be specified by the application



Configuring Custom JWT flow

Request Token Example 1

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

Response Token

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

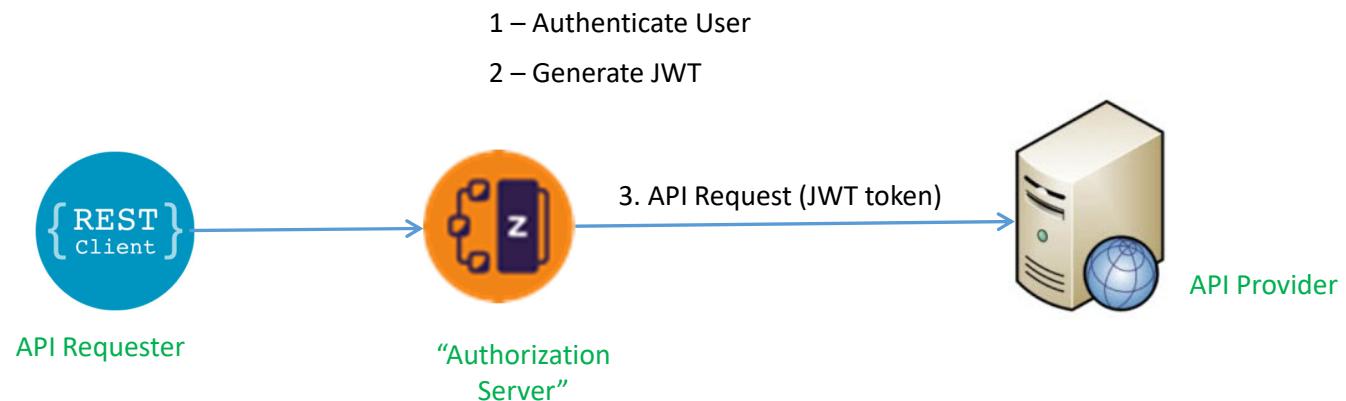
Response Token Example 2

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

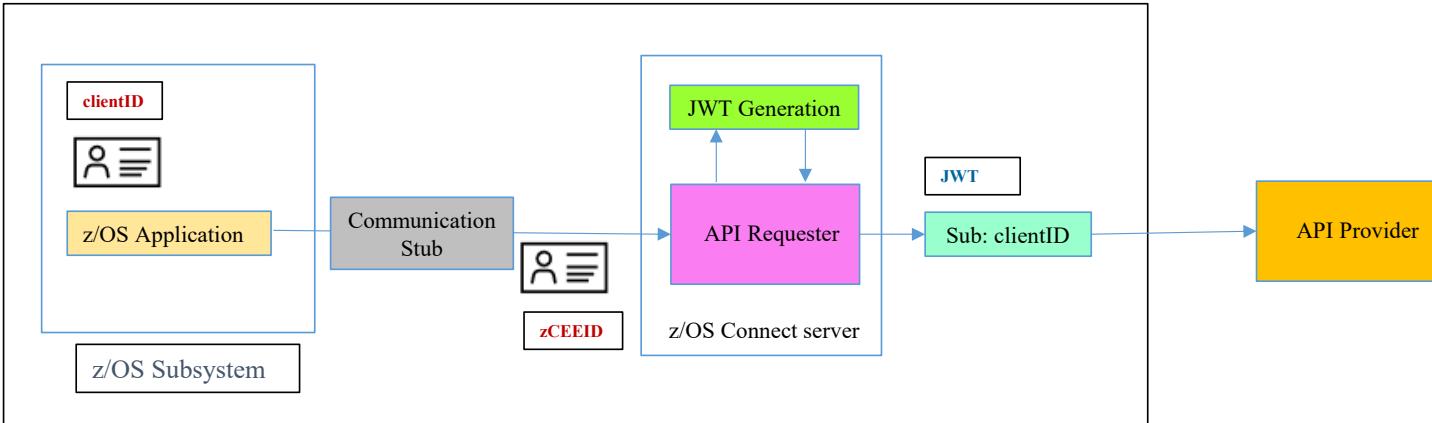
Response Token

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

z/OS Connect JWT Generation – V3.0.43



API Requester – JWT Generation



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

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

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

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



JWT generation requires setting a program control extended attribute

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

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

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

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

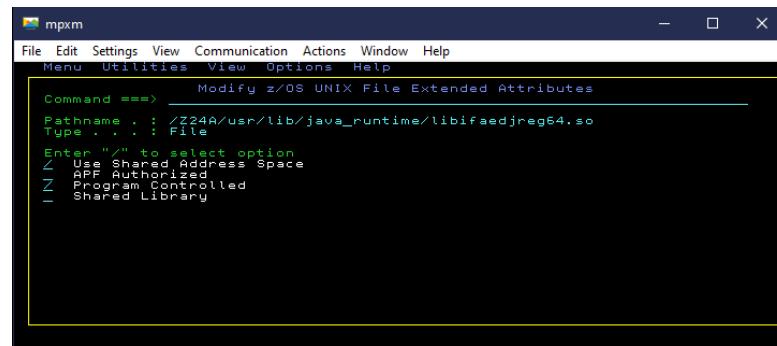
```
RDEFINE SURROGAT clientID.BAQASSRT UACC(NONE) OWNER(SYS1)  
PERMIT clientID.BAQASSRT CLASS(SURROGAT) ACCESS(READ) ID(zCEEID)
```

