

# **zOSSEC1 – IBM z/OS Connect EE Security**

A dive into Liberty and z/OS Connect Security

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



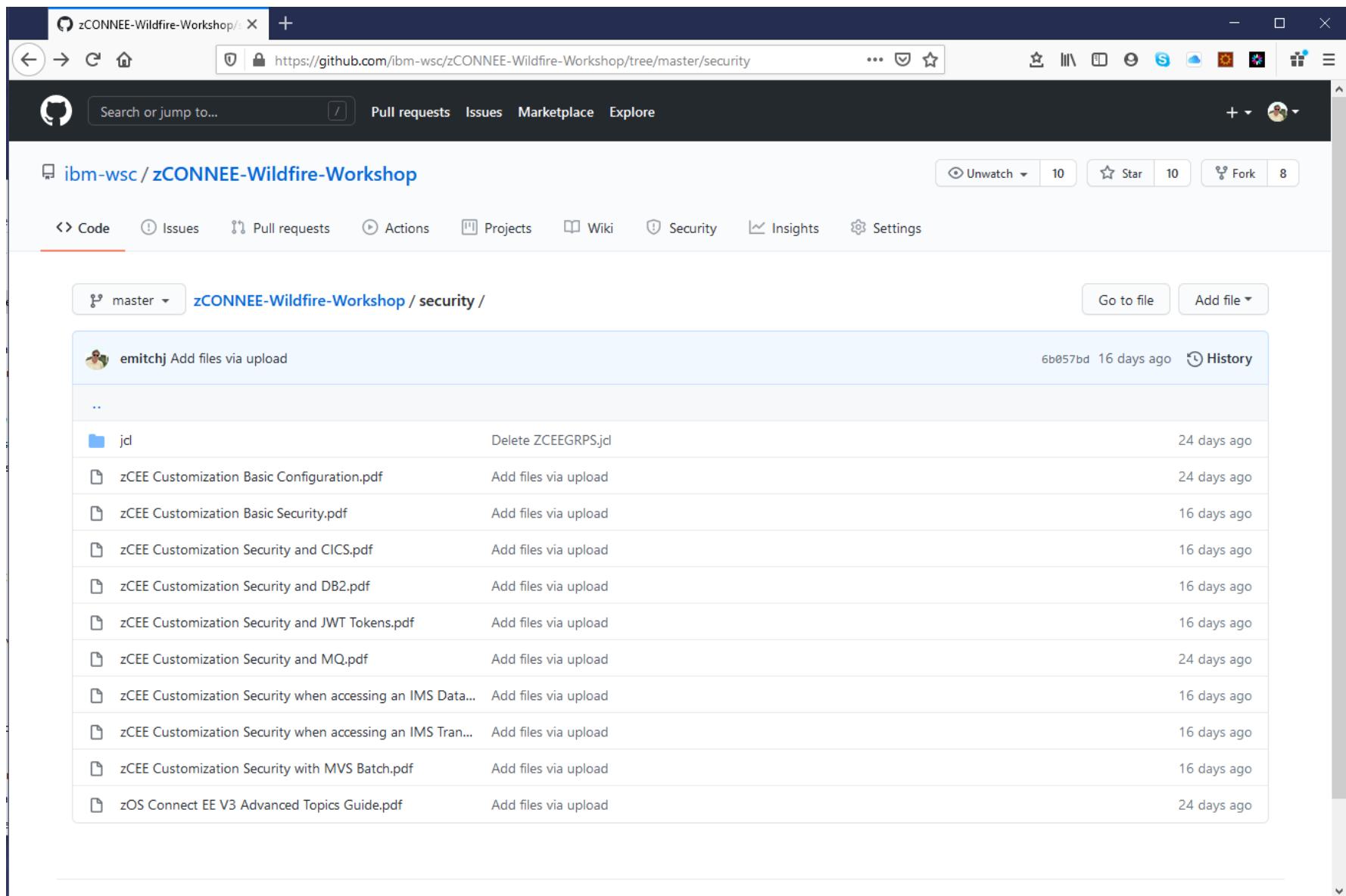
## Topics

- OMVS, Liberty, and RACF Security Review
- z/OS Connect Security Overview
- Authentication
  - Basic Authentication
  - Mutual Authentication using TLS
  - Third Party Tokens
- Encryption and Message Integrity using TLS
- Authorization
- Propagating identities to z/OS subsystems
- z/OS Connect API Requester and third-party tokens

# Disclaimer

- The information in this presentation was derived from various product Knowledge Centers (KC).
- 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 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.
- A Liberty  or z/OS Connect  icon will appear on slides where the information is specific to these products. Don't hesitate to ask questions as to why the icon does or does not appear on certain slides.
- The examples, tips, etc. present in this material are based on firsthand experiences and are not necessarily sanctioned by z/OS Connect development.

# Detailed examples of topics covered today are available

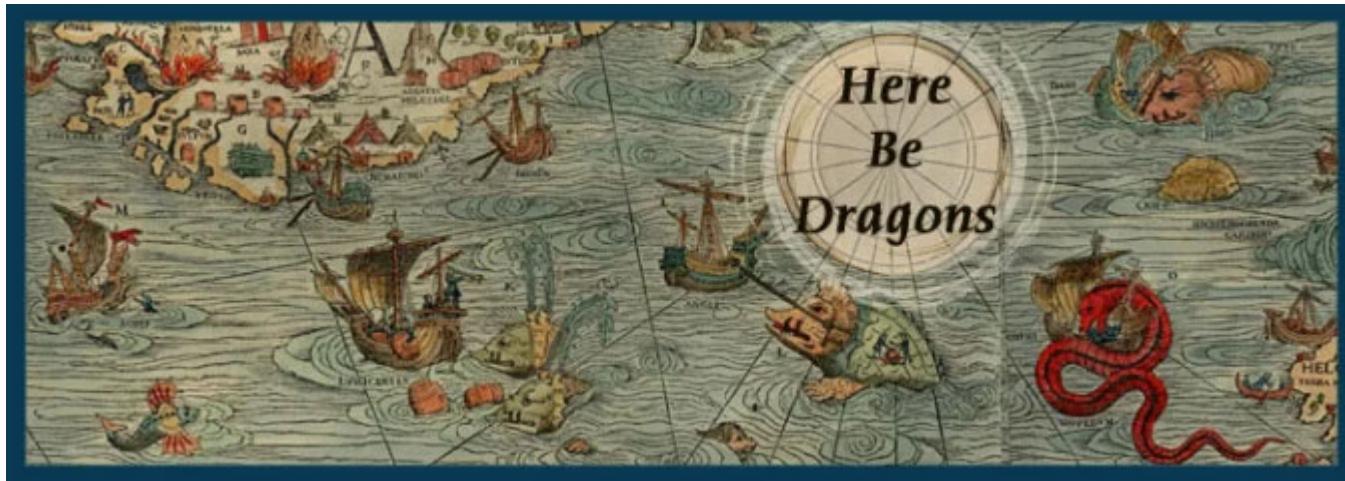


The screenshot shows a GitHub repository page for 'zCONNEE-Wildfire-Workshop'. The URL in the address bar is <https://github.com/ibm-wsc/zCONNEE-Wildfire-Workshop/tree/master/security>. The repository name is 'ibm-wsc / zCONNEE-Wildfire-Workshop'. The 'Code' tab is selected. In the 'security' directory, there are several files and a folder named 'jcl'. The files are:

File	Description	Last Modified
Delete ZCEEGRPS.jcl	Delete ZCEEGRPS.jcl	24 days ago
zCEE Customization Basic Configuration.pdf	Add files via upload	24 days ago
zCEE Customization Basic Security.pdf	Add files via upload	16 days ago
zCEE Customization Security and CICS.pdf	Add files via upload	16 days ago
zCEE Customization Security and DB2.pdf	Add files via upload	16 days ago
zCEE Customization Security and JWT Tokens.pdf	Add files via upload	16 days ago
zCEE Customization Security and MQ.pdf	Add files via upload	24 days ago
zCEE Customization Security when accessing an IMS Data...	Add files via upload	16 days ago
zCEE Customization Security when accessing an IMS Tran...	Add files via upload	16 days ago
zCEE Customization Security with MVS Batch.pdf	Add files via upload	16 days ago
zOS Connect EE V3 Advanced Topics Guide.pdf	Add files via upload	24 days ago

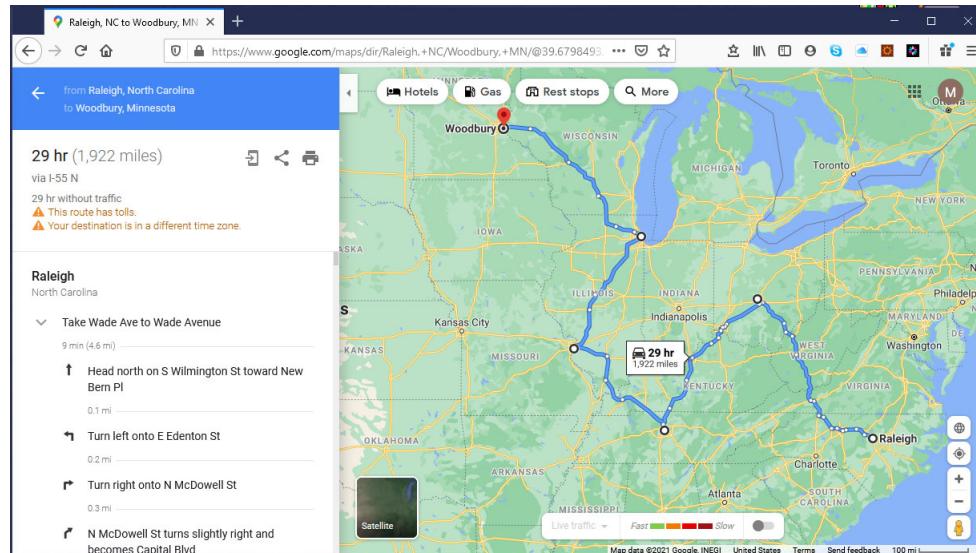
# Map of the journey to security

Sometimes it seems like implementing security is starting with a map like this:



"Here be dragons" (hic sunt dracones in Latin) means dangerous or unexplored territories.

When we would really like to have is a Google map with step-by-step directions:



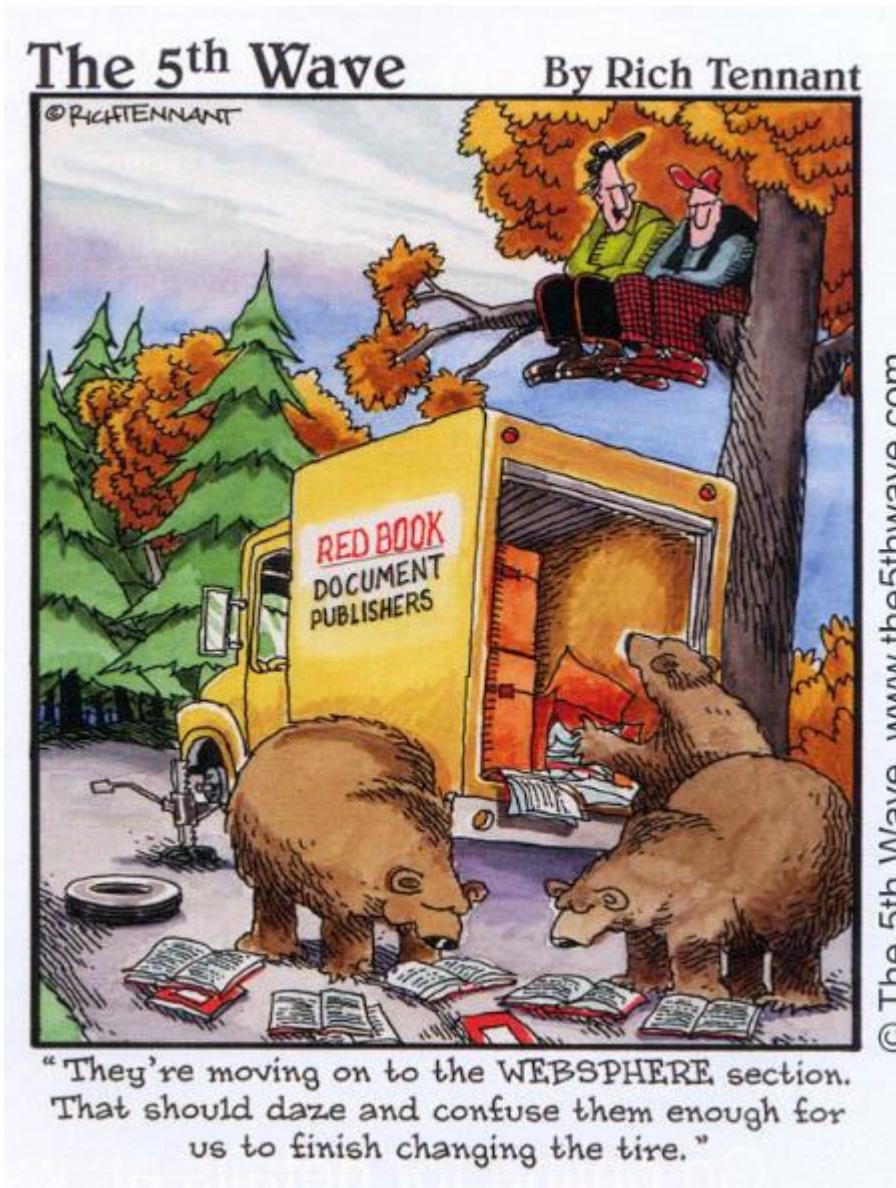
But when securing Liberty and z/OS Connect, there is probably not a direct route to your destination. There will be a series waypoints or intermediate steps on the journey which must be addressed.

# We need to understand the challenges

- Providing secure access between middleware components that use disparate security technologies e.g., registries like LDAP, SAF, TLS etc. in order to propagate security credentials from a client all the way to the targeted resource.
  - This is a driver for implementing open security models like OAuth and OpenID Connect and standard tokens like JWT
- Integrating security involves different products including z/OS Connect, WebSphere Liberty Profile on z/OS with CICS, IMS, Db2, MQ,... probably for the first time in your environment.
  - Security for of these components are all documented in different places
- Considering that security is often at odds with **performance**, the more secure techniques often mean more processing overhead, especially if not configured optimally
  - Remember security is probably not a choice but a requirement.

 *A single simple step-by-step linear map is not always possible, the best approach may be to build a solution using components, one step at a time based on the waypoints involved and the ultimate goal. Providing security by understanding these waypoints and the corresponding options is the focus of this presentation.*

# With apologies to Rich Tennant



During today's presentation  
don't hesitate to ask  
questions, or in the future, to  
reach out to your local Tech  
Sales contact to ask for  
clarification or assistance.

**Let's begin by reviewing some of the basic  
OMVS, Liberty and RACF  
security/configuration details and options**

# OMVS security - Unix file permissions

## Owner

	Read	Write	Execute
Bit	1	1	1
Base-2 Value	[4]	[2]	[1]
↓	↓	↓	↓
4 + 2 + 1 =			

**7**

The owner has **READ, WRITE and EXECUTE**



The **owner** of the file or directory

chmod **-R**\* u+rwx zceesrv1

## Group

	Read	Write	Execute
Bit	1	0	1
Base-2 Value	[4]	[2]	[1]
↓	↓	↓	↓
4 + 0 + 1 =			

**5**

The group has **READ and EXECUTE, but not WRITE**



IDs that are part of the **group** for the file or directory

chmod g+rwx server.xml

## Other

	Read	Write	Execute
Bit	0	0	0
Base-2 Value	[4]	[2]	[1]
↓	↓	↓	↓
0 + 0 + 0 =			

**0**

Others have nothing



IDs that are not the owner and not part of the group; that is, **other**

chmod **-R**\* o+rx resources  
chmod **-R**\* o-w resources/security

# Default server configuration directories and files



ID=**LIBSERV**  
Group=**LIBGRP**

```
export JAVA_HOME=<path_to_64_bit_Java>
export WLP_USER_DIR=/var/zosconnect
./server create zceesrv1
```

/var/zosconnect	750	LIBSERV LIBGRP
/servers	750	LIBSERV LIBGRP
/zceesrv1	750	LIBSERV LIBGRP
/logs	777	LIBSERV LIBGRP
messages.log	666	LIBSERV LIBGRP
/resources	750	LIBSERV LIBGRP
/zosconnect	750	LIBSERV LIBGRP
/apis	750	LIBSERV LIBGRP
/apiRequesters	750	LIBSERV LIBGRP
/rules	750	LIBSERV LIBGRP
/services	750	LIBSERV LIBGRP
server.xml	640	LIBSERV LIBGRP
server.env	640	LIBSERV LIBGRP
/workarea	750	LIBSERV LIBGRP

**It will create the directories and files under the <WLP\_USER\_DIR> and assign ownership based on the ID and Group that created the server**

**There are a few potential issues with this in a production setting:**

- If you have multiple people with a need to change configuration files, do you share the password of LIBSERV? (answer: **no**)
- If you have multiple people with a need to read or update configuration files, do you simply connect them to LIBGRP? (answer: **no**)

*CWWKB0121I: The server process UMASK value is set to 0000*

- sets permission bit for new files deployed using the RESTful APIs to rw-rw-rw (666 x'OR 000)

```
export WLP_USER_DIR=/var/zosconnect
cd $WLP_USER_DIR
chmod a+x $WLP_USER_DIR/servers
chmod a+x $WLP_USER_DIR/servers/zceesrvr
```

# Default server configuration directories and files



ID=**LIBSERV**  
Group=**LIBGRP**

```
export JAVA_HOME=<path_to_64_bit_Java>
export WLP_USER_DIR=/var/zosconnect
./server create zceesrv1
```

/var/zosconnect	751	LIBSERV	LIBGRP
/servers	751	LIBSERV	LIBGRP
/zceesrv1	751	LIBSERV	LIBGRP
/logs	771	LIBSERV	LIBGRP
messages.log	644	LIBSERV	LIBGRP
/resources	750	LIBSERV	LIBGRP
/zosconnect	750	LIBSERV	LIBGRP
/apis	760	LIBSERV	LIBGRP
/apiRequesters	760	LIBSERV	LIBGRP
/rules	760	LIBSERV	LIBGRP
/services	760	LIBSERV	LIBGRP
server.xml	460	LIBSERV	ADMGRP
server.env	460	LIBSERV	ADMGRP
/workarea	750	LIBSERV	LIBGRP

**It will create the directories and files under the <WLP\_USER\_DIR> and assign ownership based on the ID and Group that created the server**

**There are a few potential issues with this in a production setting:**

- If you have multiple people with a need to change configuration files, do you share the password of LIBSERV? (answer: **no**)  
**Sharing passwords is a bad practice. Better to take advantage SAF SURROGATE so permitted users can switch to the owning ID so they can make changes**
- If you have multiple people with a need to read or update configuration files, do you simply connect them to LIBGRP? (answer: **no**)  
**The owner group may be granted access to other resources (on z/OS SAF profiles notably: SERVER) and you do not want others inheriting that. Better to make the configuration group be something different from the owner group and grant READ/WRITE through that group.**

*CWWKB0121I: The server process UMASK value is set to 0000*

- sets permission bit for new files deployed using the RESTful artifacts to rw-rw-rw (666 x'OR 000)

~~Often you may be tempted to use command chmod -R 777 \*~~

**Access for Owner, Group, Others depend on UID and GID as stored with the directory or file, not the actual SAF identity or group. This has implications when moving entire filesystems from one LPAR to another using utility ADRDSSU**

# Tech/Tip: Providing access to configuration/log information



```
wg31.washington.ibm.com:9443/s... X +  
This XML file does not appear to have any style information associated with it. The document tree is shown below.  
  
<server description="new server">  
  <include location="${server.config.dir}/includes/safSecurity.xml"/>  
  <include location="${server.config.dir}/includes/tpicSSLIDProp.xml"/>  
  <include location="${server.config.dir}/includes/keyringOutbound.xml"/>  
  <include location="${server.config.dir}/includes/groupAccess.xml"/>  
  <include location="${server.config.dir}/includes/shared.xml"/>  
  <include location="${server.config.dir}/includes/oauth.xml"/>  
  <include location="${server.config.dir}/includes/adminCenter.xml"/>  
  <include location="${server.config.dir}/includes/mqClientTLS.xml"/>  
  <include location="${server.config.dir}/includes/web.xml"/>  
  <!-- Enable features -->  
  <featureManager>  
    <feature>zosconnect:zosConnect-2.0</feature>  
    <feature>zosconnect:zosConnectCommands-1.0</feature>  
  </featureManager>  
  <!--  
    To access this server from a remote client add a http endpoint  
  -->  
  <httpEndpoint id="defaultHttpEndpoint" host="*" httpPort="9443" />  
  <!--  
    add cors to allow cross origin access, e.g. when using angular  
  -->
```

```
<webApplication id="serverConfig-location" name="serverConfig"  
  location="${server.config.dir}">  
  <web-ext context-root="/server/config"  
    enable-file-serving="true" enable-directory-browsing="true">  
    <file-serving-attribute name="extendedDocumentRoot"  
      value="${server.config.dir}" />  
  </web-ext>  
</webApplication>
```

```
wg31.washington.ibm.com:9443/s... X +  
*****  
product = WAS FOR Z/OS 20.0.0.6, z/OS Connect 03.00.41 (wlp-1.0.41.c1200620200528-0414)  
wlp.install.dir = /shared/IBM/zosconnect/v3r0/wlp/  
server.config.dir = /var/zosconnect/servers/myServer/  
java.home = /shared/java/j8_0_64  
java.version = 1.8.0_261  
java.runtime = Java(TM) SE Runtime Environment (8.0.6.15 - pmz6480sr6fp15-20200724_01(SR6 FP15))  
os = z/OS (02.03.00; s390x) (en_US)  
process = 167788793@q31  
*****  
[2/19/21 15:48:18:901 GMT] 0000000b com.ibm.ws.kernel.launch.internal.FrameworkManager  
[2/19/21 15:48:19:861 GMT] 00000017 com.ibm.ws.config.xml.internal.XMLConfigParser  
/var/zosconnect/servers/myServer/includes/safSecurity.xml  
[2/19/21 15:48:19:892 GMT] 00000017 com.ibm.ws.config.xml.internal.XMLConfigParser  
/var/zosconnect/servers/myServer/includes/tpicIDProp.xml  
[2/19/21 15:48:19:894 GMT] 00000017 com.ibm.ws.config.xml.internal.XMLConfigParser  
/var/zosconnect/servers/myServer/includes/keyringOutbound.xml  
[2/19/21 15:48:19:901 GMT] 0000000b com.ibm.ws.kernel.launch.internal.FrameworkManager  
[2/19/21 15:48:19:901 GMT] 0000000b com.ibm.ws.kernel.launch.internal.FrameworkManager  
[2/19/21 15:48:19:901 GMT] 0000000b com.ibm.ws.kernel.launch.internal.FrameworkManager  
*****  
A CNWKE0001I: The server myServer has been launched.  
A CNWKG0028A: Processing included configuration resource:  
A CNWKG0028A: Processing included configuration resource:  
A CNWKG0028A: Processing included configuration resource:  
*****  
*****  
product = WAS FOR Z/OS 21.0.0.1, z/OS Connect 03.00.42 (wlp-1.0.48.c1210120210113-1459)  
wlp.install.dir = /shared/IBM/zosconnect/v3r0/wlp/  
server.config.dir = /var/zosconnect/servers/myServer/  
java.home = /shared/java/j8_0_64  
java.version = 1.8.0_261  
java.runtime = Java(TM) SE Runtime Environment (8.0.6.15 - pmz6480sr6fp15-20200724_01(SR6 FP15))  
os = z/OS (02.03.00; s390x) (en_US)  
process = 16779882#q31  
trace.specification  
*=info:Credentials=>all:SSL=>all:SSLChannel=>all:Security.Authorization=>all:UserRegistry=>all:com.ibm.ws.security.*=>all:com.ibm.ws.webcontainer.*=>all:com.ibm.ws.wlm.*=>all:org.apache.http.client.*=>all:zosConnect=>all:zosConnectSaf=>all  
*****  
[2/25/21 17:27:54:497 GMT] 0000000b id=00000000 com.ibm.ws.logging.internal.TraceSpecification  
I TRAS0018I: The trace state has been changed. The new trace state is  
*=info:Credentials=>all:SSL=>all:SSLChannel=>all:Security.Authorization=>all:UserRegistry=>all:com.ibm.ws.security.*=>all:com.ibm.ws.webcontainer.*=>all:com.ibm.ws.wlm.*=>all:org.apache.http.client.*=>all:zosConnect=>all:zosConnectSaf=>all  
[2/25/21 17:27:54:497 GMT] 00000016 id=078ec277 ty.thread.zos.hooks.internal.ThreadIdentityBundleFileWrapper > getEntry Entry org/apache/felix/scr/impl/manager  
/DependencyManager$SingleDynamicCustomizer.class  
[2/25/21 17:27:54:493 GMT] 00000016 id=078ec277 ty.thread.zos.hooks.internal.ThreadIdentityBundleFileWrapper < getEntry Exit org/apache/felix/scr/impl/manager  
/DependencyManager$SingleDynamicCustomizer.class  
[2/25/21 17:27:54:491 GMT] 00000017 id=00000000 com.ibm.ws.zos.core.internal.CoreBundleActivator  
value is set to 0000. I CWWKB0121I: The server process UMASK  
INF/com.ibm.ws.zos.logging.config.xml  
[2/25/21 17:27:54:494 GMT] 00000017 id=32c3d2ff ty.thread.zos.hooks.internal.ThreadIdentityBundleFileWrapper > getEntry Entry OSGI-  
INF/com.ibm.ws.zos.logging.config.xml  
[2/25/21 17:27:54:494 GMT] 0000001b id=459954a0 ty.thread.zos.hooks.internal.ThreadIdentityBundleFileWrapper > getEntry Entry com/ibm/ws/config/xml/internal/
```

© 2017, 2021 IBM Corporation

mitchj@us.ibm.com

# Tech/Tip: Providing access to service archives files

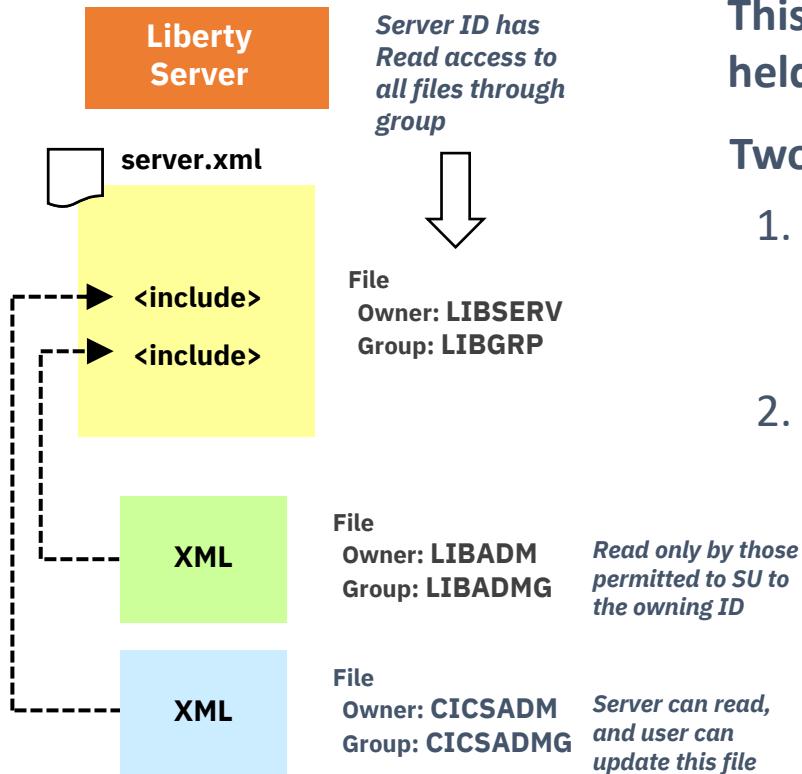


Name	Last Modified	Size	Description
<a href="#">apis</a>	Fri Feb 19 13:46:13 GMT 2021	-	Directory
<a href="#">services</a>	Sat Feb 20 20:54:41 GMT 2021	-	Directory
<a href="#">apiRequesters</a>	Wed Feb 07 17:59:04 GMT 2018	-	Directory
<a href="#">rules</a>	Tue Jan 26 20:34:05 GMT 2021	-	Directory

```
<webApplication id="resources-location" name="resources" location="${server.config.dir}/resources/zosconnect">
<web-ext context-root="resources/zosConnect"
  enable-file-serving="true" enable-directory-browsing="true">
  <file-serving-attribute name="extendedDocumentRoot"
    value="${server.config.dir}/resources/zosconnect" />
</web-ext>
</webApplication>
```

Name	Last Modified	Size	Description
<a href="#">cscvincDeleteService.sar</a>	Thu Feb 18 18:02:19 GMT 2021	4362	File
<a href="#">cscvincInsertService.sar</a>	Thu Feb 18 18:02:19 GMT 2021	4491	File
<a href="#">cscvincSelectService.sar</a>	Thu Feb 18 18:02:19 GMT 2021	4590	File

# Liberty supports server XML “include” file processing



This allows portions of the configuration to be held in files outside the main server.xml file

Two primary uses:

1. Hold sensitive configuration information in file that is READ to select people, but not the read group
2. Allow a user to update their portion of the server configuration, but not other parts of it

For the second use-case it is important to ensure the user can not override configuration in the main XML. Use the "onConflict" tag in the <include> element:

```
<include location="myIncludeFile.xml" onConflict="IGNORE"/>
```

This tells Liberty to ignore XML elements in include file that are also found in the main server.xml

It does not prevent them from injecting configuration elements not found in the main server.xml. If there is a concern about that, don't use include processing.

**Yes, nesting of includes is possible**

# An example, using “include” to manage the server XML



- Setup a server.xml using ‘include’ statements and allow other administrator to manage those included files, but not the server.xml itself.
- Control what configuration can be overridden in included files using the ‘onConflict’ option provided with the include element (see Ignore, Replace, Merge).

[https://www.ibm.com/support/knowledgecenter/en/SSAW57\\_liberty/com.ibm.websphere.wlp.nd.multiplatform.doc/ae/cwlp\\_config\\_include.html](https://www.ibm.com/support/knowledgecenter/en/SSAW57_liberty/com.ibm.websphere.wlp.nd.multiplatform.doc/ae/cwlp_config_include.html)

## server.xml (owned by ID ADMIN1)

```
<featureManager>
  <feature>appSecurity-1.0</feature>
<featureManager>
<include location="${server.config.dir}/includes/db2.xml
onConflict="IGNORE"/>
<include location="${server.config.dir}/includes/cics.xml
onConflict="IGNORE"/>
<include location="${server.config.dir}/includes/imsDB.xml
onConflict="IGNORE"/>
```

## imsDB.xml (owned by a IMS administrator)

```
<server description="IMS DATABASE">
<featureManager>
  <feature>zosconnect:dbService-1.0</feature>
</featureManager>
<connectionFactory id="DFSIVPACConn">
<properties.imsudbJLocal databaseName="DFSIVPA"
datastoreName="IVP1" driverType="4" portNumber="5555"
datastoreServer="wg31.washington.ibm.com"
user="USER1" password="USER1" flattenTables="True"/>
</connectionFactory>
</server>
```

mitchj@us.ibm.com

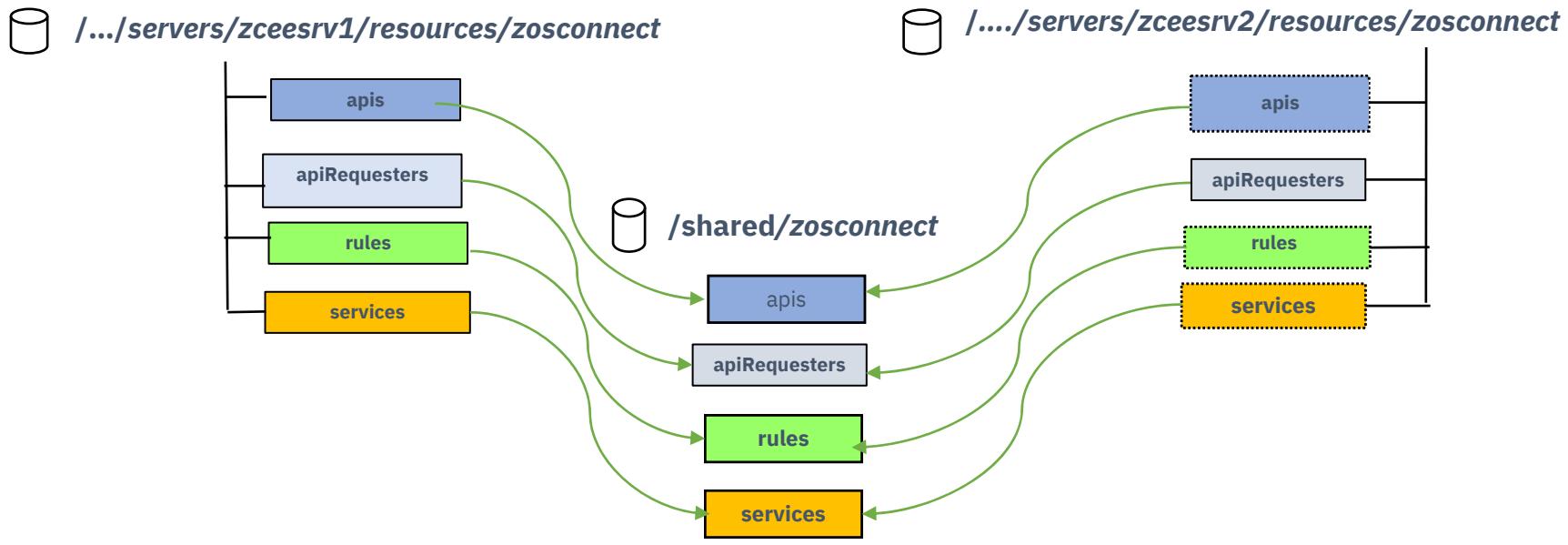
## db2.xml (owned by a DBA)

```
<server description="Db2 REST">
<zosconnect_zosConnectServiceRestClientConnection
  id="Db2Conn"
  host="wg31.washington.ibm.com" port="2446"
  basicAuthRef="dsn2Auth" />
<zosconnect_zosConnectServiceRestClientBasicAuth id="dsn2Auth"
  applName=DSN2APPL"/>
</server>
```

## cics.xml (owned by a CICS administrator)

```
<server description="CICS">
<featureManager>
  <feature>zosconnect:cicsService-1.0</feature>
</featureManager>
<zosconnect_cicsIpicConnection id="catalog"
  host="wg31.washington.ibm.com" port="1491"/>
<zosconnect_cicsIpicConnection id="cscvinc"
  host="wg31.washington.ibm.com" port="1492"
  zosConnectApplid= "ZOSCONN "
  zosConnectNetworkid= " ZOSCONN " />
</server>
```

# Remember the use of symbolic links to share artifacts



*Use symbolic links*

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

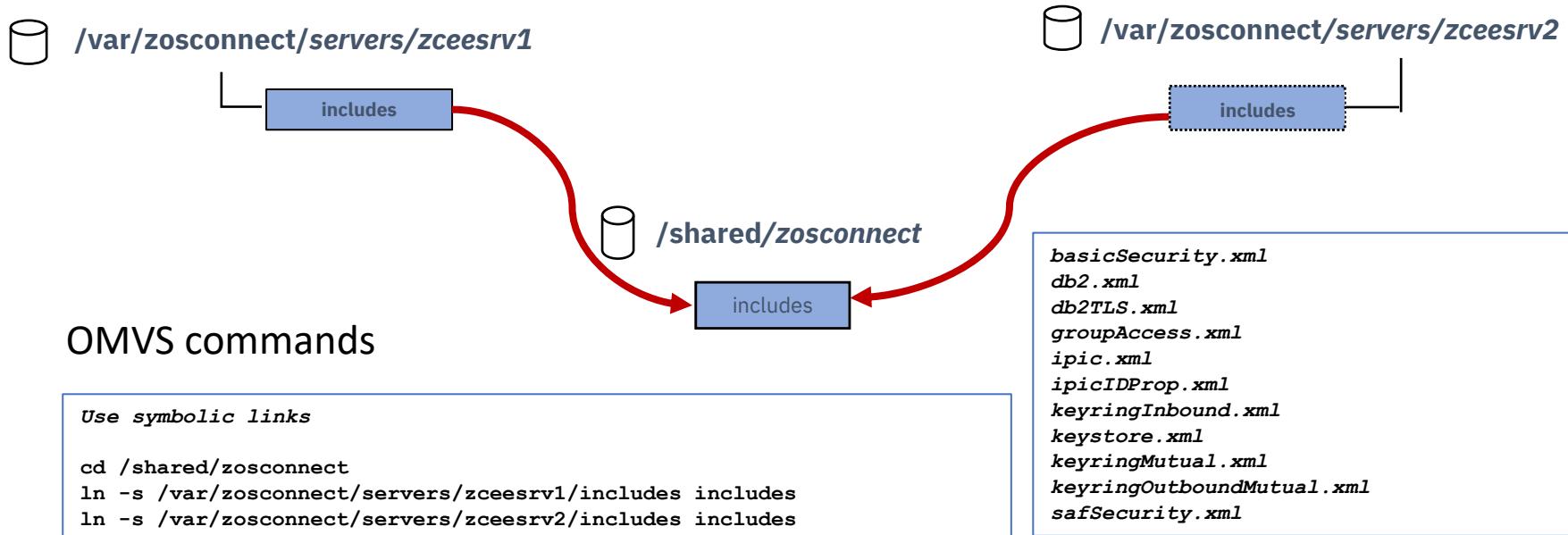
ln -s ./../servers/zceesrv1/resources/zosconnect/apiRequesters apiRequesters
ln -s ./../servers/zceesrv2/resources/zosconnect/apiRequesters apiRequesters

ln -s ./../servers/zceesrv1/resources/zosconnect/rules rules
ln -s ./../servers/zceesrv2/resources/zosconnect/rules rules

ln -s ./../servers/zceesrv1/resources/zosconnect/services services
ln -s ./../servers/zceesrv2/resources/zosconnect/services services
```

**F ZCEESRV1,ZCON,REFRESH  
F ZCEESRV2,ZCON,REFRESH**

# Sharing security XML configuration files between servers



Each server.xml file includes these statements

```
<include location="${server.config.dir}/includes/basicSecurity.xml"/>
<include location="${server.config.dir}/includes/ipicIDProp.xml"/>
<include location="${server.config.dir}/includes/keyringOutboundMutual.xml"/>
<include location="${server.config.dir}/includes/groupAccess.xml"/>
<include location="${server.config.dir}/includes/shared.xml"/>
<include location="${server.config.dir}/includes/oauth.xml"/>
```

These includes can be in a shared location which then can be accessed from multiple servers

F BAQSTRT,REFRESH,CONFIG

# Server XML configuration files – bootstrap.properties file



zceesrv1's bootstrap.properties

```
httpPort=9080
httpsPort=9443
ipicPort=1491
cicsHost=wg31.washington.ibm.com
network=ZOSCONN1
applid=ZOSCONN1
```

zceesrv2's bootstrap.properties

```
httpPort=9090
httpsPort=9453
ipicPort=1492
cicsHost=wg31.washington.ibm.com
network=ZOSCONN2
applid=ZOSCONN2
```

server.xml

```
<!-- To access this server from a remote client, add a host attribute to the following
element, e.g. host="*" -->
<httpEndpoint id="defaultHttpEndpoint"
               host="*"
               httpPort="\${httpPort}"
               httpsPort="\${httpsPort}" />
```

ipicIDProp.xml

```
<zosconnect_cicsIpicConnection id="catalog"
                                host="\${cicsHost}" port="\${ipicPort}"
                                zosConnectNetworkid="\${network}"    zosConnectApplid="\${applid}"/>

<zosconnect_cicsIpicConnection id="cscvinc"
                                host="\${cicsHost}" port="\${ipicPort}"
                                zosConnectNetworkid="\${network}"    zosConnectApplid="\${applid}"/>

<zosconnect_cicsIpicConnection id="miniloan"
                                host="\${cicsHost}" port="\${ipicPort}"
                                zosConnectNetworkid="\${network}"    zosConnectApplid="\${applid}"/>
```

[BAQR0660E Failed to establish CICS connection IPIC Capability exchange error](#)

© 2017, 2021 IBM Corporation

mitchj@us.ibm.com

[tinyurl.com/3xc76kpp](http://tinyurl.com/3xc76kpp)

# Sharing security XML configuration files – variables



variablesBBG.xml

```
<variable name= "unauthenticatedUser" value= "WSGUEST" />
<variable name="profilePrefix" value="BBGZDFLT" />
```

variablesEMJ.xml

```
<variable name= "unauthenticatedUser" value="ZCGUEST" />
<variable name="profilePrefix" value="EMJZDFLT" />
```

safSecurity.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<server description="SAF security">

    <!-- Enable features -->
    <featureManager>
        <feature>appSecurity-2.0</feature>
        <feature>zosSecurity-1.0</feature>
    </featureManager>

    <webAppSecurity allowFailOverToBasicAuth="true" />

    <safRegistry id="saf" />
    <safAuthorization racRouteLog="ASIS" />
    <safCredentials unauthenticatedUser="${unauthenticatedUser}"
        profilePrefix="${profilePrefix}" />

</server>
```

server.xml

```
<server description="new server">
<include location="${server.config.dir}/includes/safSecurity.xml"/>
<include location="${server.config.dir}/variablesEMJ.xml"/>

    <!-- Enable features -->
    <featureManager>
        <feature>zosconnect:zosConnect-2.0</feature>
        <feature>zosconnect:zosConnectCommands-1.0</feature>
    </featureManager>
```

# A practical example-PTF V3.0.35 included a CORS update



July 2020	
V3.0.35 (APAR PH26291) Server code update	<p><b>Enhancements</b></p> <ul style="list-style-type: none"><li>The text of messages BAQR0417W and BAQR0418W has been updated. For more information, see z/OS Connect EE <a href="#">Runtime Messages</a>.</li></ul> <p><b>Fixes</b></p> <ul style="list-style-type: none"><li>PH21761 A CICS region reports <b>SOS DFHSM0133 WBSETBUF</b> when z/OS Connect EE requester is in use.</li><li>PH25345 Passing user credentials in the request body to the authentication server to obtain a JWT causes a NPE in z/OS Connect EE.</li><li>PH21819 z/OS Connect EE sets some CORS headers automatically.</li></ul>
	<p><b>Attention</b></p> <p>When this fix is applied, additional CORS configuration is required in <code>server.xml</code> to enable connections from the z/OS Connect EE API toolkit and JavaScript clients. For more information, see <a href="#">Configuring Cross-Origin Resource Sharing on a z/OS Connect Enterprise Edition Server</a></p>

`cors.xml`

```
<?xml version="1.0" encoding="UTF-8"?>
<server description="CORS entries">

    <!-- add cors to allow cross origin access, e.g. when using swagger doc from zOS Connect Enterprise
Edition -->
    <cors id="defaultCORSConfig"
        domain="/"
        allowedOrigins="*"
        allowedMethods="GET, POST, PUT, DELETE, OPTIONS"
        allowedHeaders="Origin, Content-Type, Authorization, Cache-Control, Expires, Pragma"
        allowCredentials="true"
        maxAge="3600"/>

</server>
```

`server.xml`

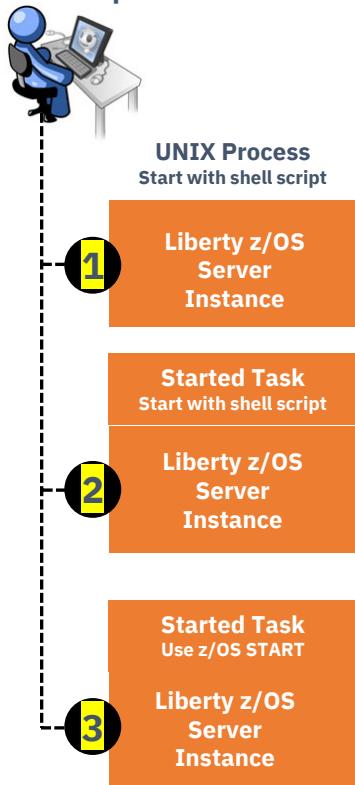
```
<include location="${server.config.dir}/includes/cors.xml"/>
```

# z/OS : Starting Liberty Servers



All three options result in a Liberty z/OS server, and functionally there's very little difference.

When started as a UNIX process, the MODIFY command interface is not present. For production use, the best practice is to use a started task.



1. UNIX Process
  - Use the 'server' shell script in the installation /bin directory
  - Syntax: `server start zceesvr1`
  - ID of server will be based on ID that issued the command
2. Started Task using server shell script (`server start zceesrv1`)
  - Set **WLP\_ZOS\_PROCEDURE** environment variable in `server.env` file
  - Example: `WLP_ZOS_PROCEDURE=ZCEEPROC.JOBNAME=ZCEESVR1.PARMS='ZCEESVR1'`
  - ID of the server will be based on the SAF STARTED profile that takes effect
3. **Started Task using START command**
  - **Common procedure:** `START ZCEEPROC,JOBNAME=ZCEESVR1,PARMS='ZCEESVR1'`
  - **Dedicated proc:** `START ZCEEPROC`
  - **ID of the server will be based on the SAF STARTED profile that takes effect**

Expectation is for production servers #3 will be used

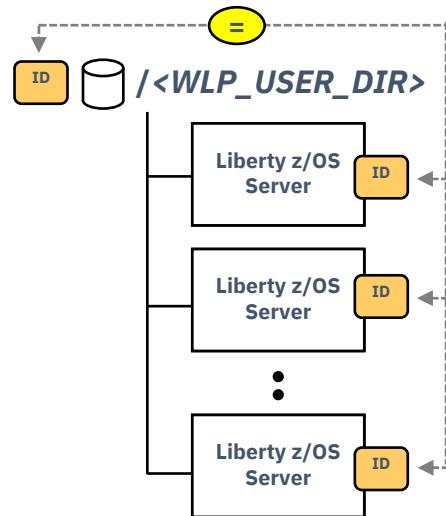
Liberty z/OS good practices:

<https://www.ibm.com/support/pages/node/6355605>

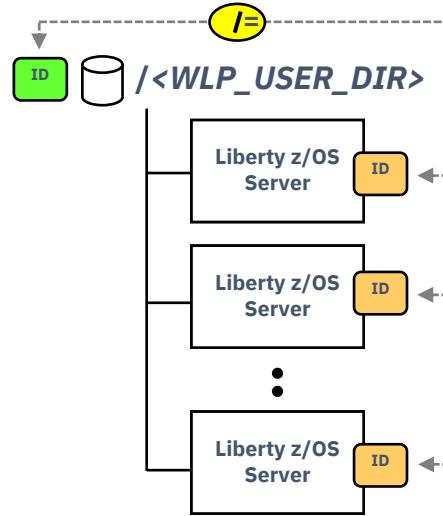
# z/OS Security – Range of options – Started Task IDs



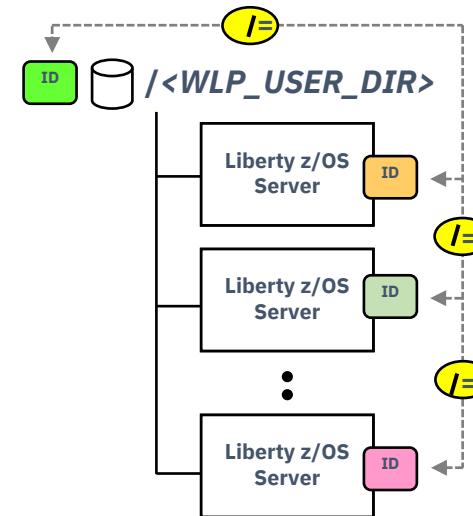
On z/OS, the best practice for Liberty servers in production is that they run as ‘Started Tasks’ (STCs).



- Multiple servers
- All have same STC ID
- STC ID = File Owner ID



- Multiple servers
- All have same STC ID
- STC ID  $\neq$  File Owner ID



- Multiple servers
- Different STC IDs
- STC IDs  $\neq$  File Owner ID

Should all servers sharing WLP\_USER\_DIR share the same STC ID?  
It is a matter of the degree of identity isolation that is required

# z/OS : Assigning ID to started tasks: SAF STARTED



The first question here is whether you wish to have a common started task ID that is shared among servers, or if you wish each server to have a unique ID

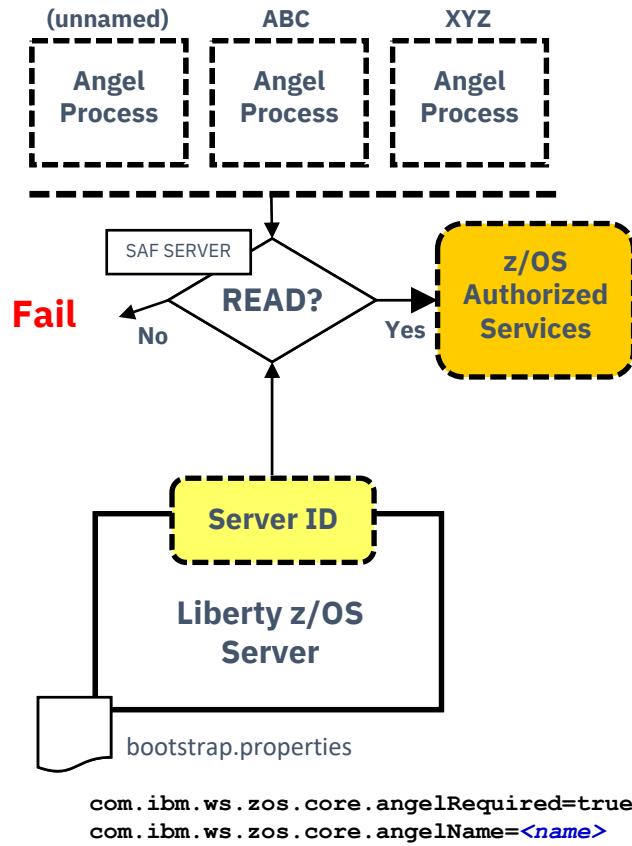
Then the second question is whether servers under a WLP\_USER\_DIR will share a common JCL start proc, or use unique start procs for each server

	<i>Common Identity per task</i>	<i>Unique Identities per task</i>
<i>Common JCL Procedure</i>	<pre>RDEFINE STARTED ZCEEPROC.* S ZCEEPROC,JOBNAM=server1,PARMS='server1' S ZCEEPROC,JOBNAM=server2,PARMS='server2'</pre>	<pre>RDEFINE STARTED ZCEEPROC.server1 RDEFINE STARTED ZCEEPROC.server1 S ZCEEPROC,JOBNAM=server1,PARMS='server1' S ZCEEPROC,JOBNAM=server2,PARMS='server2'</pre>
<i>Unique JCL Procedure per server</i>	<pre>RDEFINE STARTED ZCEE*.* S ZCEESRV1,JOBNAM=server1,PARMS='server1' S ZCEESRV2,JOBNAM=server2,PARMS='server2'</pre>	<pre>RDEFINE STARTED ZCEESRV1.* RDEFINE STARTED ZCEESRV2.* S ZCEESRV1,JOBNAM=server1,PARMS='server1' S ZCEESRV2,JOBNAM=server2,PARMS='server2'</pre>

Note: Using unique JCL procedure eliminates the need to specify PARMS on the start commands

**It's possible to use a combination of the above, even under the same WLP\_USER\_DIR. So there's no "one best answer" here. What's best is what's best for you.**

# z/OS : The Angel process – what is this about?



**The Angel Process is a started task that is used to protect access to z/OS privileged or authorized services. This is done with SAF SERVER profiles.**