OR

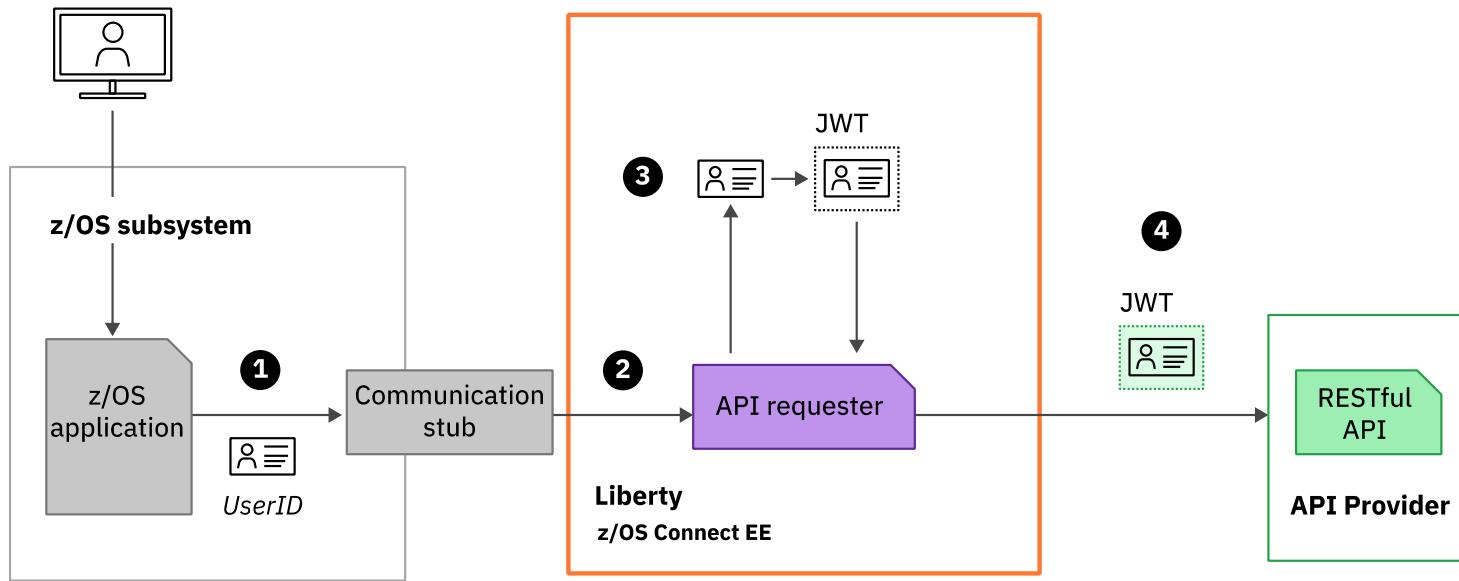
```
RDEFINE SURROGAT *.BAQASSRT UACC(NONE) OWNER(SYS1)  
PERMIT *.BAQASSRT CLASS(SURROGAT) ACCESS(READ) ID(zCEEID)  
SETROPTS RACLIST(SURROGAT) REFRESH
```

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

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



JWT Generation



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



Configuring JWT Generation support

```
<zosconnect_endpointConnection id="conn"
    host="http://api.server.com" port="8080"
    authenticationConfigRef="jwtConfig" />

<zosconnect_authTokenLocal id="jwtConfig"
    tokenGeneratorRef="jwtBuilder"
    header="Authorization" >
    <claims>{ "name":"JohnSmith",
        "ID":"1234567890" }
    </claims> One or more Public claim (e.g., aud,exp,nbf,iat,jti) or
one or more private claims

<jwtBuilder id="jwtBuilder"
    scope="scope1"
    audiences="myApp1"
    jti="true"
    signatureAlgorithm="RS256"
    keyStoreRef="myKeyStore"
    keyAlias="jwtsigner"
    issuer="z/OS Connect EE Default"/>
```