- Authorized services include: WOLA, SAF, WLM, RRS, DUMP
- The ability to start multiple Angel processes on an LPAR was introduced in 16.0.0.4. This is called "Named Angels". It provides a way to separate Angel usage between Liberty servers:
  - An Angel process can be started with a NAME='<name>' parameter (or it can be started as a "default" without a name). The name may be up to 54 characters.
  - Liberty servers can be pointed at a specific Angel with a bootstrap property

## Best practice:

- You may create separate named Angels for isolation of Test and Production, but do not take this practice too far. A few Angels, yes; dozens, no.
- Establish automation routines to start the Angels at IPL
- Grant SAF GROUP access to the SERVER profiles, then connect server IDs as needed

Current list of Liberty Features

[https://www.ibm.com/support/knowledgecenter/SSEQTP\\_liberty/com.ibm.websphere.wlp.doc/ae/rwlp\\_feat.html](https://www.ibm.com/support/knowledgecenter/SSEQTP_liberty/com.ibm.websphere.wlp.doc/ae/rwlp_feat.html)

# z/OS : SAF SERVER profiles related to the Angel



- Establish all the SERVER profiles ahead of time. Existence of profile does not grant access; READ to it does.
- Determine what access a server needs and grant only that; check "is available" messages in messages.log to verify

Tech/Tip: The SAFLOG parameter was added in a recent Liberty drop. If this parameter is set to Y, additional security related messages will be written to the JES messages and console if a Liberty server does not have authorization to use an angel-controlled privileged function. See URL

[https://www.ibm.com/support/knowledgecenter/SS7K4U\\_liberty/com.ibm.websphere.wlp.zseries.doc/ae/rwlp\\_newinrelease.html](https://www.ibm.com/support/knowledgecenter/SS7K4U_liberty/com.ibm.websphere.wlp.zseries.doc/ae/rwlp_newinrelease.html)

# SAF APPL and EJBRole Resources

*Connect z/OS Connect users to a common group*

**CONNECT (FRED,USER1,JOHNSON) GROUP(ZCEEUSRS)**

*Define a APPL profile for the server's SAF profilePrefix and permit access*

**RDEFINE APPL BBGZDFLT UACC(NONE) OWNER(SYS1)**

**PERMIT BBGZDFLT CLASS(APPL) ACCESS(READ) +  
ID(WSGUEST#, ZCEEUSRS)**

**SETROPTS RACLIST(APPL) REFRESH**

*Define an EJBROLE profile for the server's SAF profilePrefix and permit access*

**RDEFINE EJBROLE BBGZDFLT.zos.connect.access.roles.zosConnectAccess +  
OWNER(SYS1) UACC(NONE)**

**PERMIT BBGZDFLT.zos.connect.access.roles.zosConnectAccess +  
CLASS(EJBROLE) ID(ZCEEUSRS) ACCESS(READ)**

*Refresh the EJBROLE in storage profiles*

**SETROPTS RACLIST(EJBROLE) REFRESH**

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

- # [https://www.ibm.com/support/knowledgecenter/SS7K4U\\_liberty/com.ibm.websphere.wlp.zseries.doc/ae/twlp\\_config\\_security\\_saf.html](https://www.ibm.com/support/knowledgecenter/SS7K4U_liberty/com.ibm.websphere.wlp.zseries.doc/ae/twlp_config_security_saf.html)  
[https://www.ibm.com/support/knowledgecenter/SS4SVW\\_beta/securing/saf\\_unauthenticated\\_id.html#concept\\_saf\\_unauthenticated\\_id](https://www.ibm.com/support/knowledgecenter/SS4SVW_beta/securing/saf_unauthenticated_id.html#concept_saf_unauthenticated_id)

# Tech/Tip: z/OS : SAF SURROGAT Resources

RACF Surrogate access allows a designated administrative identity the ability to invoke commands and perform functions as if they were running under the identity that will be used for the z/OS Connect server started task. This may be useful because identities associated with started task are normally restricted and cannot be used for accessing TSO or OMVS shells,

Use the following examples as guides and create the surrogate resources and permit access. In these examples, *LIBSERV* represents the RACF identity under which the z/OS Connect server will be running and *adminUser* represent the administrative RACF identity.

*Define a SURROGAT profile for the server's SAF identity*

**RDEFINE SURROGAT BPX.SRV.*LIBSERV***

*Define a SURROGAT submit profile to allow job submission as the server's SAF identity*

**RDEFINE SURROGAT *LIBSERV*.SUBMIT**

*Permit an administrative identity to act as a surrogate of the Liberty task identity*

**PERMIT BPX.SRV.*LIBSERV* CLASS(SURROGAT) ID(*adminGrp*) ACC(READ)**

**PERMIT *LIBSERV*.SUBMIT CLASS(SURROGAT) ID(*adminGrp*) ACC(READ)**

*Refresh the SURROGAT in storage profiles*

**SETROPTS RACLIST(SURROGAT) REFRESH**

Now any identity in group *adminGrp* can submit JCL with the *USER=LIBSERV* parameter on the job card or use the OMVS switch user command (*su -s LIBSERV*) to execute OMVS scripts or commands as LIBSERV.

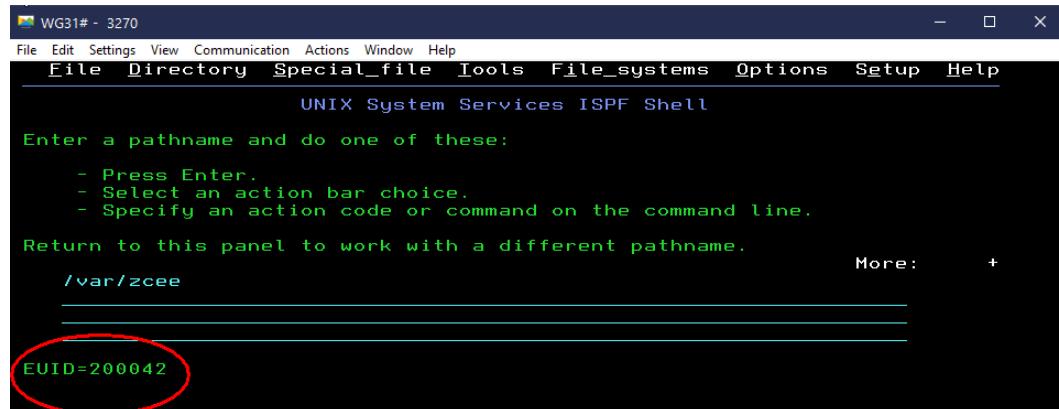
# Tech/Tip: z/OS : A JCL example of using SURROGAT access

```
//MYSERVER JOB 'ZCEE',CLASS=A,REGION=0M,NOTIFY=&SYSUID,USER=LIBSERV
//*****
//**  SET SYMBOLS
//*****
//EXPORT EXPORT SYMLIST=(*)
// SET JAVAHOME='/usr/lpp/java/J8.0_64'
// SET ZCEEPATH='/usr/lpp/IBM/zosconnect/v3r0'
// SET SERVER='baqstrt'
// SET TEMPLATE='zosconnect:default'
// SET WLPUSER='/var/zosconnect'
// SET USER='LIBSERV'
// SET GROUP='LIBGRP'
//*****
//** Step ZCEESRVR - Use the zosconnect command to create a server
//*****
//ZCEESRVR EXEC PGM=IKJEFT01,REGION=0M
//SYSTSPRT DD SYSOUT=*
//SYSERR DD SYSOUT=*
//STDOUT DD SYSOUT=*
//SYSTSIN DD *,SYMBOLS=EXECSYS
BPXBATCH SH +
export JAVA_HOME=&JAVAHOME; +
export WLP_USER_DIR=&WLPUSER; +
&ZCEEPATH/bin/zosconnect create &SERVER +
--template=&TEMPLATE
//*****
//** Step CHOWN - Change directory and file ownership
//*****
//CHOWN EXEC PGM=IKJEFT01,REGION=0M
//SYSERR DD SYSOUT=*
//STDOUT DD SYSOUT=*
//SYSTSPRT DD SYSOUT=*
//SYSTSIN DD *,SYMBOLS=EXECSYS
BPXBATCH SH +
export WLP_USER_DIR=&WLPUSER; +
chown -R &USER:&GROUP $WLP_USER_DIR/servers/&SERVER
```

There is no need provide LIBSERV's password, in fact LIBSERV may be protected and not even have a password. Any files or directories created will be owned by LIBSERV.

Alternatively, use the change ownership command, *chown*, to change the user and group attributes.

# Tech/Tip: z/OS : ISPF/OMVS examples of using SURROGAT access



WG31# - 3270

File Edit Settings View Communication Actions Window Help

File Directory Special\_file Tools File\_systems Options Setup Help

UNIX System Services ISPF Shell

Enter a pathname and do one of these:

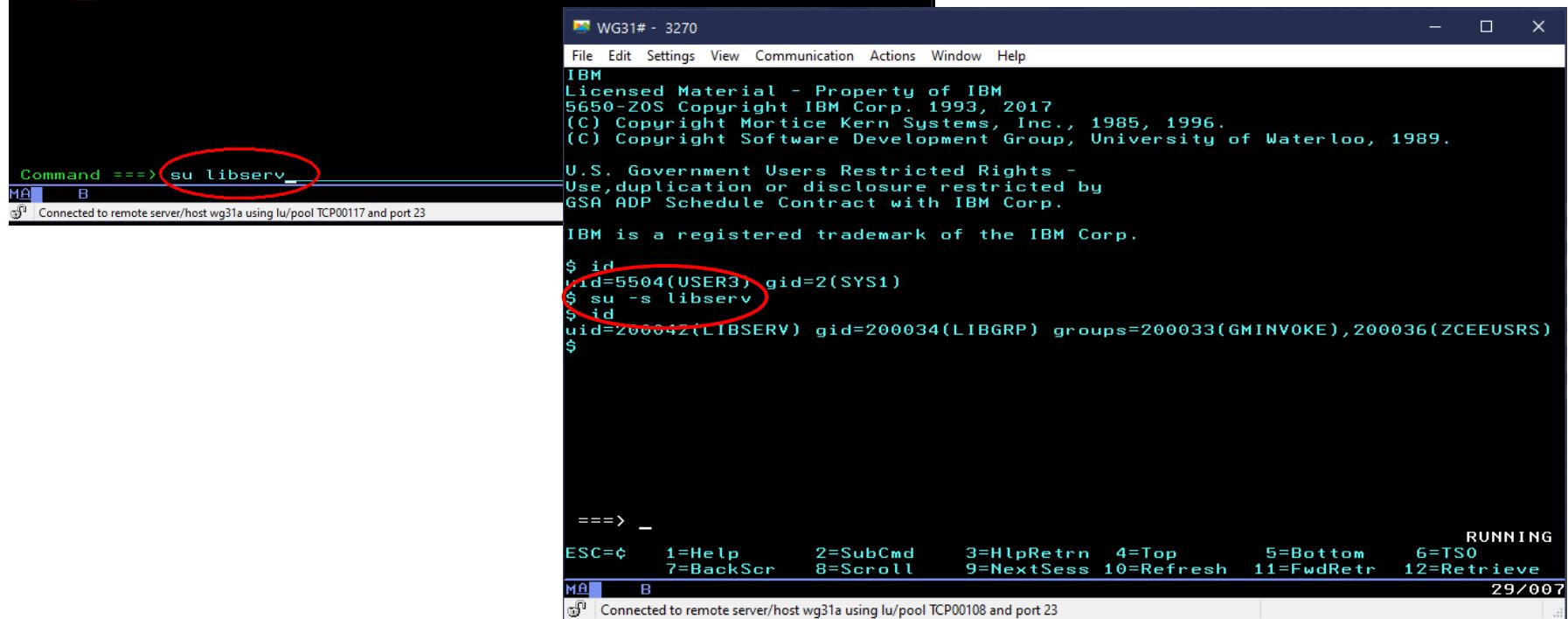
- Press Enter.
- Select an action bar choice.
- Specify an action code or command on the command line.

Return to this panel to work with a different pathname.

More: +

/var/zcee

EUID=200042



WG31# - 3270

File Edit Settings View Communication Actions Window Help

IBM  
Licensed Material - Property of IBM  
5650-ZOS Copyright IBM Corp. 1993, 2017  
(C) Copyright Mortice Kern Systems, Inc., 1985, 1996.  
(C) Copyright Software Development Group, University of Waterloo, 1989.

U.S. Government Users Restricted Rights -  
Use, duplication or disclosure restricted by  
GSA ADP Schedule Contract with IBM Corp.

IBM is a registered trademark of the IBM Corp.

MA B

Connected to remote server/host wg31a using lu/pool TCP00117 and port 23

Command ==> su libserv

\$ id  
uid=5504(USER3) gid=2(SYS1)

\$ su -s libserv

\$ id  
uid=200042(LIBSERV) gid=200034(LIBGRP) groups=200033(GMINVOKE),200036(ZCEEUSRS)

\$

====> \_

RUNNING

ESC=< 1=Help 2=SubCmd 3=HlpRetrn 4=Top 5=Bottom 6=TSO  
7=BackScr 8=Scroll 9=NextSess 10=Refresh 11=FwdRetrn 12=Retrieve

MA B

Connected to remote server/host wg31a using lu/pool TCP00108 and port 23

29/007

# Tech/Tip: z/OS : SAF UNIXPRIV/FACILITY Resources

An alternative to using a surrogate access is to permit the identity under which the customization will be done to enhanced Unix privileges. Specially, permitting the identity to Unix privileges SUPERUSER.FILESYS, SUPERUSER.FILESYS.CHANGEPERMS and SUPERUSER.FILESYS.CHOWN.

- *Permit an administrative identity to write to any local directory or file*  
**PERMIT SUPERUSER.FILESYS CLASS(UNIXPRIV)  
ID(adminUser) ACC(CONTROL)**
- *Permit an administrative identity to change permission bit of any local directory or file*  
**PERMIT SUPERUSER.FILESYS.CHANGEPERMS CLASS(UNIXPRIV)  
ID(adminUser) ACC(READ)**
- *Permit an administrative identity to change the ownership of any directory or file*  
**PERMIT SUPERUSER.FILESYS.CHOWN CLASS(UNIXPRIV)  
ID(adminUser) ACC(READ)**
- *Permit an administrative identity switch to root (su -s root)*  
**PERMIT BPX.SUPERUSER CLASS(FACILITY) ID(adminUser) ACC(READ)**
- *Refresh the UNIXPRIV and/or FACILITY instorage profiles*  
**SETROPTS RACLIST(UNIXPRIV,FACILITY) REFRESH**

**Do not use these commands if you do not understand the implications.**

[https://www.ibm.com/support/knowledgecenter/en/SSLTBW\\_2.4.0/com.ibm.zos.v2r4.bpxb200/usspriv.htm](https://www.ibm.com/support/knowledgecenter/en/SSLTBW_2.4.0/com.ibm.zos.v2r4.bpxb200/usspriv.htm)

© 2017, 2021 IBM Corporation

mitchj@us.ibm.com

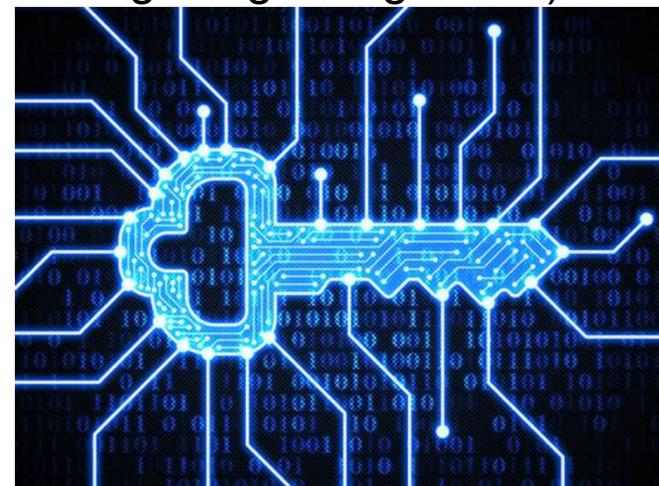
# **z/OS Connect Security**

## **Overview**

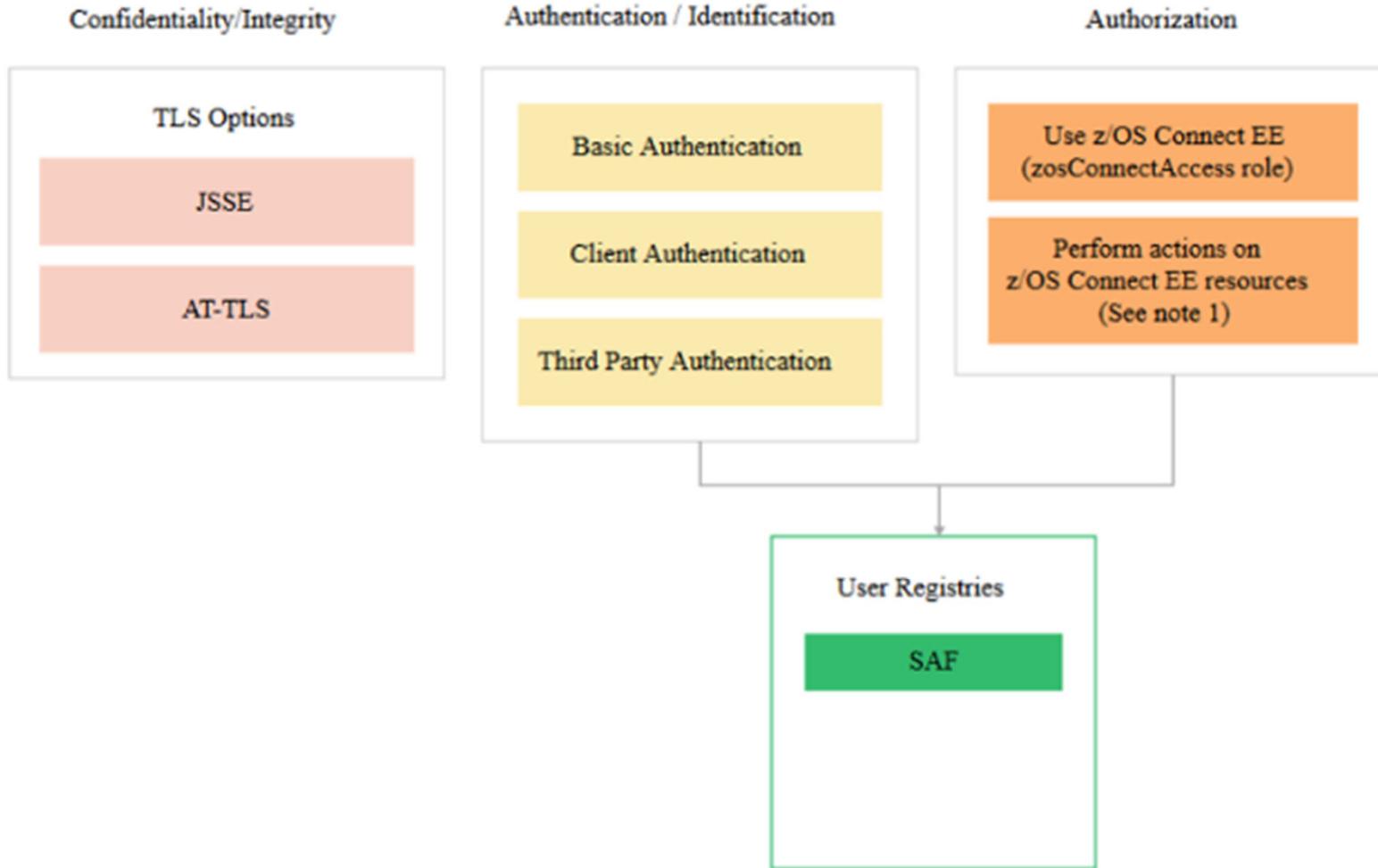
# General security terms or considerations

Security involves

- Identifying who or what is requesting access (**Authentication**)
  - Basic Authentication
  - Mutual Authentication using TLS
  - 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 generating/sending 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.

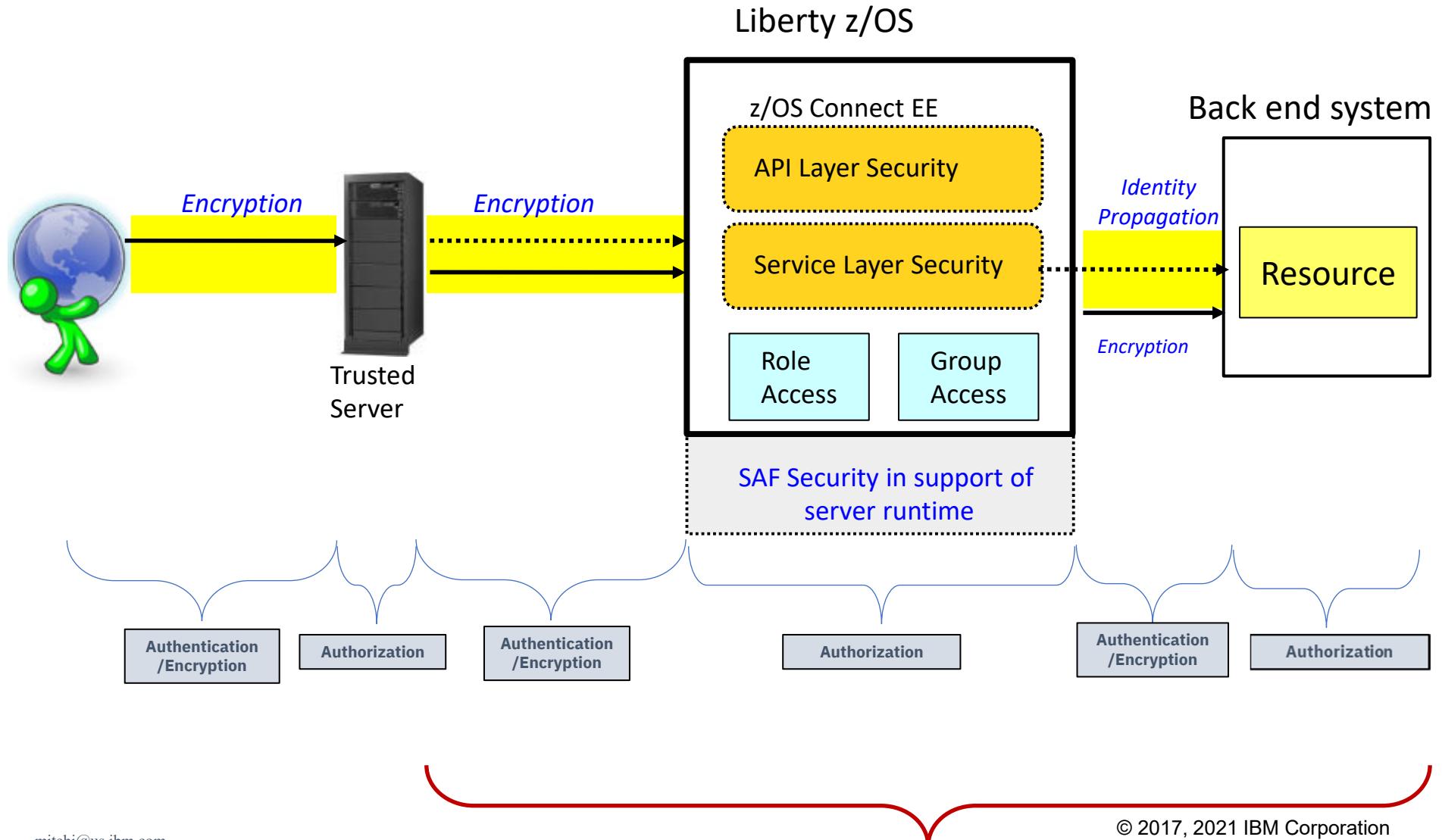


# Liberty and z/OS Connect EE security options

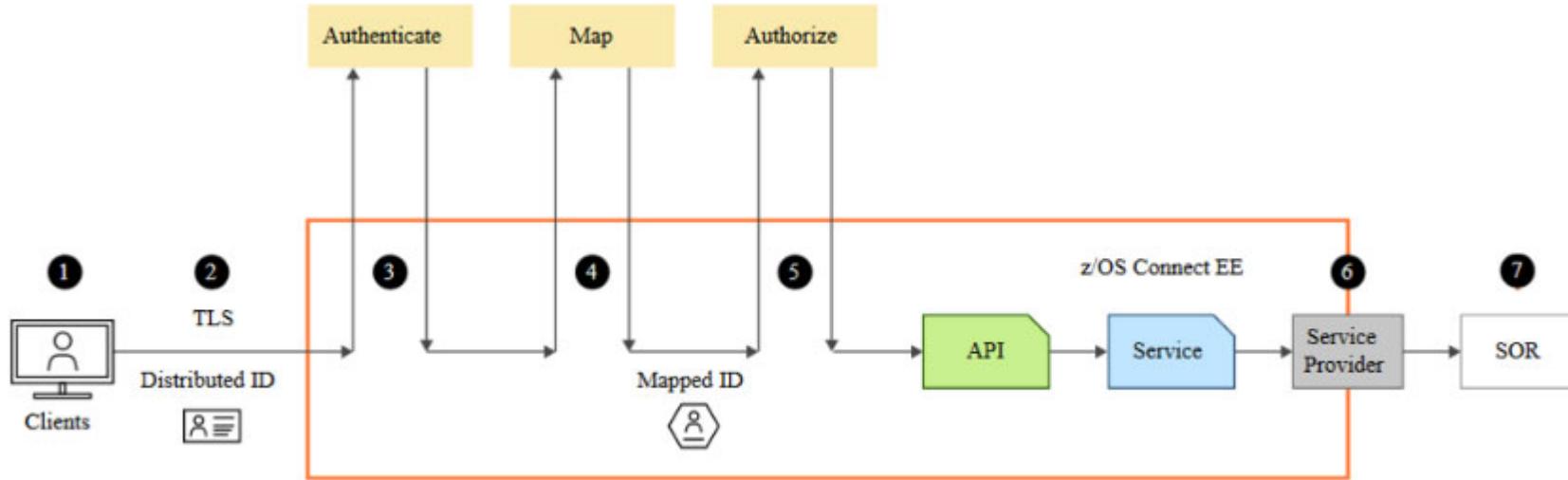


The actions which can be controlled by authorization are deploying, querying, updating, starting, stopping and deleting of APIs, services and API requesters.

# API provider security overview (waypoints)



# Details of a typical z/OS Connect EE API Provider security flow

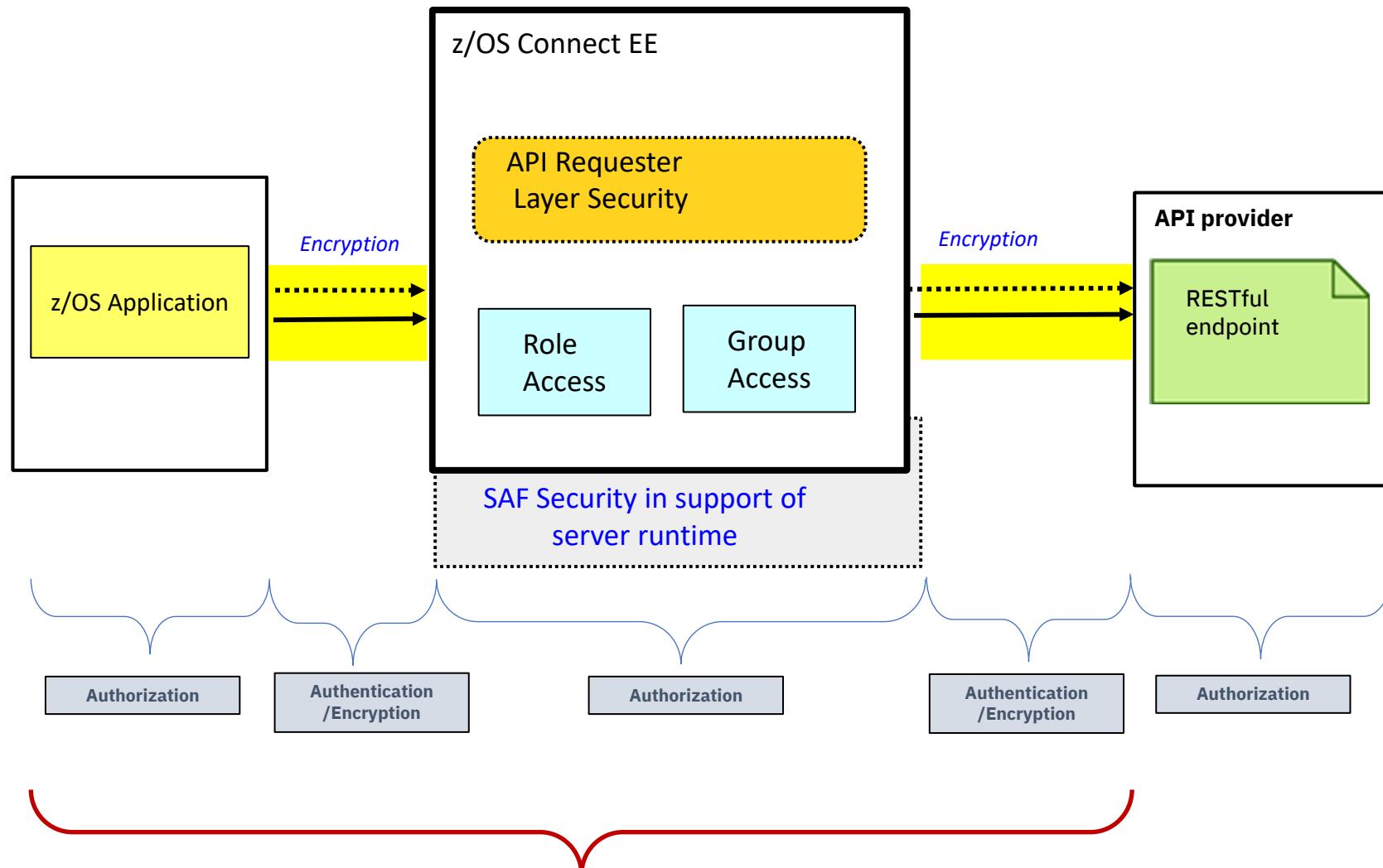


1. The credentials provided by the client
2. Secure the connection to the Liberty server
3. Authenticate the client. This can be within the Liberty server or by requesting verification from a third-party server
4. Map the authenticated identity to a user ID in the user registry
5. Authorize the mapped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

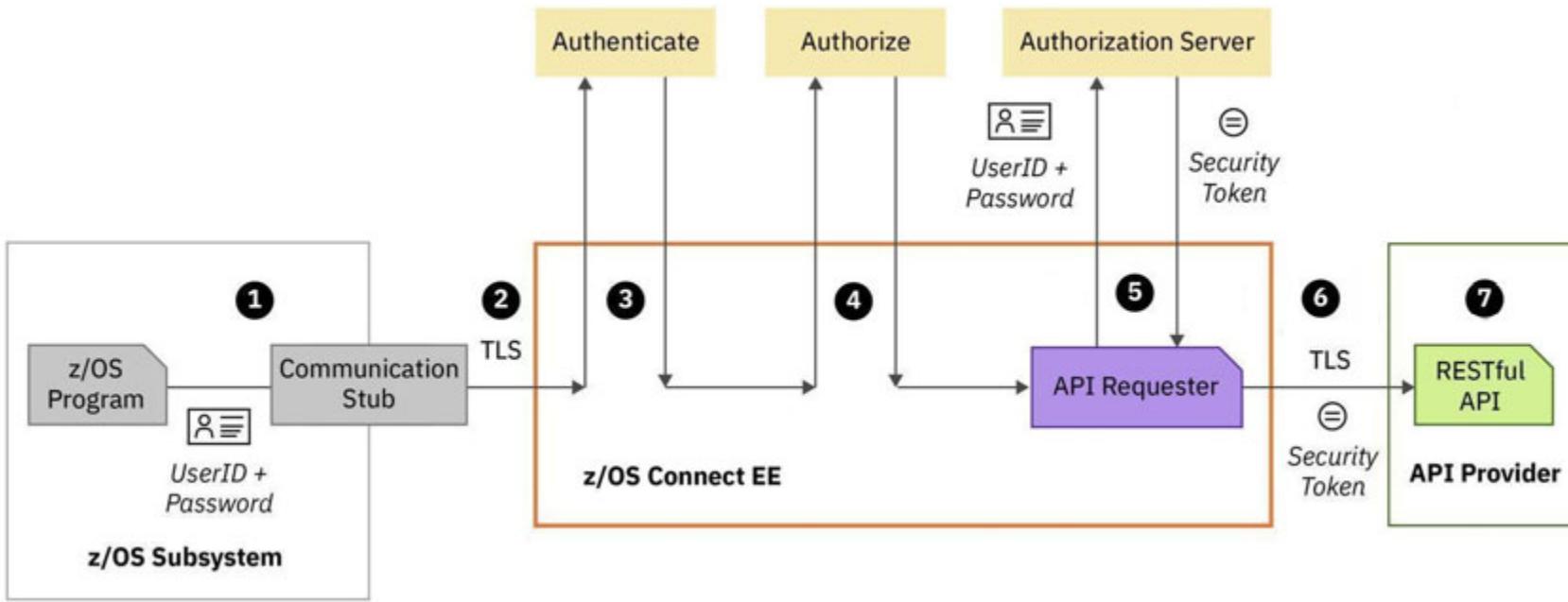
# API requester security – overview (waypoints)



Liberty z/OS



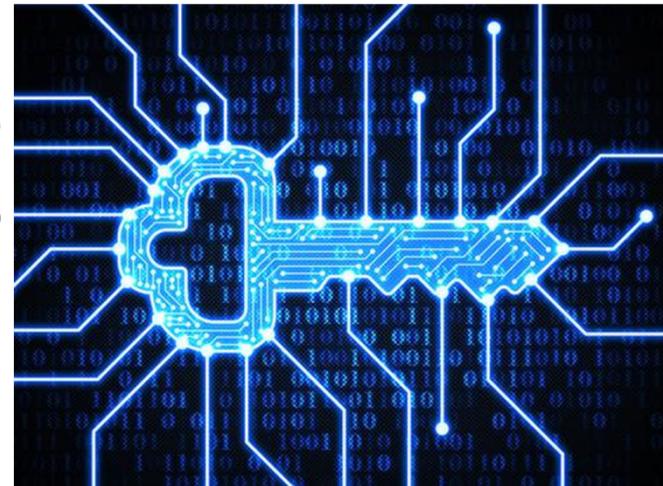
# Details of a typical z/OS Connect EE API Requester security flow



**Now let's explore security for  
inbound connections**

# General security terms or considerations

- Identifying who or what is requesting access (**Authentication**)
  - Basic Authentication
  - Mutual Authentication using TLS
  - 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 generating/sending 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.

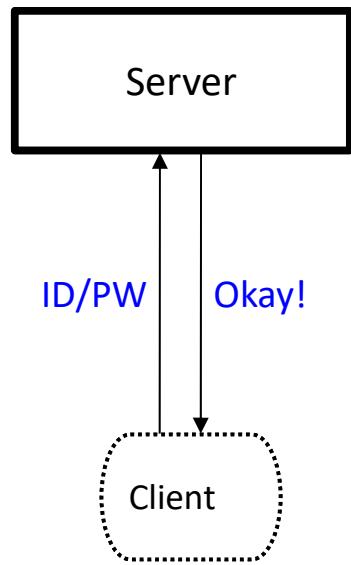


# Liberty Authentication Options



Several different ways this can be accomplished:

## Basic Authentication

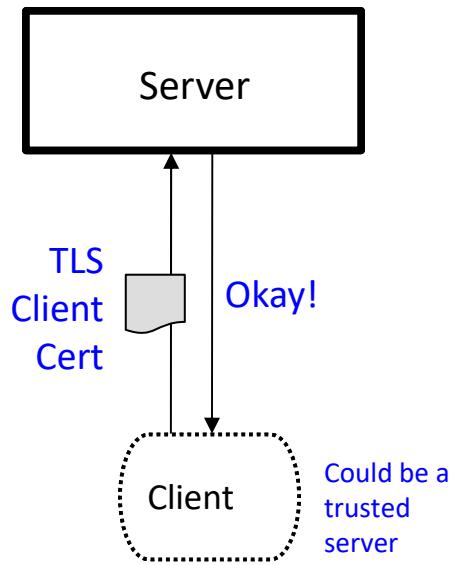


**Client supplies ID/PW or ID/PassTicket**

**Server checks registry:**

- **Basic (server.xml)**
- **SAF**

## Client Certificate



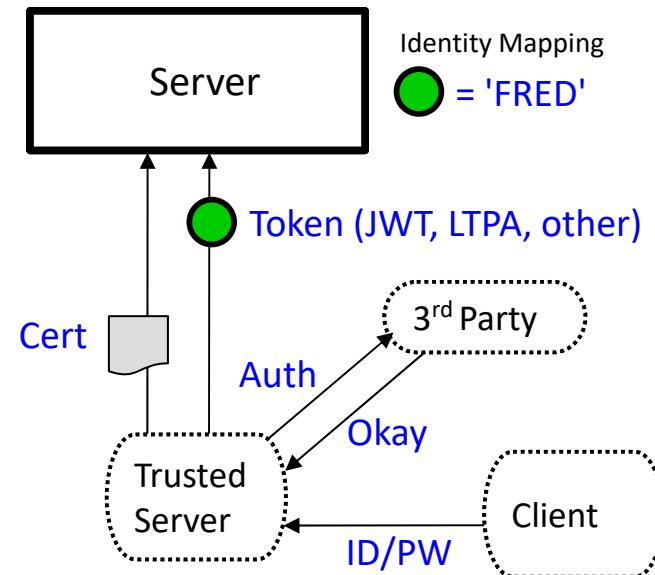
**Client supplies client personal certificate**

**Server validates client personal certificate and maps it to an identity**

**Registry options:**

- **SAF**

## Third Party Authentication



**Client authenticates to 3<sup>rd</sup> party sever**

**Client receives a trusted 3<sup>rd</sup> party token**

**Token flows to server and is mapped to an identity**

**Registry options:**

- **We may not need to know these details.**

# z/OS Connect Security server XML Configuration



```
<zosconnect_zosConnectManager  
    requireAuth="true|false"  
    requireSecure="true"/>  
  
<zosconnect_zosConnectAPIs>  
    <zosConnectAPI name="catalog"  
        requireAuth="true|false"  
        requireSecure="true"/>  
</zosconnect_zosConnectAPIs>  
  
<zosconnect_services>  
    <service id="selectByEmployee"  
        name="selectEmployee"  
        requireAuth="true|false"  
        requireSecure="true"/>  
</zosconnect_services>  
  
<zosconnect_apiRequesters>  
    requireAuth="true|false"  
    <apiRequester name="cscvincapi_1.0.0"  
        requireAuth="true|false"  
        requireSecure="true"/>  
</zosconnect_apiRequesters>  
  
<zosconnect_zosconnect_service  
    id="selectByEmployee"  
    name="selectEmployee"  
    requireAuth="true|false"  
    requireSecure="true"/>
```

Globally, requires that users specify security credentials to be authenticated order to access APIs, services and API requesters, unless overridden on the specific resource definitions.

Requires that users specify security credentials to be authenticated in order to access the API.

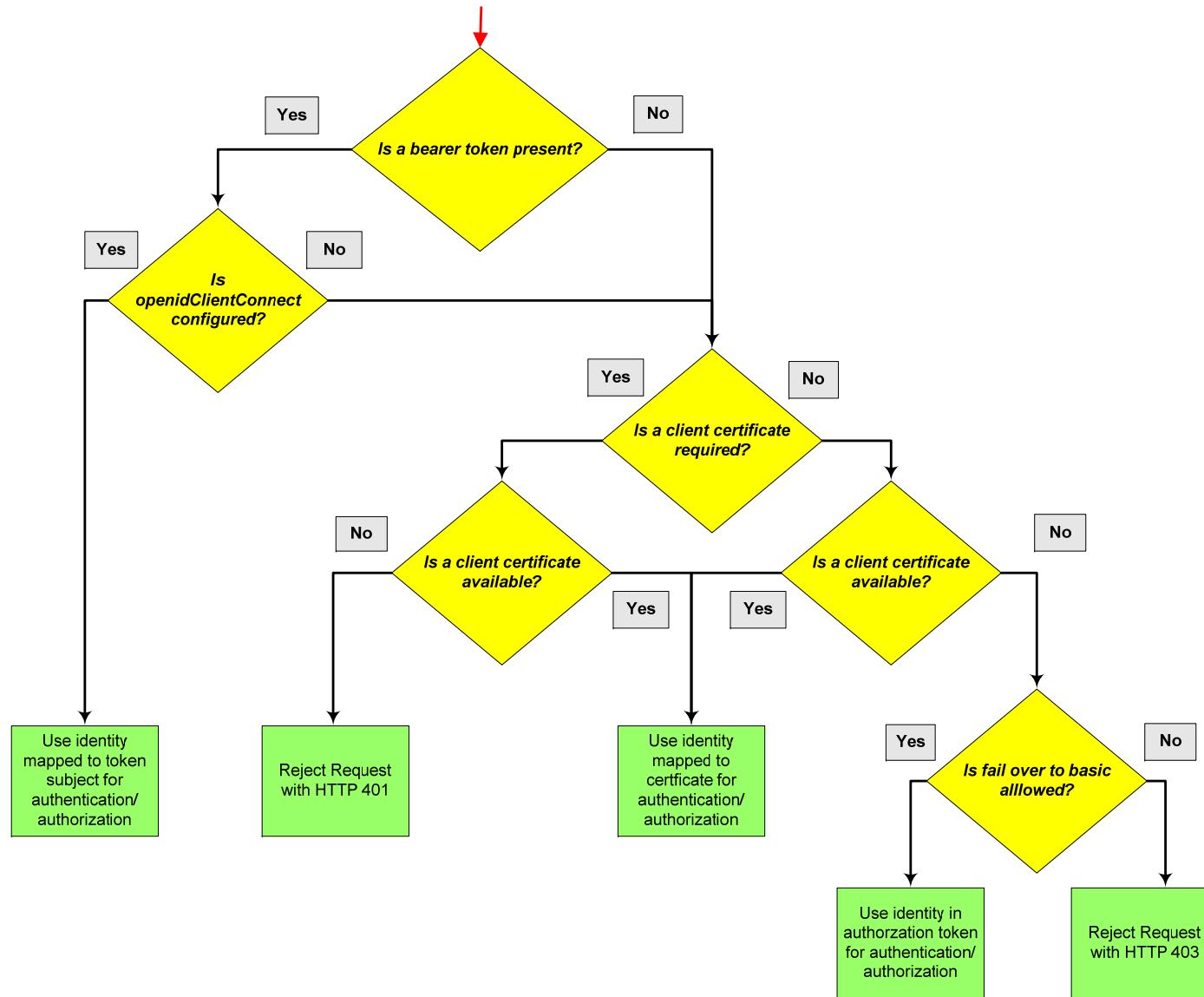
Requires that users specify security credentials to be authenticated in order to directly access the service. This attribute is ignored when the service is invoked from an API, then only the API requireAuth attribute is relevant.

Requires that users specify security credentials to be authenticated in order to access all API requesters. If the requireAuth attribute is not set, the global setting on the zosconnect\_zosConnectManager element is used instead, unless the requireAuth attribute is overridden on the specific API requester.

Requires that users specify security credentials to be authenticated in order to access the service.

**The requireAuth attribute controls whether an inbound request must provide credentials**

# Authentication/Authorization Precedence

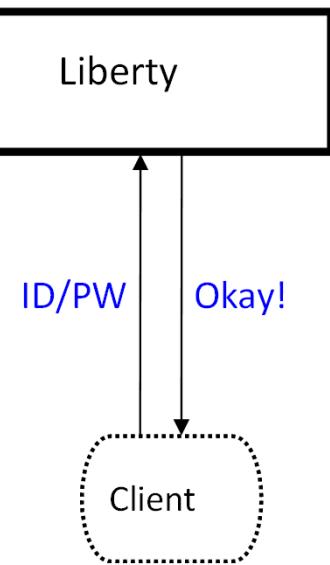


# Authentication - Basic Authentication



Several different ways this can be accomplished:

## Basic Authentication

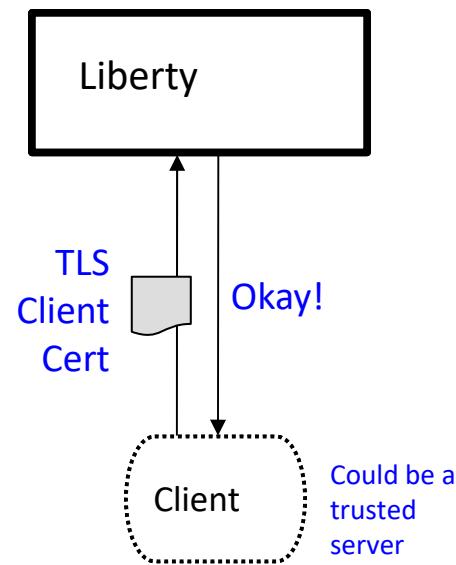


**Client supplies an identity/password or an identity/PassTicket**

**Server checks registry:**

- **Basic (server.xml)**
- **SAF**

## Client Certificate



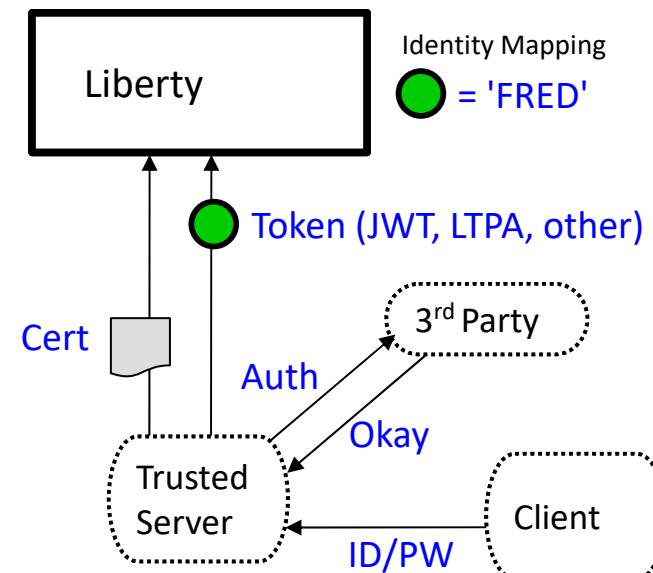
**Client supplies client certificate**

**Server validates client certificate and maps it to an identity**

**Registry options:**

- **SAF**

## Third Party Authentication



**Client authenticates to 3<sup>rd</sup> party sever**

**Client receives a trusted 3<sup>rd</sup> party token**

**Token flows to server and is mapped to an identity**

**Registry options:**

- **We do not need to know the details.**

# Basic authentication – REST Client



- ❑ server XML security configuration:

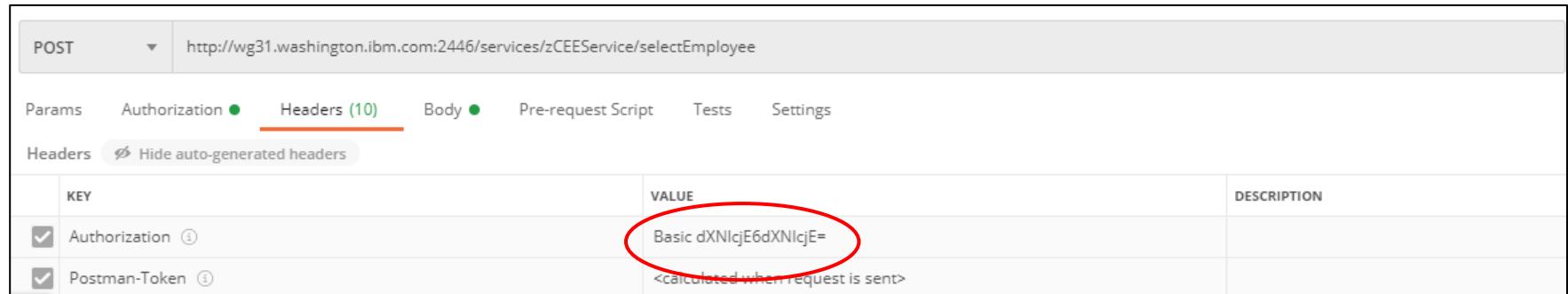
```
<featureManager>
    <feature>appSecurity-2.0</feature>
    <feature>zosSecurity-1.0</feature>
</featureManager>

<webAppSecurity allowFailOverToBasicAuth="true" />

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

Note that these are Liberty configuration elements documented in the Liberty KC, i.e., no `zosconnect` prefix.

- ❑ When sending a request to a Liberty server running z/OS Connect, basic authentication information (identity and password) is provided in the HTTP header in a *Basic Authorization* token with the identity and password encoded or formatted using Base64.
  - An example with Postman:



The screenshot shows a Postman request configuration. The method is set to POST, and the URL is `http://wg31.washington.ibm.com:2446/services/zCEEService/selectEmployee`. The 'Headers' tab is selected, showing two entries: 'Authorization' and 'Postman-Token'. The 'Authorization' header has its value circled in red, displaying the encoded string `Basic dXNlcjE6dXNlcjE=`. A note below the value states '`<calculated when request is sent>`'.

- An example with cURL:

```
curl -X GET --user fred:fredpwd --header "Content-Type: application/json ...
```

# Basic Authentication – identity and password



The screenshot shows an IDE interface with the following details:

- Code Editor:** The file "ZceeGet.java" is open, showing Java code for making an HTTP GET request to a URL. A red circle highlights the line where the "Authorization" header is set to "Fred:fredpwd".
- Console Output:** The output window shows the command run: "<terminated> ZCEEGet [Java Application] C:\Program Files\IBM\Java80\jre\bin\javaw.exe (Apr 27, 2021, 4:08:52 PM)" and the URL: "URL: https://wg31.washington.ibm.com:9453/db2/department?dept1=C01&dept2=C01".
- Execution Results:** The output displays three sets of employee details, each starting with "Employee Number:" followed by a unique ID (000030, 000130, 000140) and then listing First Name, Last Name, Middle Initial, Phone Number, and Department.