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" />
```

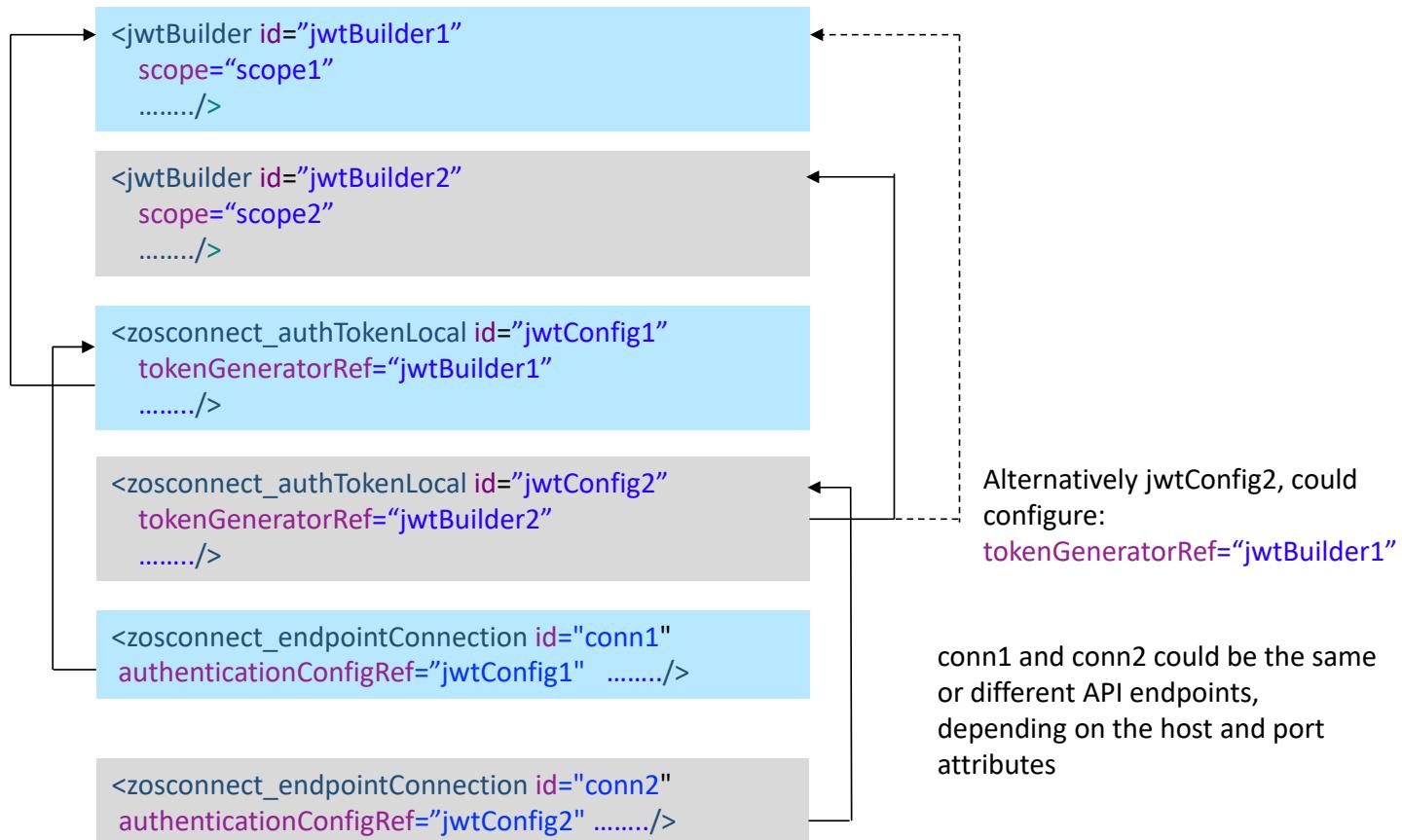
Configure the Liberty jwtBuilder element in server.xml.

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.

Using different claims for different API endpoints



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



The image displays three GitHub repository pages side-by-side, each featuring a red oval highlighting specific file uploads.

Left Repository: <https://github.com/ibm-wsc/zCONNEE-Wildfire-Workshop>

- Commits:** emitchj Add files via upload (8e503b5)
- Files:**
 - AdminSecurity Delete ZC2OMVS2.jcl
 - OpenAPI2 Delete Developing
 - cobol Add files via upload
 - xml Add files via upload
 - README.md Update README.md
 - ZCADMIN - zOS Connect Administrat... Add files via upload
 - ZCESEC - zOS Connect Security.pdf Add files via upload
 - ZCINTRO - Introduction to zOS Conn... Add files via upload
 - ZCREQUEST - Introduction to zOS Co... Add files via upload** (highlighted by a red oval)
 - zOS Connect EE V3 Advanced Topics ... Add files via upload
 - zOS Connect EE V3 Getting Started.pdf Add files via upload
- README.md**
- Notes:** This repository contains material from the z/OS Connect EE Wildfire workshops run by the IBM Center. It is should be referenced frequently for updates to the presentations, exercises, sam...

Middle Repository: <https://github.com/ibm-wsc/zCONNEE-Wildfire-Workshop>

- Commits:** emitchj Delete Developing (d880029 on Apr 23)
- Files:**
 - Developing CICS API Requester Applications.pdf Add files via upload (2 months ago)
 - Developing IMS API Requester Applications.pdf Add files via upload (2 months ago)
 - Developing MVS Batch API Requester Applications.pdf Add files via upload (2 months ago)

Bottom Repository: <https://github.com/ibm-wsc/zCONNEE-Wildfire-Workshop>

- Commits:** emitchj Add files via upload (428fc6c 5 days ago)
- Files:**
 - 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)

- **Contact your IBM representative to schedule access to these exercises**



Thank you for listening and your questions.

And thank you for completing the Medallia survey

mitchj@us.ibm.com

© 2018, 2022 IBM Corporation
Slide 92