```
49     URL url = new URL("https://wg31.washington.ibm.com:9453/db2/department?dept1=C01&dept2=C01");
50     System.out.println("URL: " + url);
51    HttpsURLConnection conn = (HttpsURLConnection) url.openConnection();
52     conn.setRequestMethod("GET");
53     conn.setRequestProperty("Content-Type", "application/json");
54     byte[] bytesEncoded = Base64.encodeBase64("Fred:fredpwd".getBytes());
55     conn.addRequestProperty("Authorization", new String(bytesEncoded));
56
57     try {
58
59         if (conn.getResponseCode() != 200) {
60             throw new RuntimeException("Failed : HTTP error code : " + conn.getResponseCode());
61         }
62         BufferedReader bufferReader = new BufferedReader(new InputStreamReader((conn.getInputStream())));
63         String output;
64         StringBuilder stringBuffer = new StringBuilder();
65         while ((output = bufferReader.readLine()) != null) {
66             stringBuffer.append(output);
67         }
68         JSONObject json = new JSONObject(stringBuffer.toString());
69         JSONArray jsonArray = json.getJSONArray("ResultSet 1 Output");
70         JSONObject jsonEntry = new JSONObject();
71         for (int index = 0; index < jsonArray.length(); index++) {
72             jsonEntry = jsonArray.getJSONObject(index);
73             if (jsonEntry.has("employeeNumber")){

```

```
Employee Number: 000030
First Name : SALLY
Last Name: KWAN
Middle Initial: A
Phone Number: 4738
Department: C01

Employee Number: 000130
First Name : DOLORES
Last Name: QUINTANA
Middle Initial: M
Phone Number: 4578
Department: C01

Employee Number: 000140
First Name : HEATHER
Last Name: NICHOLLS
Middle Initial: A
Phone Number: 1793
Department: C01

Employee Number: 200140
First Name : KIM
Last Name: NATZ
Middle Initial: N
Phone Number: 1793
Department: C01

StatusCode: 200
StatusDescription: Execution Successful
```

# Basic authentication – COBOL API Requester



- ❑ A MVS batch or IMS 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")
```

Note the communications stub generates the Authentication header token

- ❑ 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**

© 2017, 2021 IBM Corporation

mitchj@us.ibm.com

# 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.*

# Tech/Tip: RACF resources for using PassTickets



- ❑ First define a PTKTDATA resource using the *appName* assigned to the target subsystem:

```
RDEFINE PTKTDATA appName SSIGNON(KEYMASK(keymaskValue))  
          APPLDATA('NO REPLAY PROTECTION')
```

Where:

*appName* is an application name assigned to the resource, e.g., BBGZDFLT  
*keymaskValue* is the value of the secured sign-on application key, a 64-bit hex value  
*replayProtection* indicates if a pass ticket can be reused

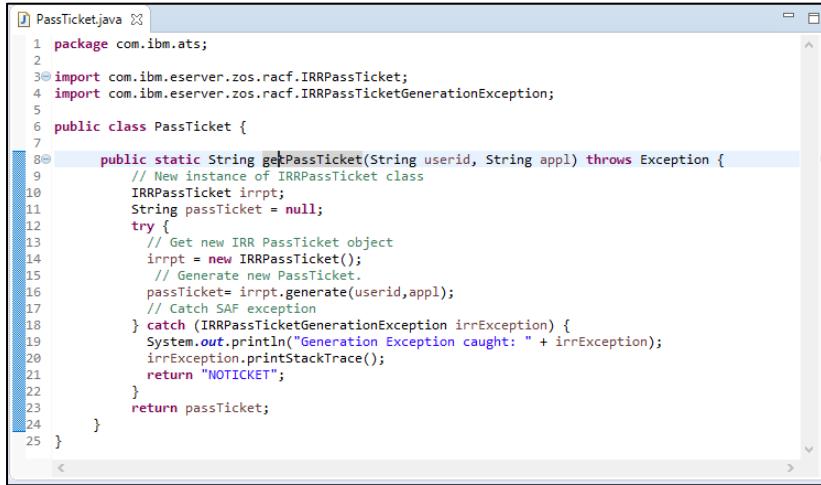
- ❑ Access to using PassTickets is controlled by another PTKTDATA resource, *IRRPTAUTH.appName.identity*. UPDATE access is required. For example, to use PassTickets to access a z/OS Connect server the resources below need to be defined and access permitted.

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

```
RDEFINE PTKTDATA BBGZDFLT SSIGNON(0123456789ABCDEF) )  
          APPLDATA('NO REPLAY PROTECTION') UACC(NONE)  
RDEFINE PTKTDATA IRRPTAUTH.BBGZDFLT.* UACC(NONE)  
PERMIT IRRPTAUTH.BBGZDFLT.* ID(LIBSERV) CLASS(PTKTDATA) ACCESS(UPDATE)  
PERMIT IRRPTAUTH.BBGZDFLT.USER1 ID(USER1) CLASS(PTKTDATA) ACCESS(UPDATE)
```

# Tech/Tip: Generating PassTickets on z/OS

- On z/OS, a COBOL user application can generate PassTickets by
  - invoking the IBM provided class *com.ibm.eserver.zos.racf.IRRPassTicket*:



```
PassTicket.java |  
1 package com.ibm.ats;  
2  
3@ import com.ibm.eserver.zos.racf.IRRPassTicket;  
4 import com.ibm.eserver.zos.racf.IRRPassTicketGenerationException;  
5  
6 public class PassTicket {  
7  
8@     public static String getPassTicket(String userid, String appl) throws Exception {  
9         // New instance of IRRPassTicket class  
10        IRRPassTicket irppt;  
11        String passTicket = null;  
12        try {  
13            // Get new IRR PassTicket object  
14            irppt = new IRRPassTicket();  
15            // Generate new PassTicket.  
16            passTicket= irppt.generate(userid,appl);  
17            // Catch SAF exception  
18        } catch (IRRPassTicketGenerationException irrException) {  
19            System.out.println("Generation Exception caught: " + irrException);  
20            irrException.printStackTrace();  
21            return "NOTICKET";  
22        }  
23    }  
24    return passTicket;  
25 }
```

```
*-----  
* Invoke method getPassTicket in Java class  
*      com.ibm.ats.PassTicket  
*-----  
*-----  
Invoke PassTicketClass "getPassTicket"  
using by value jIdentity jAppl returning jResponse.  
Perform JavaExceptionCheck
```

- calling RACF service IRRSPK00:

```
CALL W-IRRSPK00 USING irr-workarea,  
IRR-ALET, irr-safrc,  
IRR-ALET, irr-racfrc,  
IRR-ALET, irr-racfrsn,  
IRR-ALET, irr-functionCode,  
irr-optionWord,  
IRR-PASSTICKET,  
irr-ticketOptions-ptr,  
IRR-IDENTITY,  
IRR-APPLID.
```

# Tech/Tip: Generating PassTickets on z/OS

- On non z/OS systems, a PassTicket can be generated in Java and other means

The screenshot shows an IDE interface with a Java code editor and a terminal window.

**Java Code (Main.java):**

```
public static void main(String[] args) {
    try {
        //RDEFINE PTKTDATA BBGZDFLT SSIGNON(KEYMASK(123456789ABCDEF0)) +
        // APPLDATA('NO REPLAY PROTECTION')
        String identity = null;
        String passTicket = null;
        String applName = "BBGZDFLT";
        String keyMask = "123456789ABCDEF0";
        PassTicketGenerator pt = new PassTicketGenerator();
        if (args.length == 0) {
            System.out.println("\nRACF identity required");
            return;
        }
        if (args.length >= 1) {
            identity = args[0].toUpperCase();
        }
        if (args.length >= 2) {
            applName = args[1];
        }
        if (args.length >= 3) {
            keyMask = args[2];
        }
        passTicket = pt.generate(identity, applName, keyMask);
        System.out.print(identity+":"+passTicket);
    } catch (PassTicketException | PassTicketGeneratorException e) {
        System.out.println(e.getLocalizedMessage());
    }
}
```

**Terminal Output:**

```
Microsoft Windows [Version 10.0.19041.928]
(c) Microsoft Corporation. All rights reserved.

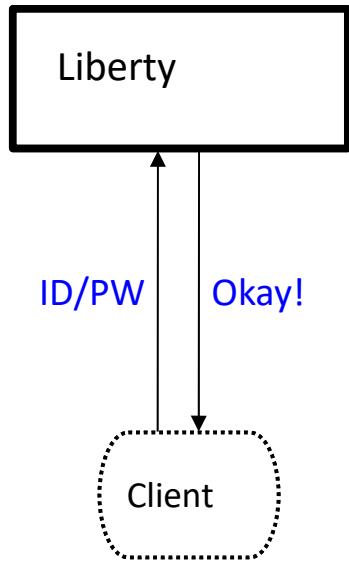
c:\z>cd admin
c:\z\admin>curl -X POST --user USER1:B2LT07TS --header "Content-Type: application/json" -d @inquireSingle.json --insecure https://wg31.washington.ibm.com:9443/zosConnect/services/inquireSingle?action=invoke -w '%{http_code}'
<terminated> Main (1) [Java Application] C:\Program Files\IBM\Java
USER1:B2LT07TS
{"DFH0XCP1":{"CA_RESPONSE_MESSAGE":"RETURNED ITEM: REF =0020","CA_INQUIRE_SINGLE":{"CA_SINGLE_ITEM":{"CA_SNGL_ITEM_REF":20,"CA_SNGL_DESCRIPTION":"Ball Pens Blue 24pk","CA_SNGL_DEPARTMENT":10,"IN_SNGL_STOCK":6,"CA_SNGL_COST":"002.90","ON_SNGL_ORDER":50}},"CA_RETURN_CODE":0}} 200
c:\z\admin>curl -X POST --user USER1:B2LT07TS --header "Content-Type: application/json" -d @inquireSingle.json --insecure https://wg31.washington.ibm.com:9443/zosConnect/services/inquireSingle?action=invoke -w '%{http_code}'
401
c:\z\admin>
```

# Authentication - TLS Mutual Authentication



Several different ways this can be accomplished:

## Basic Authentication



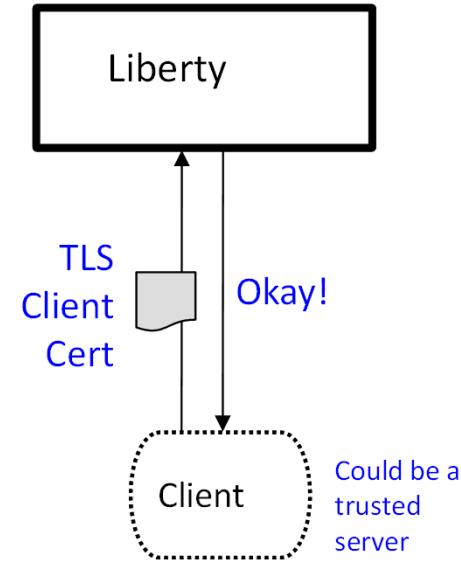
Server prompts for ID/PW

Client supplies ID/PW or ID/PassTicket

Server checks registry:

- Basic (server.xml)
- SAF

## Client Certificate



Server prompts for client certificate.

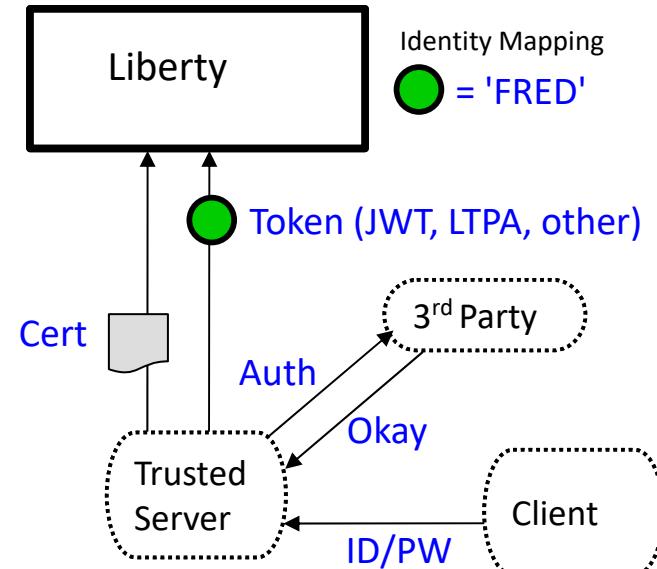
Client supplies personal certificate

Server validates client certificate and maps it to an identity

Registry options:

- SAF

## Third Party Authentication



Client authenticates to 3<sup>rd</sup> party sever

Client receives a trusted 3<sup>rd</sup> party token

Token flows to Liberty z/OS and is mapped to an identity

Registry options:

- We may not need to know these details.



# Using this Liberty JSSE server XML configuration

```
<!-- Enable features -->
<featureManager>
    <feature>transportSecurity-1.0</feature>
</featureManager>

<sslDefault sslRef="DefaultSSLSettings"
    outboundSSLRef="OutboundSSLSettings" />

<ssl id="DefaultSSLSettings"
    keyStoreRef="CellDefaultKeyStore"
    trustStoreRef="CellDefaultKeyStore"
    clientAuthenticationSupported="true"
    clientAuthentication="true"/>

<keyStore id="CellDefaultKeyStore"
    location="safkeyring:///Liberty.KeyRing"
    password="password" type="JCERACFKS"
    fileBased="false" readOnly="true" />

<ssl id="OutboundSSLSettings"
    keyStoreRef="OutboundKeyStore"
    trustStoreRef="OutboundKeyStore"/>

<keyStore id="OutboundKeyStore"
    location="safkeyring:///zCEE.KeyRing"
    password="password" type="JCERACFKS"
    fileBased="false" readOnly="true" />
```

SSL repertoires

Key ring for server certificate  
for client connections

Key ring for client certificate  
for server connections

Tech/Tip: Regarding *clientAuthentication* and *clientAuthenticationSupported*. Understand the implications of the interactions between these attributes. There may instances where you want to use HTTPS, but not always with mutual authentication Consider setting *clientAuthentication* to false when setting *clientAuthenticationSupported* to true.

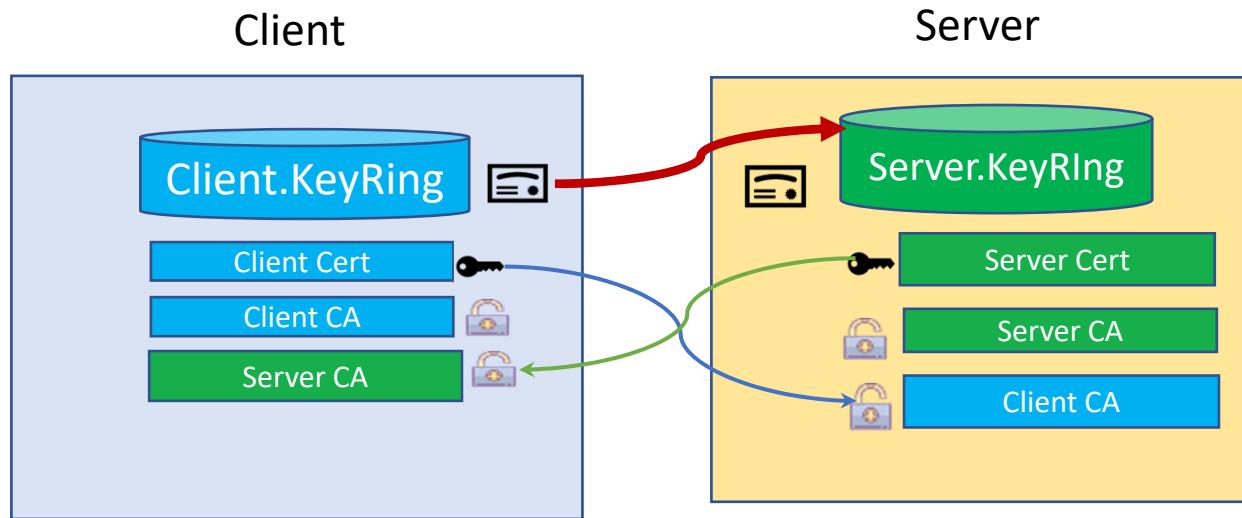
F BAQSTRT,REFRESH,KEYSTORE

# Let's explore the basic TLS Handshake Flow

TLS handshake –

Server Authentication

Mutual Authentication (optional)



safkeyring://KeyRing v safkeyring://owner/KeyRing

RACF FACILITY resources

- IRR.DIGTCERT.LISTRING
  - READ to list your own key ring
  - UPDATE to list another user's key ring
- IRR.DIGTCERT.LIST
  - READ to list your own certificate
  - UPDATE to list another user's certificate
  - CONTROL to list SITE of CERTAUTH certificates

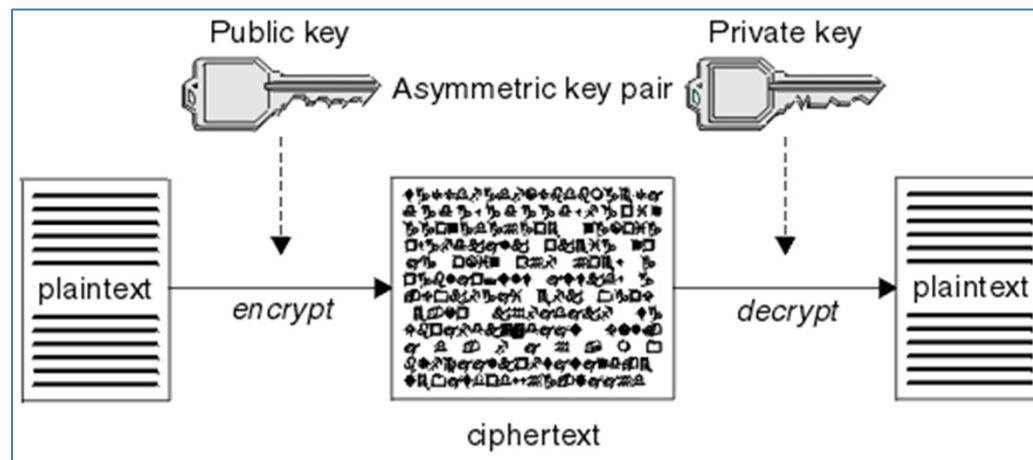
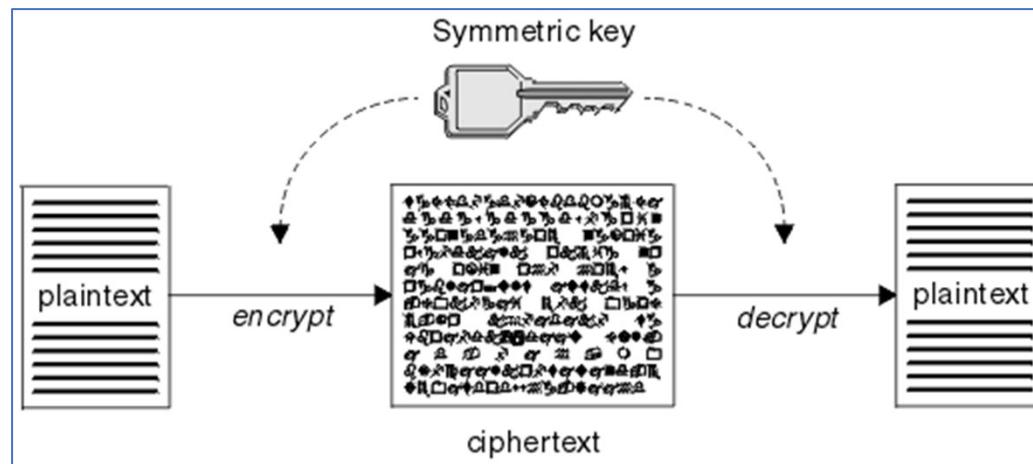
Key icon: Certificate with a private key\*

Lock icon: Certificate Authority (CA) certificate chain#

\*For server and/or mutual authentication to work, the endpoint sending its server or client certificate must use a personal certificate with a private key. The private key is required to decrypt (or encrypt) a message digest that is sent from the other endpoint during the handshake flow. Generation of a message digest also requires access to the CA certificate used to sign the certificate.

#Refers to the set or of certificates used to issue the server or client personal certificate including any intermediate certificates all the way to the root CA.

# Symmetric key v. Asymmetric key pair



# Tech/Tip: cURL trace showing mutual authentication

- \* successfully set certificate verify locations:
- \* TLSv1.3 (OUT), TLS handshake, Client hello (01):
- \* TLSv1.3 (IN), TLS handshake, Server hello (02):
- \* TLSv1.2 (IN), TLS handshake, Certificate (11):
- \* TLSv1.2 (IN), TLS handshake, Server key exchange (12):
- \* **TLSv1.2 (IN), TLS handshake, Request CERT (13):**
- \* TLSv1.2 (IN), TLS handshake, Server finished (14):
- \* **TLSv1.2 (OUT), TLS handshake, Certificate (11):**
- \* TLSv1.2 (OUT), TLS handshake, Client key exchange (16):
- \* **TLSv1.2 (OUT), TLS handshake, CERT verify (15):**
- \* TLSv1.2 (OUT), TLS change cipher, Change cipher spec (01):
- \* TLSv1.2 (OUT), TLS handshake, Finished (20):
- \* TLSv1.2 (IN), TLS handshake, Finished (20):
- \* SSL connection using TLSv1.2 / ECDHE-RSA-AES256-GCM-SHA384
- \* Server certificate:
- \* subject: O=IBM; OU=LIBERTY; CN=wg31.washington.ibm.com
- \* start date: Jan 4 04:00:00 2021 GMT
- \* expire date: Jan 1 03:59:59 2023 GMT
- \* common name: wg31.washington.ibm.com (matched)
- \* issuer: OU=LIBERTY; CN=CA for Liberty
- \* SSL certificate verify ok.

TLS 1.2 <https://tools.ietf.org/html/rfc5246>

TLS 1.3 <https://tools.ietf.org/html/rfc8446>

# Tech/Tip: RACF digital certificate (RACDCERT) command review

```
RACDCERT CERTAUTH GENCERT SUBJECTSDN(CN('Liberty CA') +
    OU('LIBERTY')) WITHLABEL('Liberty CA') TRUST SIZE(2048) NOTAFTER(DATE(2022/12/31))
RACDCERT ID(LIBSERV) GENCERT SUBJECTSDN(CN('wg31.washington.ibm.com') +
    O('IBM') OU('LIBERTY')) WITHLABEL('Liberty Client Cert') +
    ALTNAME(DOMAIN('wg31z.washington.ibm.com')) SIGNWITH(CERTAUTH LABEL('Liberty CA')) SIZE(2048) +
    NOTAFTER(DATE(2022/12/31))
```

```
RACDCERT ID(LIBSERV) GENCERT SUBJECTSDN(CN('wg31.washington.ibm.com') +
    O('IBM') OU('LIBERTY')) WITHLABEL('Liberty Server Cert')
RACDCERT ID(LIBSERV) GENREQ(LABEL('Liberty Server Cert')) DSN(CERT.REQ)
```

Send the certificate to your Certificate Authority to be signed

```
racdcert CERTAUTH withlabel('Liberty CA') add('USER1.LIBCA.PEM') TRUST
racdcert id(LIBSERV) withlabel('Liberty Server Cert') add('LIBSERV.P12') password('secret') TRUST
```

```
/* Create Liberty key ring and connect CA and personal certificates */
racdcert id(libserv) addring(Liberty.KeyRing)
racdcert id(libserv) connect(ring(Liberty.KeyRing) label('CICS CA') certauth usage(certauth))
racdcert id(libserv) connect(ring(Liberty.KeyRing) label('Liberty CA') certauth usage(certauth))
/* Connect default personal certificate */
racdcert id(libserv) connect(ring(Liberty.KeyRing) label('Liberty Client Cert') default
```

```
setropts raclist(digtcert) refresh
```

## Broadcom Support web pages

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

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

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

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

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

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

© 2017, 2021 IBM Corporation

mitchj@us.ibm.com

# Tech/Tip: RACF Certificate Filtering and Mapping

Filters for mapping certificates can be created with a RACDCERT command.

- Enter command RACDCERT ID MAP to create a filter that assigns RACF identity ATSUSER to any digital certificate signed with the ATS client signer certificate and where the subject is organizational unit ATS in organization IBM.

```
racdcert id(atsuser) map sdnfilter('OU=ATS.O=IBM')
idnfilter('CN=ATS Client CA.OU=ATS.O=IBM') withlabel('ATS USERS')
```

- Enter command RACDCERT ID MAP to create a filter that assigns RACF identity OTHUSER to any digital certificate signed by the ATS client signer certificate and where the subject is in organization IBM.

```
racdcert id(othuser) map sdnfilter('O=IBM')
idnfilter('O=IBM') withlabel('IBM USERS')
```

- Refresh the in-storage profiles for digital certificate maps.

```
SETRPTS RACLIST(DIGTNMAP) REFRESH
```

# Tech/Tip: Anatomy of a RACF Personal Digital Certificate

Digital certificate information for user ATSSERV:

Label: **RPServer-Server**

Certificate ID: 2Qfb4+Lixdn12dfihZmlhZlg4oWZpYWZ

Status: **TRUST**

Start Date: 2020/11/12 00:00:00

End Date: **2029/12/31 23:59:59**

Serial Number:

>01<

Issuer's Name:

>**CN=RPServer-CertAuth.OU=CertAuth**<

Subject's Name:

>**CN=RPServer-Server.OU=ATS.O=IBM.C=USA**<

Subject's AltNames:

Domain: **wg31.washington.ibm.com**

Signing Algorithm: sha1RSA

Key Type: RSA

Key Size: 2048

Private Key: **YES**

Ring Associations:

**Ring Owner: ATSSERV**

**Ring:**

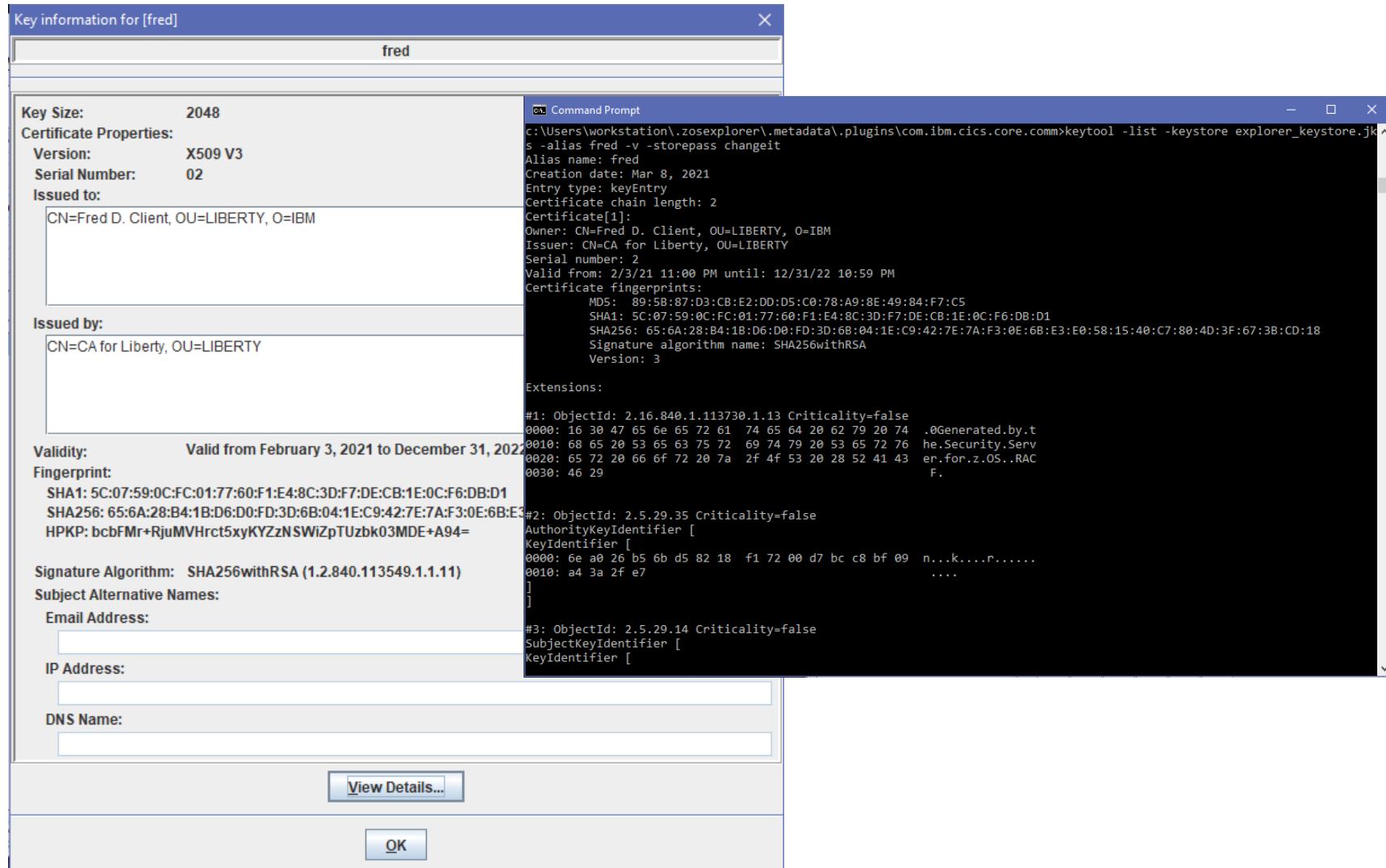
>**RpServer.KeyRing**<

**Ring Owner: LIBSERV**

**Ring:**

>**RpServer.KeyRing**<

# Tech/Tip: Anatomy of a certificate – IkeyMan/keytool

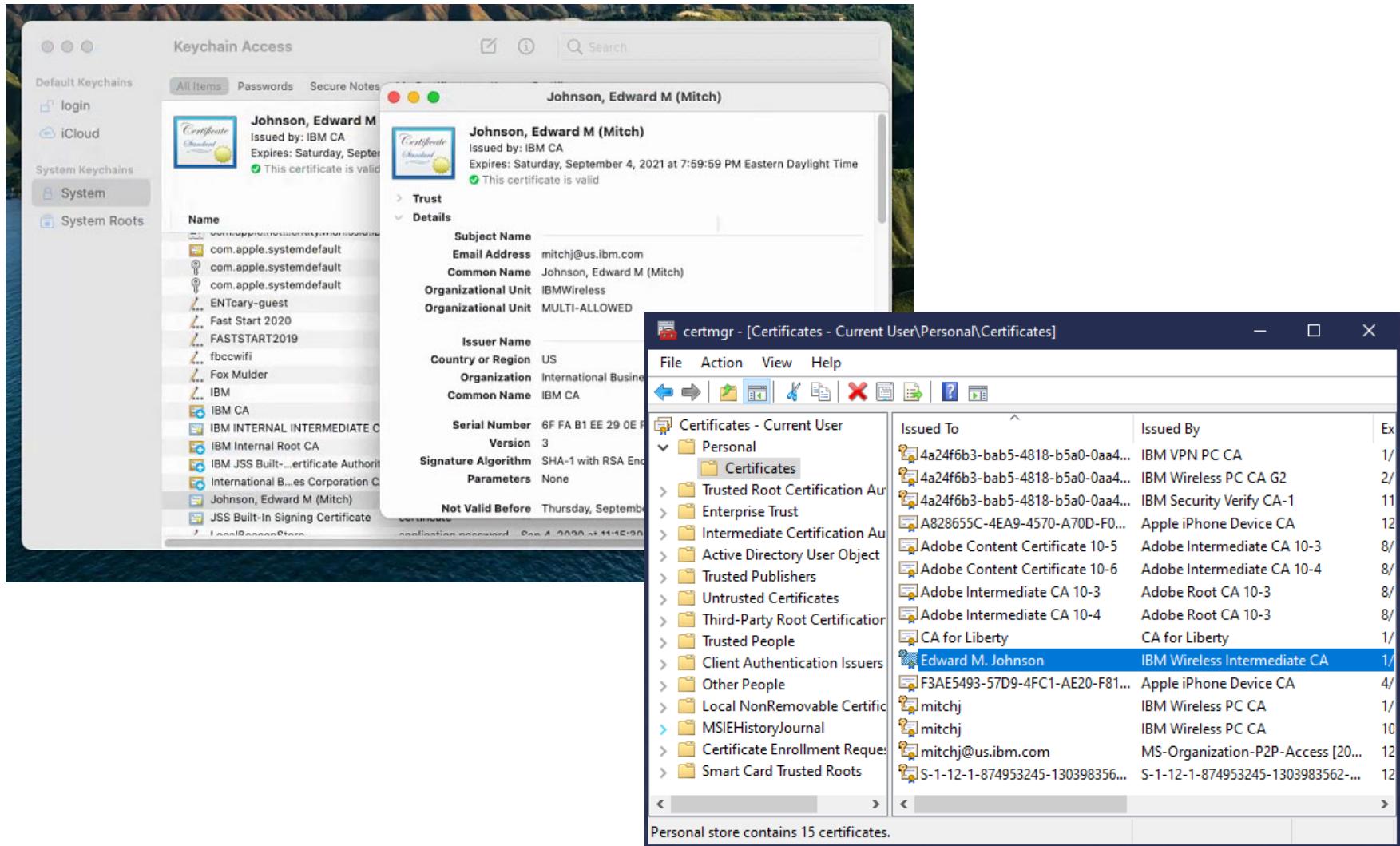


Both can be found in /usr/lpp/java/J8.0\_64/bin or c:/Program Files/IBM/Java80/jre/bin

© 2017, 2021 IBM Corporation

mitchj@us.ibm.com

# Tech/Tip: Anatomy of a certificate – Mac/Windows



# Tech/Tip: How to tell the difference?

Public certificate (site or certificate authority or certificate request)

```

CERTAUTH.PEM - Notepad
File Edit Format View Help
--BEGIN CERTIFICATE--
MIIDVjCCAj6gAwIBAgIBADANBgkqhkiG9w0BAQsFADArMRAwDgYDVQQLEwdMSUJF
U1RZMrcwFQYDVQDEw5DQSbm3IgTl1ZXJ0eTAeFw0yMDeyMTUwNDwMDBaFw0y
MzAxMDExMzU5NTlaMsxEDA0BgNVBAsTB0xJQkVSFkxZAVBGNVBAMTDkNBIGZv
ciBMaWJ1cnR5MIIBIjANBgkqhkiG9w0BAQEAAOCQ8AMIIBCgKCAQEAC7jIPa9
WuiPN1pC1a4TtwmgZM09haCitAnjnFYn10RqbhZxBKuegu+2mjE6axmRSmj7pI
kLSEK8fIZ683avEQTZYTesiow+9c0vMc61bCvok/F0FhzmTNbRmhk16jMXFeE8
RJwVxhtVgUY24J0AxG0oPI7cYoeBYAz39T+hng+4Ip03JK1IxSJUcYrgJ0W2YTV
yaL8uJbS7nhx30fhPVdpoxN5hdVmV9tM/v6644/jBKMo03jBwNhQ9f1onCsSc+
oB+a04NHVJ6dIc9Z+j95ZpIuXMYwPx8jdhRzAIR3bmJFpy02xTTckREGf927
gYWOYEc8+b1nQIDAQABo4GEMIGBD8GCWCGSAGG+EIBDQyfjBHZW51cmF0ZwQg
YnkgdGh1IFN1Y3VyaR51Fn1cnZlciBmb3Igei9PUyAuKFDRIkwDgYDVR0PAQH/
BAQDAgEGMA8GA1UdEwEB/wQFMABAf8wHQYDV00BBYEJZVpYVPqeQWqhYY2v1+
T1uzksHEMA0GCSqGSIB3DQEBCwUAA4IBAQAEPhqS75ZYp8DWnyX5I0rLdtFU36bX
4RgttX3V1+H/jSh01Wfvuxp7TguD30v3+7dtFUR5Diup60pKmaEE9v8fbqA6oFD
N6EABN6v18QUAV9cAWSPp0mPCvwre0aKJDerryEtixiX9aGUKE1Vqk1KK4ThV/8
RFw4fb8p2QHPOTj8s270eG1w0DtDwqFdbSCfmJdLUaqvIxLezPdhC+CX3t+0
iJjTSzILLcBtC6Ainycsw/U4/e0x19m0AvjKRgYm14rfC1fj8G2jhk6vxhJBxCRB
BD9wVZ9ZiuiA0NugkpzGR03HwcBYY0idkAF0ZkeVH0t1SMRzbG6q55b1
-----END CERTIFICATE-----

```

Personal certificate with private key

```

USER1.P12 - Notepad
File Edit Format View Help
-----BEGIN PRIVATE KEY-----
-----END PRIVATE KEY-----
-----BEGIN CERTIFICATE-----
-----END CERTIFICATE-----

```

```

WG31#- 3270
File Edit Settings View Communication Actions Window Help
Menu Utilities Compilers Help
BROWSE USER1.CERTAUTH.PEM Line 0000000000 Col 001 080
Command ==> ***** Top of Data *****
-----BEGIN CERTIFICATE-----
-----END CERTIFICATE-----
***** Bottom of Data *****
-----BEGIN CERTIFICATE-----
-----END CERTIFICATE-----

```

```

WG31#- 3270
File Edit Settings View Communication Actions Window Help
Menu Utilities Compilers Help
BROWSE USER1.CICS.P12 Line 0000000000 Col 001 080
Command ==> ***** Top of Data *****
-----BEGIN PRIVATE KEY-----
-----END PRIVATE KEY-----
-----BEGIN CERTIFICATE-----
-----END CERTIFICATE-----

```

# Tech/Tip: Combining TLS mutual and basic authentication

```
//*****
//* SET SYMBOLS
//*****
//EXPORT EXPORT SYMLIST=(*)
// SET CURL= '/usr/lpp/rocket/curl'
//*****
//* CURL Procedure
//*****
//CURL PROC
//CURL EXEC PGM=IKJEFT01,REGION=0M
//SYSTSPRT DD SYSOUT=*
//SYSERR DD SYSOUT=*
//STDOUT DD SYSOUT=*
// PEND
//*****
//* STEP CURL - use curl to deploy API cscvinc
//*****
//DEPLOY EXEC CURL
BPXBATCH SH export CURL=&CURL; +
$CURL/bin/curl -X PUT -s +
--cacert /u/johnson/CERTAUTH.PEM --user FRED:FRED +
https://wg31.washington.ibm.com:9445/zosConnect/apis/cscvinc?status=sto+
pped > null; +
$CURL/bin/curl -X DELETE -s +
--cacert /u/johnson/CERTAUTH.PEM --user FRED:FRED +
https://wg31.washington.ibm.com:9445/zosConnect/apis/cscvinc > null; +
$CURL/bin/curl -X POST -s +
--cacert /u/johnson/CERTAUTH.PEM --user FRED:FRED +
--data-binary @/u/johnson/cscvinc.aar +
--header "Content-Type: application/zip" +
https://wg31.washington.ibm.com:9445/zosConnect/apis
//*****
//* STEP CURL - use curl to invoke the API cscvinc
//*****
//INVOKE EXEC CURL
//SYSTSIN DD *,SYMBOLS=EXECSYS
BPXBATCH SH export CURL=&CURL; $CURL/bin/curl -X GET -s +
--cacert /u/johnson/CERTAUTH.PEM --user FRED:FRED +
https://wg31.washington.ibm.com:9445/cscvinc/employee/000100
```

```
<httpEndpoint id="defaultHttpEndpoint"
  host="*"
  httpPort="9080"
  httpsPort="9443" />

<sslDefault sslRef="DefaultSSLSettings"
  outboundSSLRef="DefaultSSLSettings" />

<ssl id="DefaultSSLSettings"
  keyStoreRef="CellDefaultKeyStore"
  trustStoreRef="CellDefaultKeyStore"
  clientAuthenticationSupported="true"
  clientAuthentication="true"/>

<keyStore id="CellDefaultKeyStore"
  location="safkeyring:///Liberty.KeyRing"
  password="password" type="JCERACFKS"
  fileBased="false" readOnly="true" />
```

```
<httpEndpoint id="AdminHttpEndpoint"
  host="*"
  httpPort="-1"
  httpsPort="9445"
  sslOptionsRef="mySSLOptions"/>

<ssLOptions id="mySSLOptions"
  sslRef="BatchSSLSettings"/>

<ssl id="BatchSSLSettings"
  keyStoreRef="CellDefaultKeyStore"
  trustStoreRef="CellDefaultKeyStore"
  clientAuthenticationSupported="true"
  clientAuthentication="false"/>
```

<https://www.rocketsoftware.com/platforms/ibm-z/curl-for-zos>

# Tech/Tip: Combining TLS and pass tickets authentication

```
//EXPORT EXPORT SYMLIST=(*)
// SET CURL='/usr/lpp/rocket/curl'
// SET USERCRED='FRED:Z0HWKDSF'
// SET CACERT='/u/johnson/zceesrvr.pem'
//IKJEFT01 PROC
//IKJEFT01 EXEC PGM=IKJEFT01,REGION=0M
//SYSTSPRT DD SYSOUT=*
//SYSERR DD SYSOUT=*
//STDOUT DD SYSOUT=*
// PEND
//*****
//** Copy a CA certificate to OMVS
//*****
//COPYPEM EXEC IKJEFT01
//INDD DD DISP=SHR,DSN=USER1.CERTAUTH.PEM
//OUTDD DD PATH='&CACERT',PATHOPTS=(ORDWR,OCREATE)
//SYSTSIN DD *,SYMBOLS=EXECSYS
OCOPY INDD(INDD) OUTDD(OUTDD) FROM1047 CONVERT((BPXFX311))
//*****
//** Use cURL to invoke an administrative API
//*****
//INVOKE EXEC IKJEFT01
//SYSTSIN DD *,SYMBOLS=EXECSYS
BPXBATCH SH export CURL=&CURL; +
$CURL/bin/curl -X GET +
--cacert &CACERT --user &USERCRED +
https://wg31.washington.ibm.com:9445/zosConnect/apis
//*****
//** Use cURL to invoke an administrative API
//*****
//INVOKE EXEC IKJEFT01
//SYSTSIN DD *,SYMBOLS=EXECSYS
BPXBATCH SH export CURL=&CURL; +
$CURL/bin/curl -X GET +
--cacert &CACERT --user &USERCRED +
https://wg31.washington.ibm.com:9445/cscvinc/employee/000100
```

Consider using a program that will customize template JCL and then submit it for execution

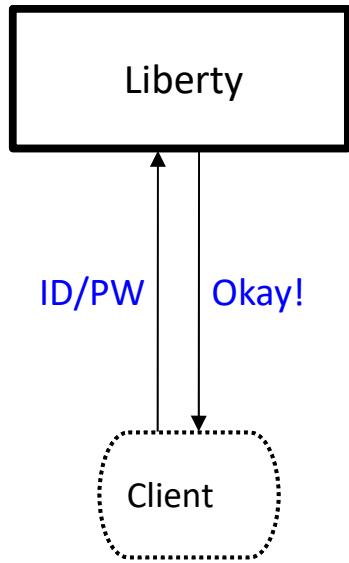
```
//BUILDJCL EXEC PGM=BUILDJCL,REGION=0M
//SYSPRINT DD SYSOUT=*
//INJCL DD DISP=SHR,DSN=JOHNSON.JCLLIB.CNTL(CURLSKEL)
//INTRDR DD SYSOUT=(A,INTRDR)
```

# Authentication - Third Party Authentication



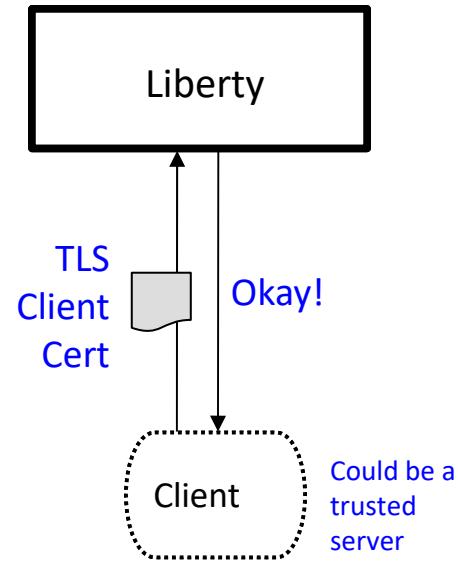
Several different ways this can be accomplished:

## Basic Authentication



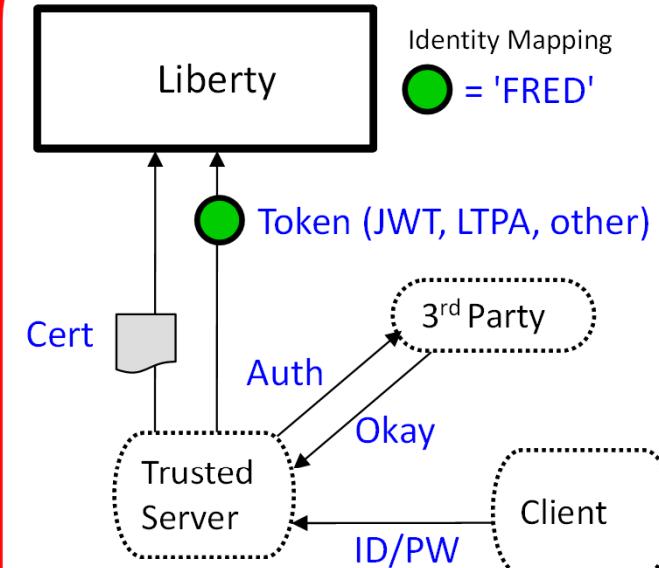
Server prompts for ID/PW  
Client supplies ID/PW or ID/PassTicket  
Server checks registry:  
• Basic (server.xml)  
• SAF

## Client Certificate



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

## Third Party Authentication



**Client authenticates to 3<sup>rd</sup> party sever  
Client receives a trusted 3<sup>rd</sup> party token  
Token flows to Liberty z/OS and is mapped to an identity  
Registry options:  
• We may know these detail.**

# Third Party Authentication Examples



Screenshot of the UPS Sign Up page (<https://wwwapps.ups.com/doapp/signups>):

The page displays a yellow banner stating "UPS is open for business: Service impacts related to Coronavirus ...More". It features the UPS logo and links for "Sign up / Log in" and "Search or Track". Below this, there's a heading "Sign Up" and a note "Already have an ID? [Log in](#)". A section titled "Use one of these sites." offers social login options: Google, Facebook, Amazon, Apple, and Twitter. A "Feedback" button is located on the right side. The main form includes fields for Name\*, Email\*, User ID\*, Password\*, and Phone (with a dropdown for US +1).

Screenshot of the myNCDMV Log In page (<https://payments.ncdot.gov/auth>):

The page has a background image of autumn foliage. It features "Log In" and "Sign Up" buttons. The "Log In" section contains fields for "Email Address" (with placeholder "name@example.com") and "Password" (with a "Show Password" link). There's also a "Remember Me" checkbox and "Log In" and "Forgot Password" buttons. Below this, there's a "Continue as Guest" button. A "NOTICE FOR PUBLIC COMPUTER USERS" message states: "If you sign in with Google, Apple, or Facebook you are also signing into that account on this computer. Remember to sign out when you're done." The page is powered by [payit](#).



# Security token types by Liberty

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

# 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 website interface. At the top, there's a navigation bar with the JUUT logo, Debugger, Libraries, Introduction, Ask, Get a T-shirt!, and a Crafted by Auth0 link. Below the navigation, there's a dropdown for 'Algorithm' set to 'RS256'. The main area is divided into 'Encoded' and 'Decoded' sections. The 'Encoded' section contains a long string of characters. The 'Decoded' section is split into 'HEADER:' and 'PAYLOAD:' sections. The HEADER content is:

```
{  
  "kid": "4qjX-  
bkXOUw_F_uccjRMkB9ivMjXSQwj0RrkyRJq8DM",  
  "alg": "RS256"  
}
```

The PAYLOAD content is:

```
{  
  "sub": "Fred",  
  "token_type": "Bearer",  
  "scope": [  
    "openid",  
    "profile",  
    "email"  
  ],  
  "azp": "rpSsl",  
  "iss": "https://wg31.washington.ibm.com:26213  
/oidc/endpoint/OP",  
  "aud": "myZcee",  
  "exp": 1604333158,  
  "iat": 16043330858,  
  "realmName": "zCEERealm",  
  "uniqueSecurityName": "Fred"  
}
```

A red oval highlights the timestamp 'Mon Nov 02 2020 11:05:58 GMT-0500 (Eastern Standard Time)' in the PAYLOAD section.

Values derived from the OAUTH configuration:

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

<https://jwt.io>

© 2017, 2021 IBM Corporation

# 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

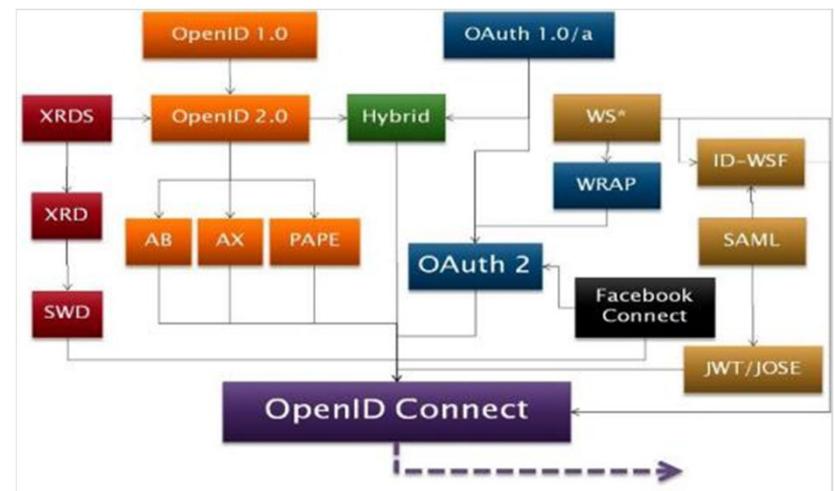
See the YouTube videos:

*OAuth 2.0 and OpenID Connect (in plain English)*

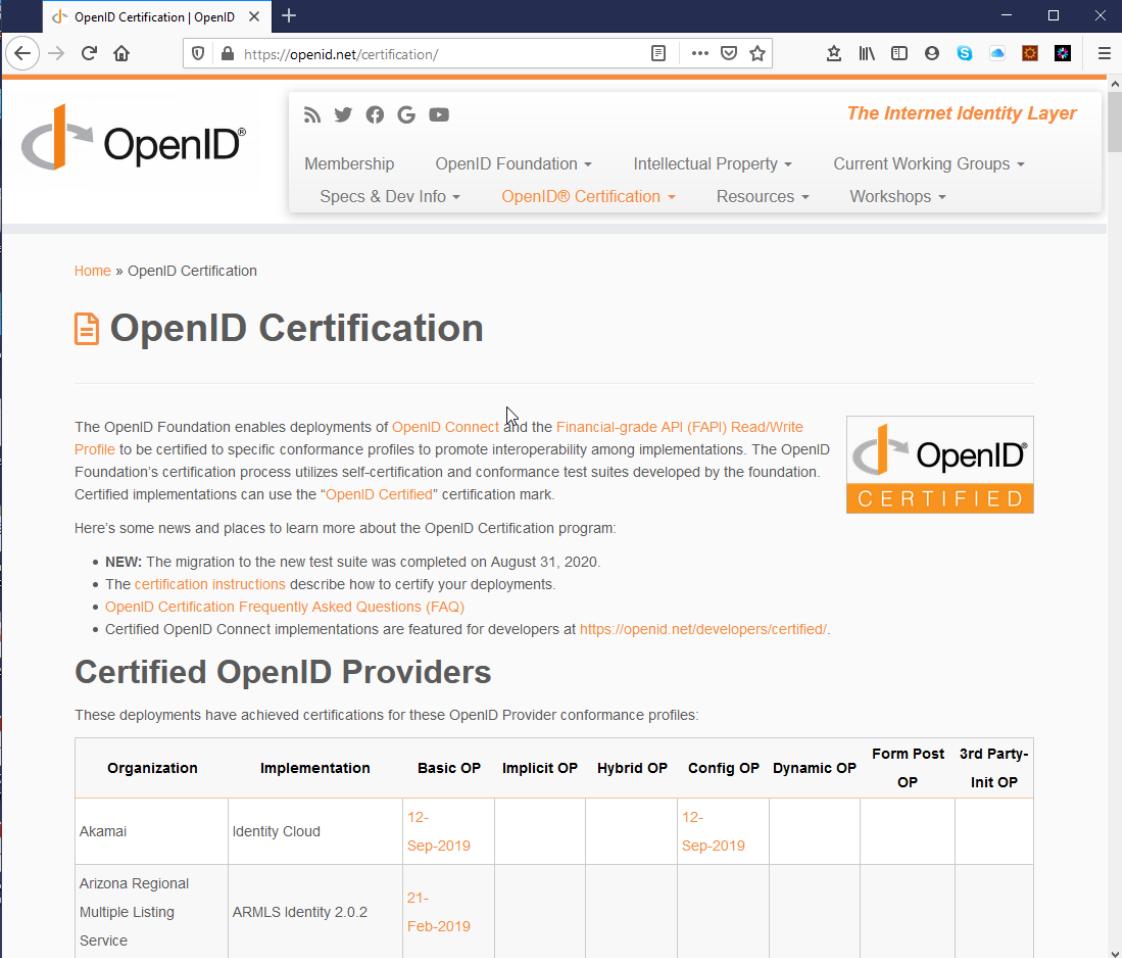
<https://www.youtube.com/watch?v=996OjexHze0>

*OpenID Connect on Liberty*

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



# There are a multitude of OpenID Certified Providers



The screenshot shows a web browser displaying the OpenID Certification page at <https://openid.net/certification/>. The page has a header with the OpenID logo and navigation links for Membership, OpenID Foundation, Intellectual Property, Current Working Groups, Specs & Dev Info, OpenID® Certification (which is highlighted), Resources, and Workshops. Below the header, it says "The Internet Identity Layer". The main content area is titled "OpenID Certification". It explains the certification process for OpenID Connect and the Financial-grade API (FAPI) Read/Write Profile. It features a "CERTIFIED" badge with the OpenID logo. A sidebar lists news items about the certification program. The "Certified OpenID Providers" section contains a table showing two entries: Akamai and Arizona Regional Multiple Listing Service, both using Identity Cloud.

Organization	Implementation	Basic OP	Implicit OP	Hybrid OP	Config OP	Dynamic OP	Form Post OP	3rd Party-Init OP
Akamai	Identity Cloud	12-Sep-2019			12-Sep-2019			
Arizona Regional Multiple Listing Service	ARMLS Identity 2.0.2	21-Feb-2019						

<https://openid.net/certification/>

# 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](https://openid.net/specs/openid-connect-core-1_0.html)
- **Again, for a very good explanation of this topic see YouTube video OAuth 2.0 and OpenID Connect (in plain English)** <https://www.youtube.com/watch?v=996OjexHze0>

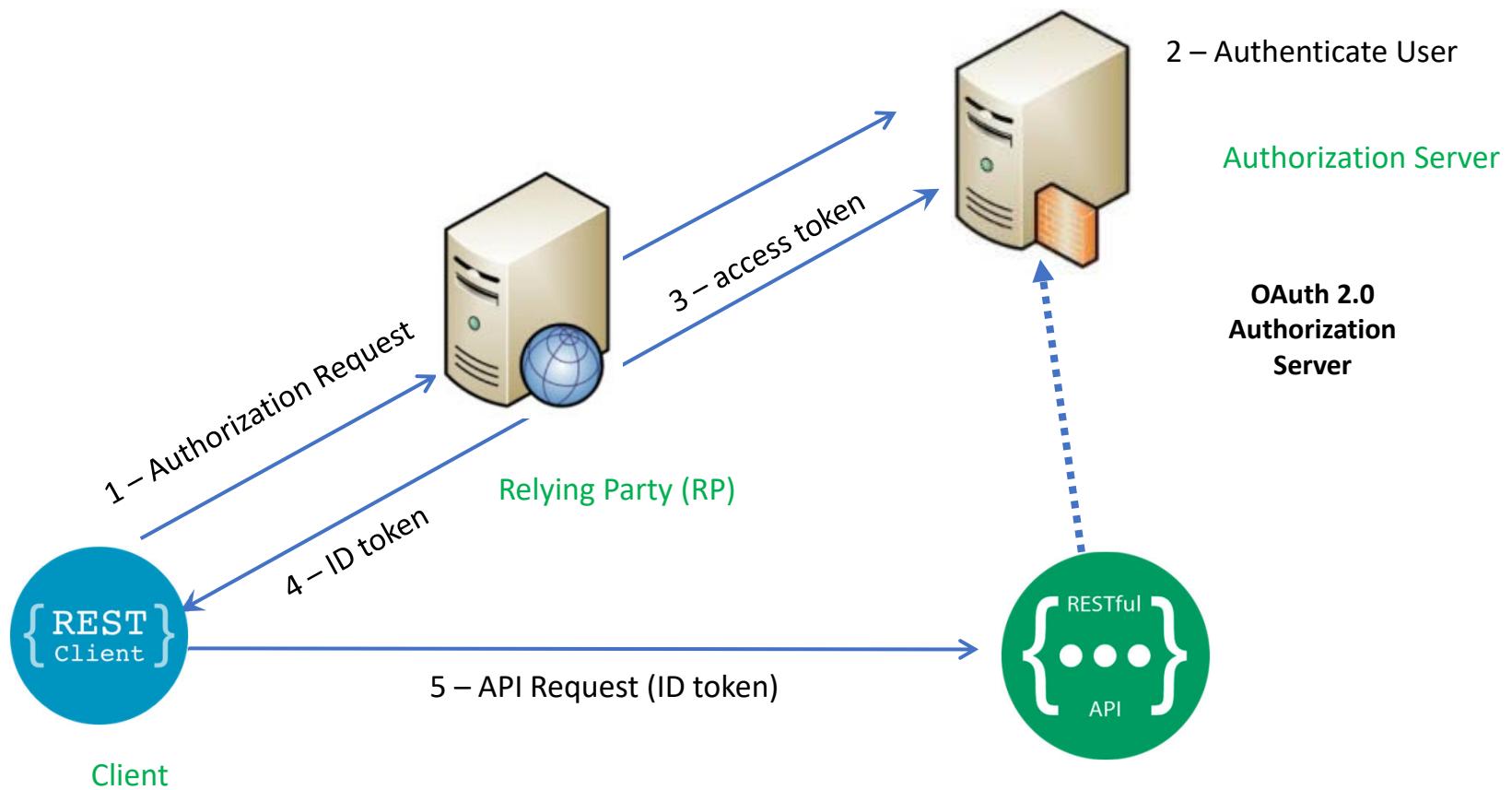
# Some basic OAuth/OpenID Connect terms

- **Authorization server** - The server that issues access tokens to the client after authenticating the resource owner and obtaining authorization. *In a z/OS Connect EE API requester scenario, the authorization server is called by the z/OS Connect EE server to retrieve an access token.*
- **Authorization Endpoint** - A service or endpoint on an OAuth authorization server that accepts an authorization request from a client to perform authentication and authorization of a user. The authorization endpoint returns an authorization grant, or code, to the client in the Authorization Code Flow. In the Implicit Flow, the authorization endpoint returns an access token to the client.
- **Token Endpoint** – A service or endpoint on an OP that accepts an authorization grant, or code, from a client in exchange for an access token, ID token, and refresh token
- **Access Token** – A credential that is used to access protected resources. An access token is a string that represents an authorization that is issued to the client. The access token is usually opaque to the client (it does not have to be opaque) and can be JSON Web Token (JWT). See URL <https://tools.ietf.org/html/rfc6749> Section 1.4 for more information.
- **OAuth token** - With OAuth 2.0, access tokens are used to access protected resources. An access token is normally a string that represents an authorization that is issued to the client. The string is usually opaque to the client. Opaque tokens may require that the token recipient call back to the server that issued the token. *However, an access token can also be in the form of a JSON Web Token (JWT) which does not require a call back (introspection).*
- **Scope** - Privilege or permission that allows access to a set of resources of a third party.

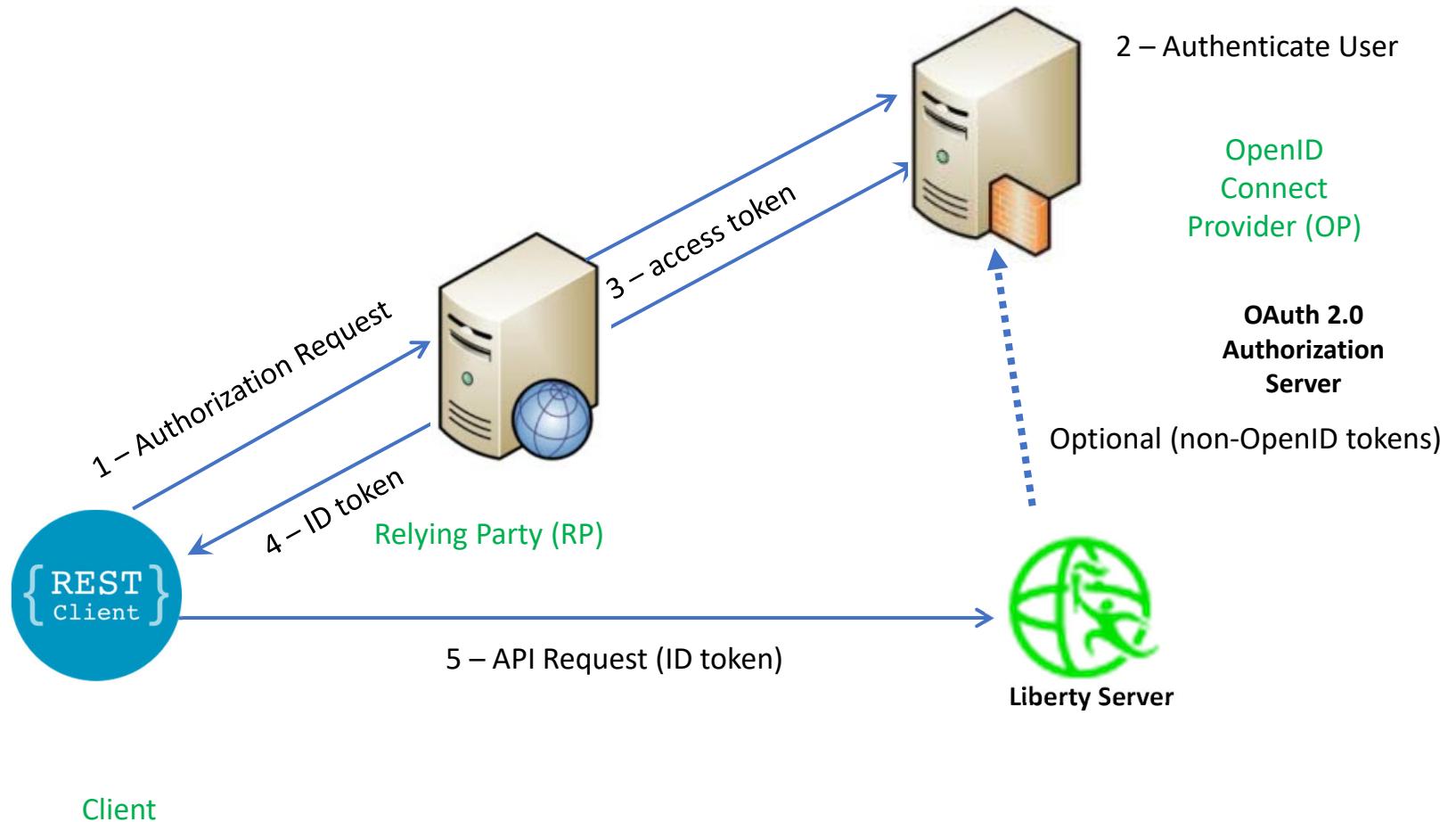
# Some basic OAuth/OpenID Connect terms

- **Relying Party (RP)** – An entity that relies on an OP to authenticate a user and obtain an authorization to access a user's resource. *For z/OS Connect API Requester, it is the Liberty server configured as an OpenID Connect Client, e.g., using <openidConnectClient/> XML configuration elements.*
- **OpenID Connect Provider (OP)** - An OAuth 2.0 authorization server that is capable of providing claims to a client or Relying Party (RP) , *an OpenID component.*
- **Resource owner** - An entity capable of granting access to a protected resource. When the resource owner is a person, it is referred to as an end user. *In a z/OS Connect EE API requester scenario, the resource owner might be the user of the CICS, IMS, or z/OS application.*
- **Resource server** - The server that hosts the protected resources and accepts and responds to protected resource requests by using access tokens. *In a z/OS Connect API provider, the resource server is the z/OS Connect server. In a z/OS Connect EE API requester scenario, the resource server is the request endpoint for the remote RESTful API*
- **ID Token** - is an OpenID Connect token that is an extension to OAuth 2.0 specification access tokens. This token is a JSON Web Token (JWT). See URL [https://openid.net/specs/openid-connect-core-1\\_0.html#IDToken](https://openid.net/specs/openid-connect-core-1_0.html#IDToken) for more information about the extensions.

# Typical OAuth Authorization Flow



# Typical Authorization Flow for an OpenID Connect token to a z/OS Connect API Provider



# Tech/Tip: Let's explore a Liberty OpenID Provider flow



```
<httpEndpoint host="*" httpPort="26212" httpsPort="26213" id="defaultHttpEndpoint"/>

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

<oauthProvider id="OIDCssl"
    httpsRequired="true"
    jwtAccessToken="true"
    autoAuthorize = "true"
    accessTokenLifetime="300">

<!-- Define OIDC Client for zCEE Authentication -->
<autoAuthorizeClient>zCEEClient</autoAuthorizeClient>
<localStore>
    <client name="zCEEClient"
        secret="secret"
        displayname="zCEEclient"
        scope="openid"
        enabled="true"
        resourceIds="myZcee"/>
</localStore>
```

## Key Points:

- **keyStoreRef** - A keystore containing the private key necessary for signing with an asymmetric algorithm.
- **jwtAccessToken** - generate a JSON Web Token, serialize it as a string and put in the place of the access token.

# Tech/Tip: Generating a JWT using Liberty as the OP



```
<autoAuthorizeClient>rpSsl</autoAuthorizeClient>
<localStore>
  <client name="rpSsl" 
    displayname="rpSsl"
    grantType="implicit"
    redirect="https://wg31.washington.ibm.com:26223/oidcclient/redirect/RPssl"
    scope="openid profile scope1 email phone address"
    enabled="true"
    resourceIds="myZcee"/>
</localStore>
</oauthProvider>

<!--Key store that contains certificate used to sign JWT-->
<keyStore fileBased="false" id="jwtStore"
  location="safkeyring:///JWT.KeyRing"
  password="password" readOnly="true" type="JCERACFKS"/>

<!-- Define a basic user registry -->
<basicRegistry id="basicRegistry"
  realm="zCEERealm">
  <user name="auser" password="pwd"/>
  <user name="Fred" password="fredpwd"/>
  <user name="distuser1" password="pwd"/>
  <user name="distuser2" password="pwd"/>
</basicRegistry>
```

```
RACMAP ID(FRED) MAP USERDIDFILTER(NAME('Fred'))
  REGISTRY(NAME('*')) WITHLABEL('zCEE JWT FRED')
RACMAP ID(USER1) MAP USERDIDFILTER(NAME('distuser1'))
  REGISTRY(NAME('*')) WITHLABEL('zCEE JWT distuser1')
RACMAP ID(USER2) MAP USERDIDFILTER(NAME('distuser2'))
  REGISTRY(NAME('*')) WITHLABEL('zCEE JWT distuser2')
```

# Tech/Tip: RACMAP Commands

```
RACMAP ID(USER1) MAP USERDIDFILTER(NAME('distuser1'))
    REGISTRY(NAME('*')) WITHLABEL('zCEE token user1')
RACMAP ID(USER1) MAP USERDIDFILTER(NAME('distributeUser1'))
    REGISTRY(NAME('zCEERealm')) WITHLABEL('zCEE user1')
RACMAP ID(USER1) MAP USERDIDFILTER(NAME('UID=user1,CN=User Name,OU=IBM ATG,O=IBM,C=US'))
    registry(name('*')) withlabel('USER X500 DN')
RACMAP ID(ATSUSER) MAP USERDIDFILTER(NAME('OU=IBM ATS,O=IBM,C=US'))
    registry(name('*')) withlabel('ATS USER')
RACMAP ID(IBMUSER) MAP USERDIDFILTER(NAME('O=IBM,C=US'))
    registry(name('*')) withlabel('IBM USER')
```

```
RACMAP ID(USER1) LISTMAP
Label: zCEE token user1
Distributed Identity User Name Filter:
>distuser1<
Registry Name:
>*<

Label: zCEE user1
Distributed Identity User Name Filter:
>distributeUser1<
Registry Name:
>zCEERealm<

Label: USER X500 DN
Distributed Identity User Name Filter:
>UID=user1,CN=User Name,OU=IBM ATG,O=IBM,C=US<
Registry Name:
>*<
```

```
RACMAP ID(USER1) LISTMAP(LABEL('USER X500 DN'))

RACMAP ID(USER1) DELMAP (LABEL('zCEE distuser1'))

RACMAP QUERY USERDIDFILTER(NAME('USER1')) REGISTRY(NAME('*'))
```

# Liberty/zCEE OpenID Client identity mapping configuration attributes



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

Use distributed identity filters to map the distributed identities to SAF user IDs, e.g., IDIDMAP, e.g., RACMAP.

```
<authFilter id="ATSAuthFilter">
    <requestUrl id="ATSDemoUrl"
        name="ATSRefererUri"
        matchType="contains"
        urlPattern="/cscvinc/employee|/db2/employee|/mqapi/loan"/>
</authFilter>
<openidConnectClient id="ATS"
    httpsRequired="true"
    authFilterRef="ATSAuthFilter"
    inboundPropagation="required"
    scope="openid profile email"
    audiences="myZcee"
    issuerIdentifier="https://wq31.washington.ibm.com:26213/oidc/endpoint/OP"
    mapIdentityToRegistryUser="false"
    signatureAlgorithm="RS256"
    userIdentityToCreateSubject="sub"
    trustAliasName="JWT-Signer-Certificate"
    trustStoreRef="jwtTrustStore"
    authnSessionDisabled="true"
    disableLtpaCookie="true">
</openidConnectClient>
<keyStore fileBased="false" id="jwtTrustStore"
    location="safkeyring:///JWT.KeyRing"
    password="password" readOnly="true" type="JCERACFKS"/>
```

Specifies whether to map the identity to a registry user. If this is set to false, then the user registry is not used to create the user subject.

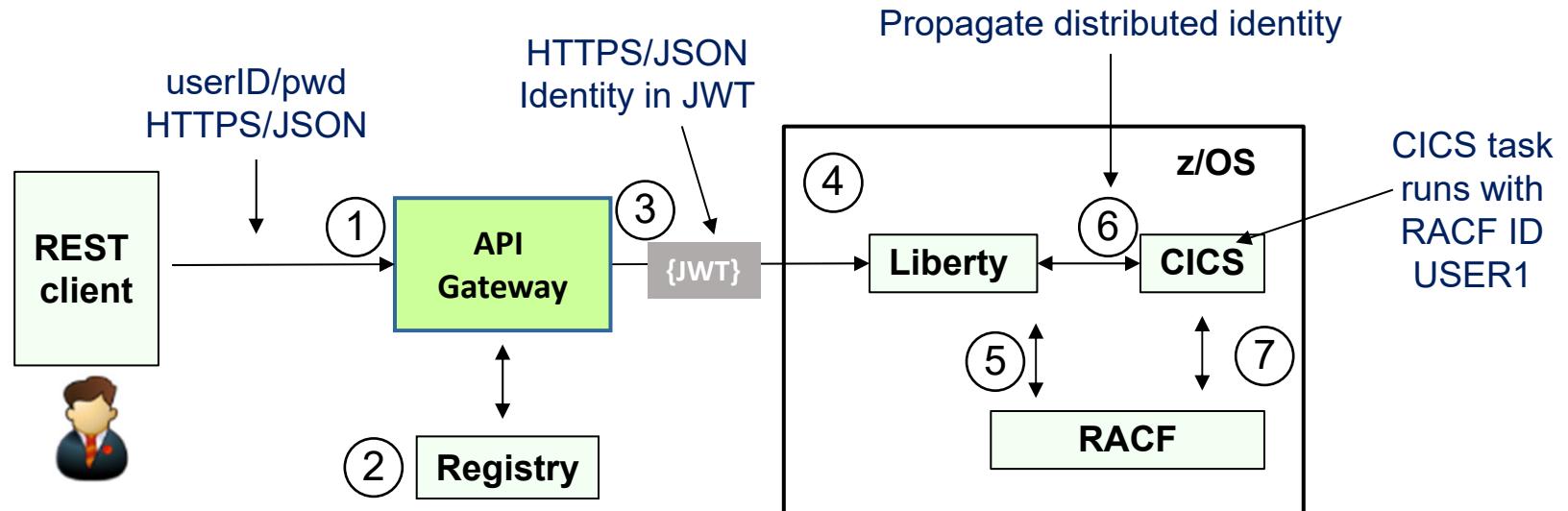
# Liberty/zCEE OpenID Client identity mapping configuration attributes (JWK)



```
{  
  "kid": "574eafad-fcb5-412e-97a3-8100a1c1fa5b",  
  "alg": "RS256"  
}  
{  
  "sub": "mitchj",  
  "aud": "myZCEE",  
  "iss": "https://wg31.washington.ibm.com:26213/oidc/endpoint/OP",  
  "exp": 1610451176,  
  "iat": 1610451876  
}
```

```
<openidConnectClient  
  id="ATSJWK"  
  clientId="RS-JWT-ZCEE"  
  authFilterRef="jwkAuthFilter"  
  inboundPropagation="required"  
  signatureAlgorithm="RS256"  
  userIdentifier="sub"  
  mapIdentityToRegistryUser="true"  
  issuerIdentifier="https://wg31.washington.ibm.com:26213/oidc/endpoint/OP"  
  disableLtpaCookie="true"  
  audiences="myZcee"  
  jwkEndpointUrl="https://wg31.washington.ibm.com:26213/oidc/endpoint/OP/jwk"  
  jwkClientId="jwtClient"  
  jwkSecret="jwtSecret"/>  
</openidConnectClient>
```

# Example scenario – security flow



Edward Johnson

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

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

## JWT used in scenario

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

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

# Configuring authentication with JWT



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

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

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

# Using authorization filters with z/OS Connect EE



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

```
<openidConnectClient id="RPssl" inboundPropagation="required"
    signatureAlgorithm="RS256" trustAliasName="JWT-Signer"
    trustStoreRef="jwtTrustStore"
    userIdentityToCreateSubject="sub" mapIdentityToRegistryUser="true"
    issuerIdentifier="https://wg31.washington.ibm.com:26213/oidc/endpoint/OPssl"
    authnSessionDisabled="true" audiences="myZcee"
    authFilterRef="JwtAuthFilter"/>
<authFilter id="API Gateway">
    <remoteAddress id="ApiAddress" ip="10.7.1.*" matchType="equals"/>
</authFilter>
<authFilter id="URLFilter">
    <requestUrl id="URL" urlPattern="/cscvinc/employee|/db2/employee|/mqapi/loan"/>
        matchType="equals"/> </authFilter>
<authFilter id="JwtAuthFilter" >
    <requestHeader id="authHeader" name="Authorization" value="Bearer" matchType="contains"/>
</authFilter>
```

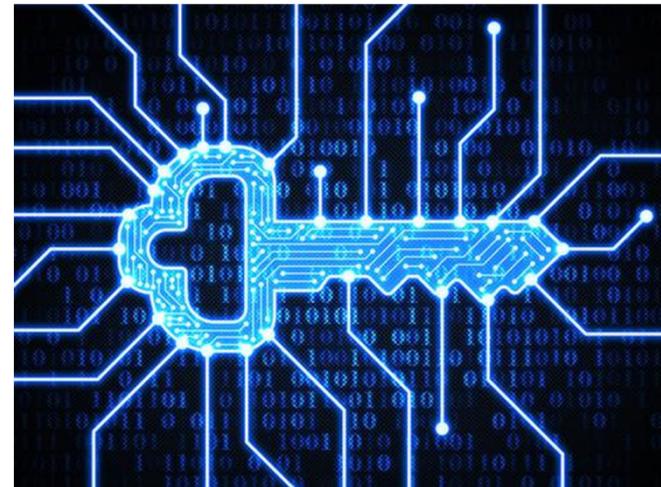
## Some alternative filter types

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

# General security terms or considerations

- Identifying who or what is requesting access (**Authentication**)
  - Basic Authentication
  - Mutual Authentication using TLS
  - 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 generating/sending 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.



# z/OS Connect Security server XML Configuration



```
<zosconnect_zosConnectManager  
    requireAuth="true"  
    requireSecure="true|false"/>  
  
<zosconnect_zosConnectAPIs>  
    <zosConnectAPI name="catalog"  
        requireAuth="true"  
        requireSecure="true|false"/>  
</zosconnect_zosConnectAPIs>  
  
<zosconnect_services>  
    <service id="selectByEmployee"  
        name="selectEmployee"  
        requireAuth="true"  
        requireSecure="true|false"/>  
</zosconnect_services>  
  
<zosconnect_apiRequesters>  
    requireAuth="true"  
    <apiRequester name="cscvincapi_1.0.0"  
        requireAuth="true"  
        requireSecure="true|false"/>  
</zosconnect_apiRequesters>  
  
<zosconnect_zosconnect_service  
    id="selectByEmployee"  
    name="selectEmployee"  
    requireAuth="true"  
    requireSecure="true|false"/>
```

Globally, requires that inbound request using HTTPS in order to access APIs, services and API requesters, unless overridden on the specific resource definitions.

Requires that inbound request use HTTPS in order to access the API.

Requires that inbound request use HTTPS when directly accessing this service.

Requires that all inbound request for this API requester use HTTPS..

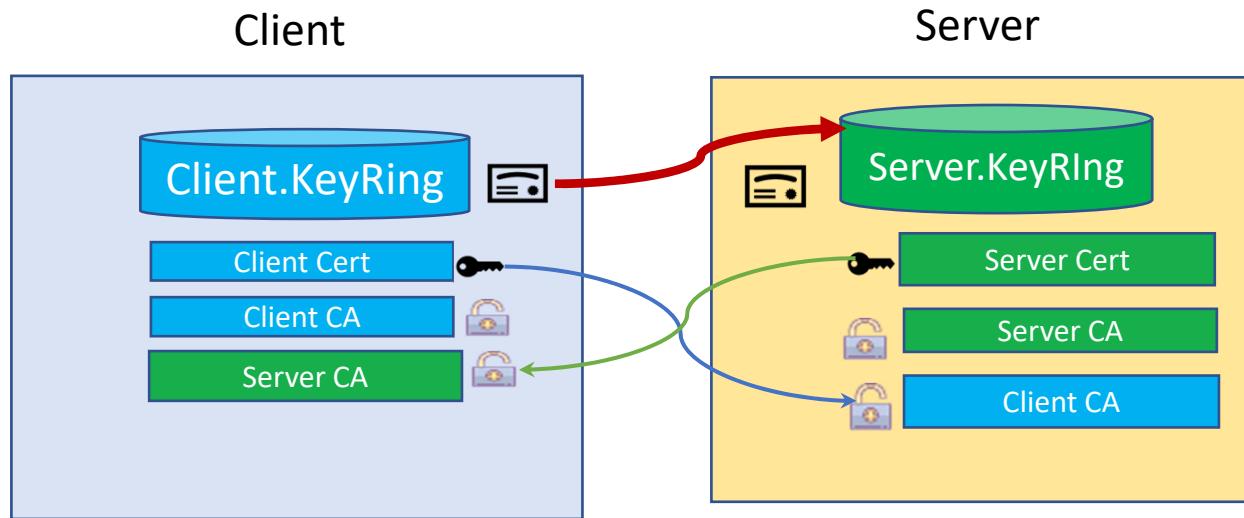
Requires that inbound request for this service use HTTPS when directly accessing this service.

## requireSecure controls inbound TLS connections

# Basic TLS Handshake Flow

TLS handshake –

Encryption/Message Integrity



safkeyring://KeyRing v safkeyring://owner/KeyRing

RACF FACILITY resources

- IRR.DIGTCERT.LISTRING
  - READ to list your own key ring
  - UPDATE to list another user's key ring
- IRR.DIGTCERT.LIST
  - READ to list your own certificate
  - UPDATE to list another user's certificate
  - CONTROL to list SITE of CERTAUTH certificates

Key icon: Certificate with a private key\*

Padlock icon: Certificate Authority (CA) certificate chain#

\*For server and/or mutual authentication to work, the endpoint sending its server or client certificate must use a personal certificate with a private key. The private key is required to decrypt (or encrypt) a message digest that is sent from the other endpoint during the handshake flow. Generation of a message digest also requires access to the CA certificate used to sign the certificate.

#Refers to the set or of certificates used to issue the server or client personal certificate including any intermediate certificates all the way to the root CA.

# Tech/Tip: cURL trace of a TLS Handshake

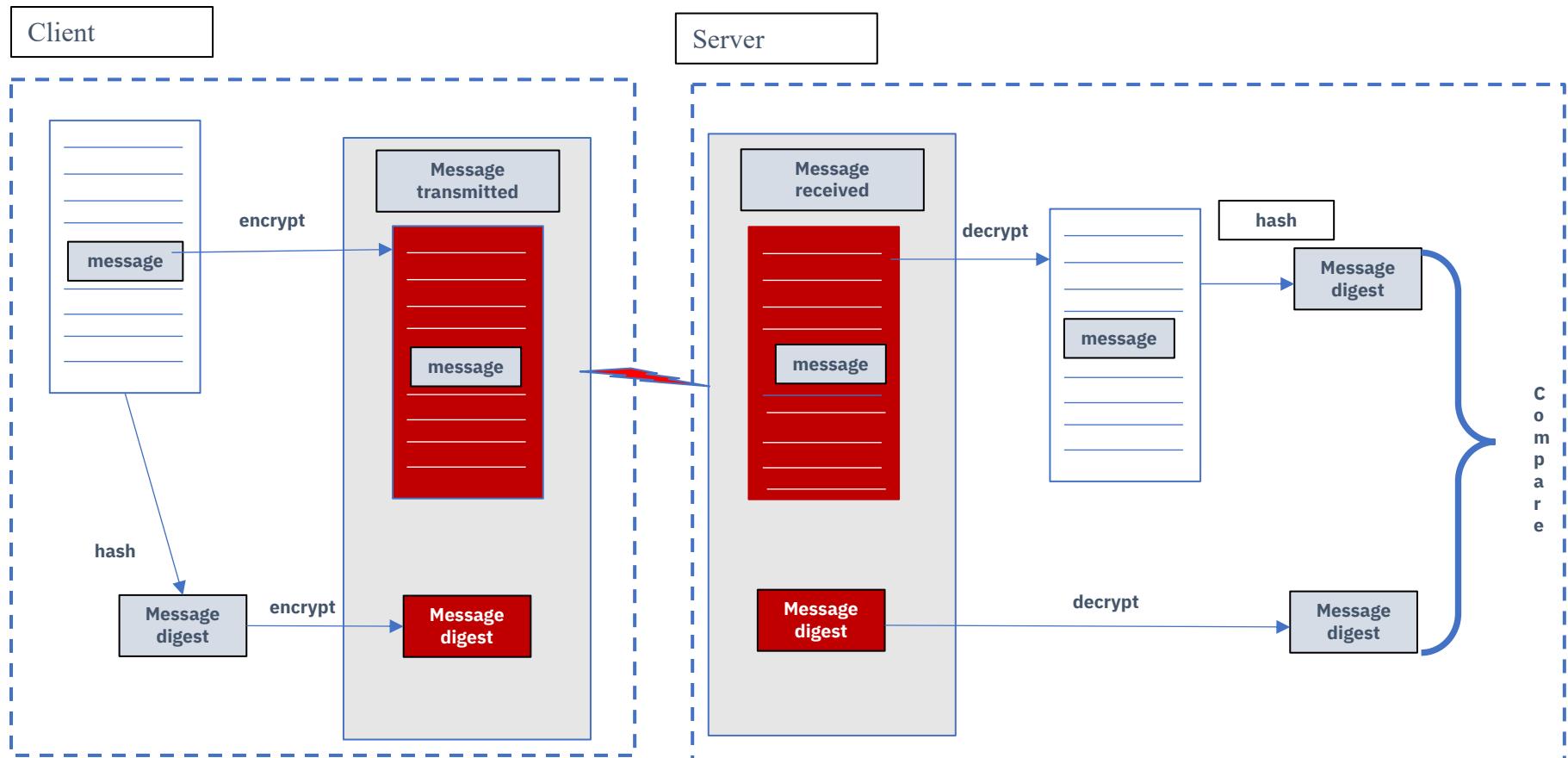
- \* successfully set certificate verify locations:
  - \* CAfile: certauth.pem
  - CApath: none
- \* TLSv1.3 (OUT), TLS handshake, Client hello (1):
- \* TLSv1.3 (IN), TLS handshake, Server hello (2):
- \* TLSv1.2 (IN), TLS handshake, Certificate (11):
- \* TLSv1.2 (IN), TLS handshake, Server key exchange (12):
- \* TLSv1.2 (IN), TLS handshake, Server finished (14):
- \* TLSv1.2 (OUT), TLS handshake, Client key exchange (16):
- \* TLSv1.2 (OUT), TLS change cipher, Change cipher spec (1):
- \* TLSv1.2 (OUT), TLS handshake, Finished (20):
- \* TLSv1.2 (IN), TLS handshake, Finished (20):
- \* SSL connection using TLSv1.2 / ECDHE-RSA-AES256-GCM-SHA384
- \* Server certificate:
  - \* subject: O=IBM; OU=LIBERTY; CN=wg31.washington.ibm.com
  - \* start date: Dec 23 04:00:00 2020 GMT
  - \* expire date: Jan 1 03:59:59 2023 GMT
  - \* common name: wg31.washington.ibm.com (matched)
  - \* issuer: OU=LIBERTY; CN=CA for Liberty
  - \* SSL certificate verify ok.
- \* TLSv1.2 (IN), TLS handshake, Server key exchange (12):  
~~\* TLSv1.2 (IN), TLS handshake, Request CERT (13):~~
- \* TLSv1.2 (IN), TLS handshake, Server finished (14):  
~~\* TLSv1.2 (OUT), TLS handshake, Certificate (11):~~
- \* TLSv1.2 (OUT), TLS handshake, Client key exchange (16):  
~~\* TLSv1.2 (OUT), TLS handshake, CERT verify (15):~~
- \* TLSv1.2 (OUT), TLS change cipher, Change cipher spec (01):

TLS 1.2 <https://tools.ietf.org/html/rfc5246>

TLS 1.3 <https://tools.ietf.org/html/rfc8446>

© 2017, 2021 IBM Corporation

# Message Integrity and Encryption



# Tech/Tip: A note on cipher suite names

A CipherSuite is a suite of cryptographic algorithms used by a TLS connection. A suite comprises three distinct algorithms:

- The key exchange and authentication algorithm, used during the handshake
- The encryption algorithm, used to encipher the data
- The MAC (Message Authentication Code) algorithm, used to generate the message digest

There are several options for each component of the suite, but only certain combinations are valid when specified for a TLS connection. The name of a valid CipherSuite defines the combination of algorithms used. For example, the CipherSuite ***TLS\_RSA\_WITH\_AES\_128\_CBC\_SHA*** specifies:

- The RSA key exchange and authentication algorithm
- The AES encryption algorithm, using a 128-bit key and cipher block chaining (CBC) mode
- The SHA-1 Message Authentication Code (MAC)

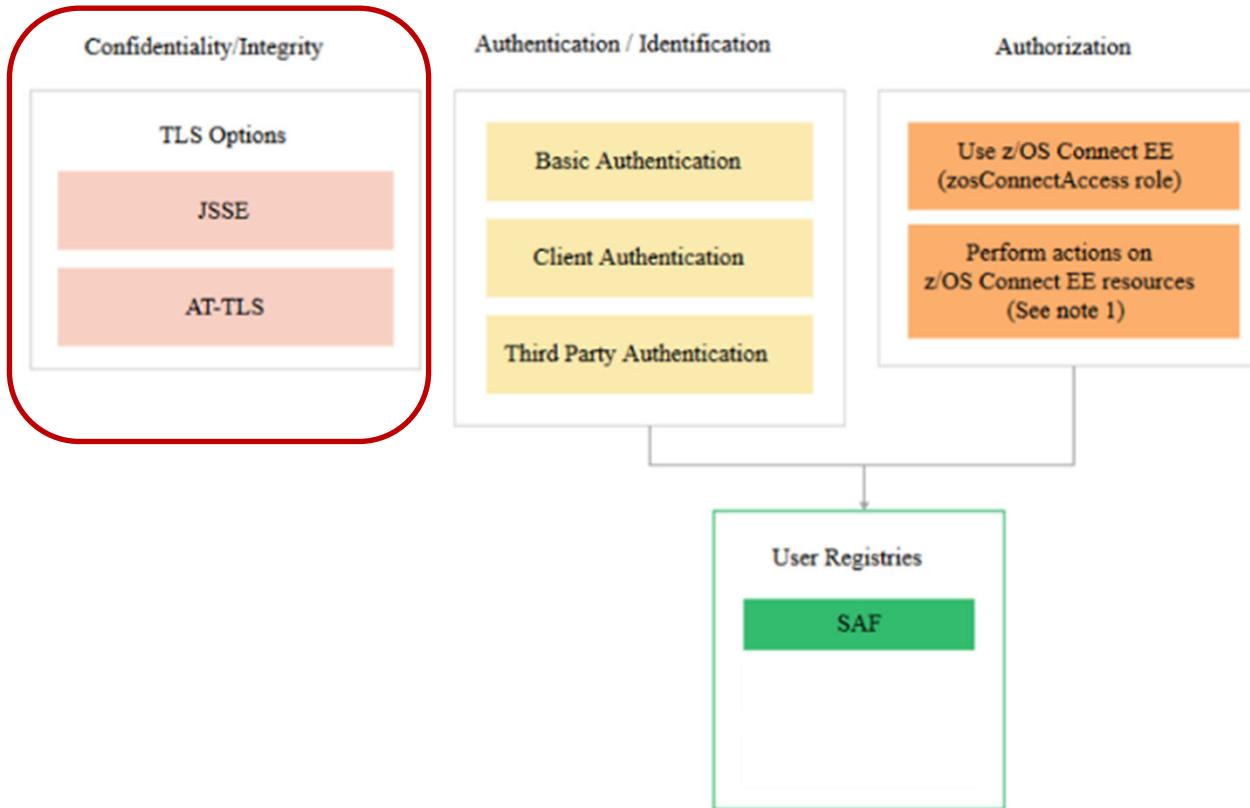
Note				
To use some CipherSuites, the 'unrestricted' policy files need to be configured in the JRE. For more details of how policy files are set up in an SDK or JRE, see the <i>IBM SDK Policy files</i> topic in the <i>Security Reference for IBM SDK, Java Technology Edition</i> for the version you are using.				
Table 1. CipherSpecs supported by IBM MQ and their equivalent CipherSuites				
CipherSpec	Equivalent CipherSuite (IBM JRE)	Equivalent CipherSuite (Oracle JRE)	Protocol	FIPS 140-2 compatible
ECDHE_ECDSA_3DES_EDE_CBC_SHA256	SSL_ECDHE_ECDSA_WITH_3DES_EDE_CBC_SHA	TLS_ECDHE_ECDSA_WITH_3DES_EDE_CBC_SHA	TLS 1.2	yes
ECDHE_ECDSA_AES_128_CBC_SHA256	SSL_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256	TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256	TLS 1.2	yes
ECDHE_ECDSA_AES_128_GCM_SHA256	SSL_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256	TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256	TLS 1.2	yes
ECDHE_ECDSA_AES_256_CBC_SHA384	SSL_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384	TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384	TLS 1.2	yes
ECDHE_ECDSA_AES_256_GCM_SHA384	SSL_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384	TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384	TLS 1.2	yes
ECDHE_ECDSA_NULL_SHA256	SSL_ECDHE_ECDSA_WITH_NULL_SHA	TLS_ECDHE_ECDSA_WITH_NULL_SHA	TLS 1.2	no
ECDHE_ECDSA_RC4_128_SHA256	SSL_ECDHE_ECDSA_WITH_RC4_128_SHA	TLS_ECDHE_ECDSA_WITH_RC4_128_SHA	TLS 1.2	no
ECDHE_RSA_3DES_EDE_CBC_SHA256	SSL_ECDHE_RSA_WITH_3DES_EDE_CBC_SHA	TLS_ECDHE_RSA_WITH_3DES_EDE_CBC_SHA	TLS 1.2	yes
ECDHE_RSA_AES_128_CBC_SHA256	SSL_ECDHE_RSA_WITH_AES_128_CBC_SHA256	TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256	TLS 1.2	yes
ECDHE_RSA_AES_128_GCM_SHA256	SSL_ECDHE_RSA_WITH_AES_128_GCM_SHA256	TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256	TLS 1.2	yes
ECDHE_RSA_AES_256_CBC_SHA384	SSL_ECDHE_RSA_WITH_AES_256_CBC_SHA384	TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384	TLS 1.2	yes
ECDHE_RSA_AES_256_GCM_SHA384	SSL_ECDHE_RSA_WITH_AES_256_GCM_SHA384	TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384	TLS 1.2	yes
ECDHE_RSA_NULL_SHA256	SSL_ECDHE_RSA_WITH_NULL_SHA	TLS_ECDHE_RSA_WITH_NULL_SHA	TLS 1.2	no
ECDHE_RSA_RC4_128_SHA256	SSL_ECDHE_RSA_WITH_RC4_128_SHA	TLS_ECDHE_RSA_WITH_RC4_128_SHA	TLS 1.2	no

[https://www.ibm.com/support/knowledgecenter/SSFKSJ\\_9.1.0/com.ibm.mq.dev.doc/q113210.htm](https://www.ibm.com/support/knowledgecenter/SSFKSJ_9.1.0/com.ibm.mq.dev.doc/q113210.htm)

© 2017, 2021 IBM Corporation

mitchj@us.ibm.com

# Liberty and z/OS Connect EE security options

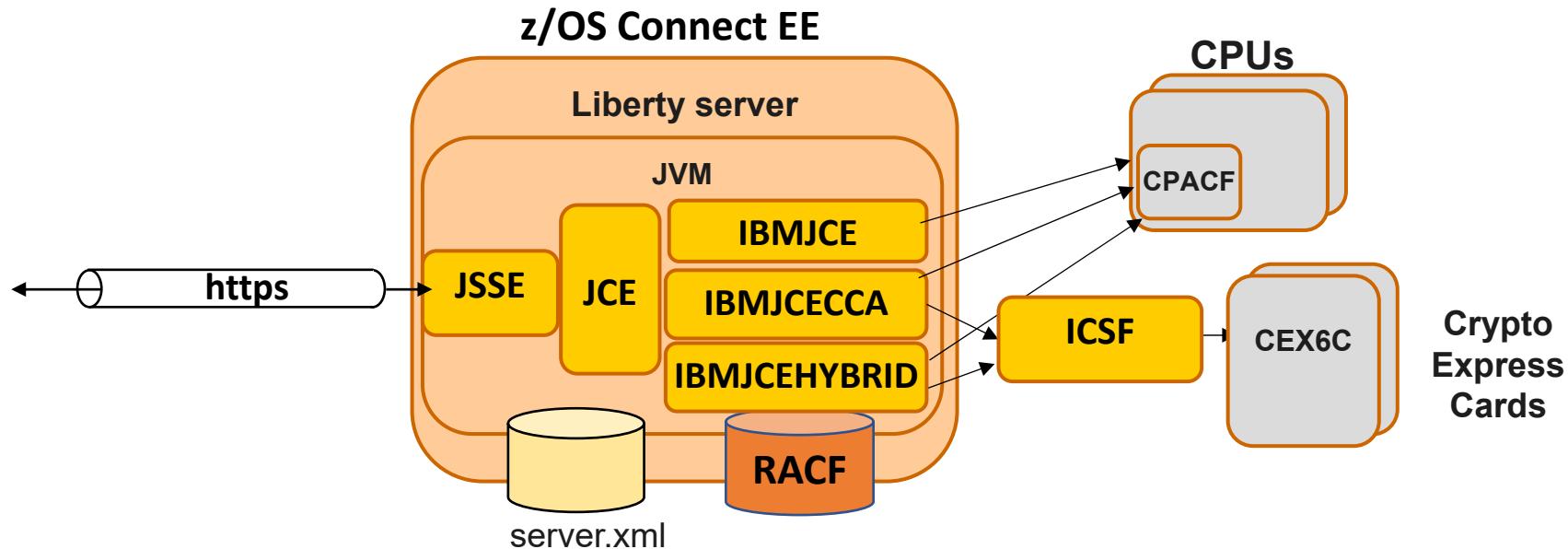


The actions which can be controlled by authorization are deploying, querying, updating, starting, stopping and deleting of APIs, services and API requesters.

# Using JSSE with Liberty



The server XML configuration defines the HTTPS ports, key rings, and other JSSE attributes

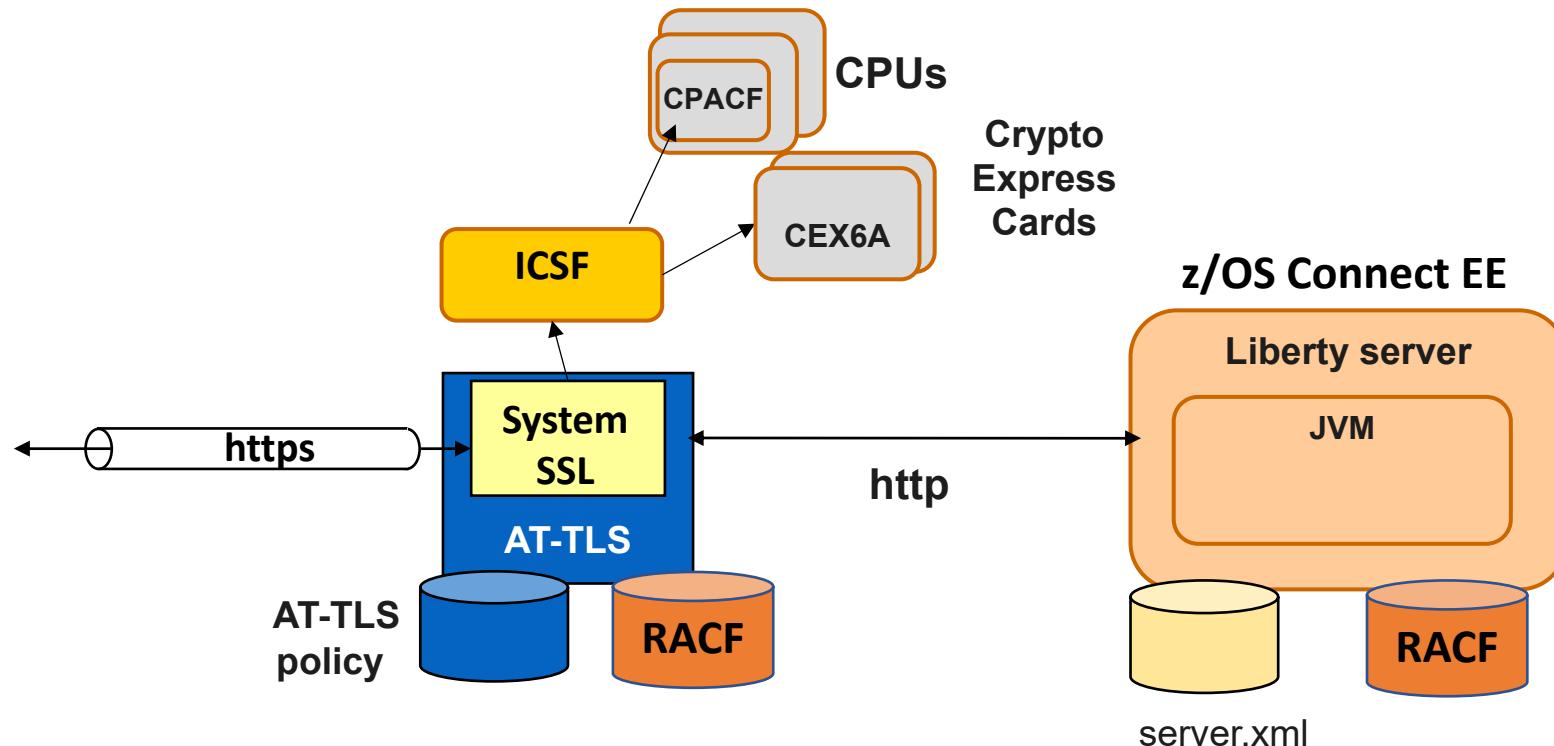


- z/OS Connect EE support for TLS is based on **Liberty** server support
- **Java Secure Socket Extension (JSSE)** API provides framework and Java implementation of TLS protocols used by Liberty HTTPS support
- **Java Cryptography Extension (JCE)** is standard extension to the Java Platform that provides implementation for cryptographic services
- **IBM Java SDK for z/OS** provides three different JCE providers, **IBMJCE**, **IBMJCECCA** and **IBMJCEHYBRID**.
- The JCE providers access **CPACF (CP Assist for Cryptographic Functions)** directly, therefore keep your Java service levels current.

# Using AT-TLS with Liberty



The server XML configuration defines no HTTPS ports, key rings or other JSSE attributes

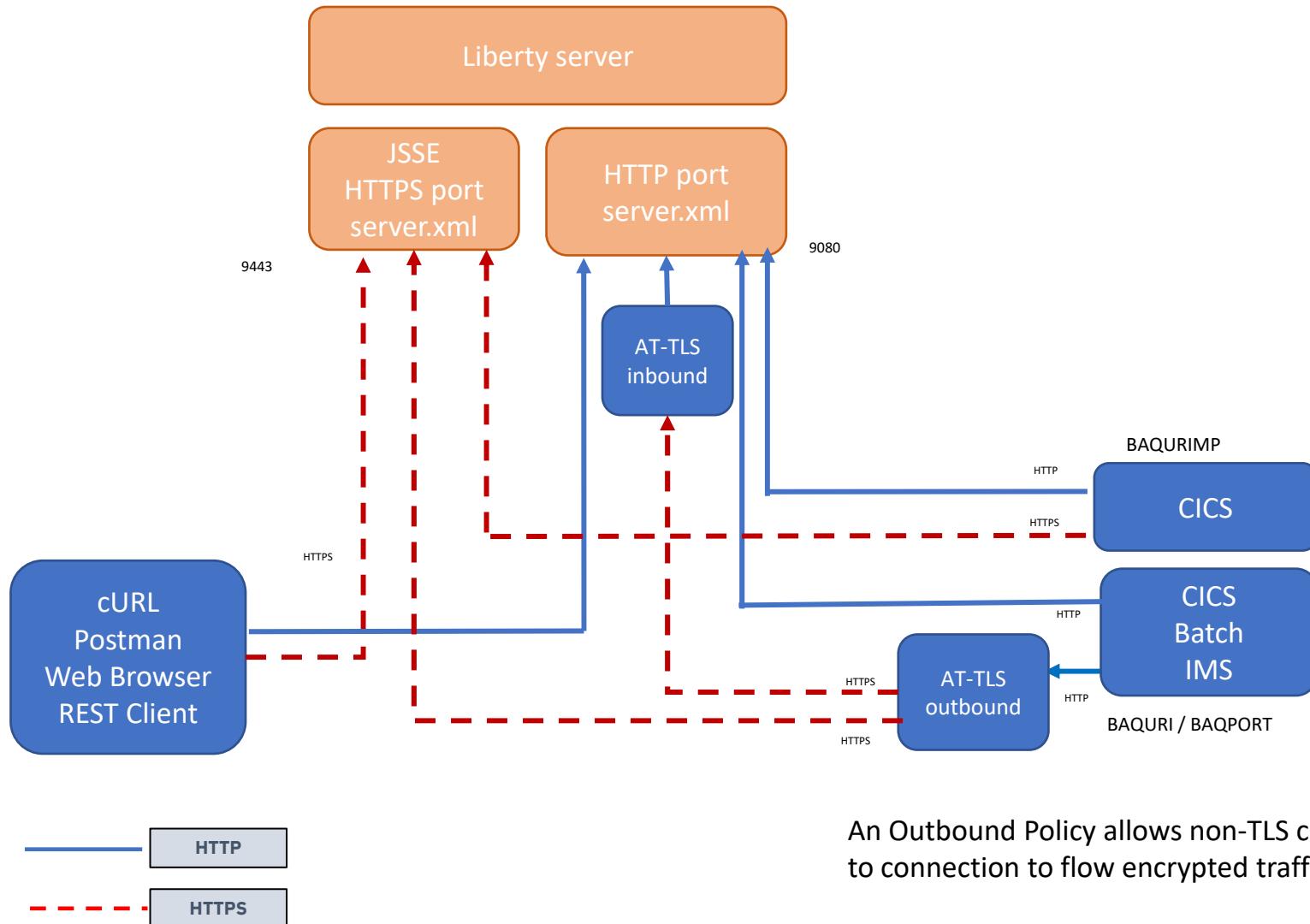


- Application Transparent TLS (AT-TLS) creates a secure session on behalf of z/OS Connect
- Only define http ports in server.xml (z/OS Connect does not know that TLS session exists)
- Define TLS protection for all applications (including z/OS Connect) in **AT-TLS policy**
- AT-TLS uses **System SSL**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

# JSSE and AT-TLS comparison

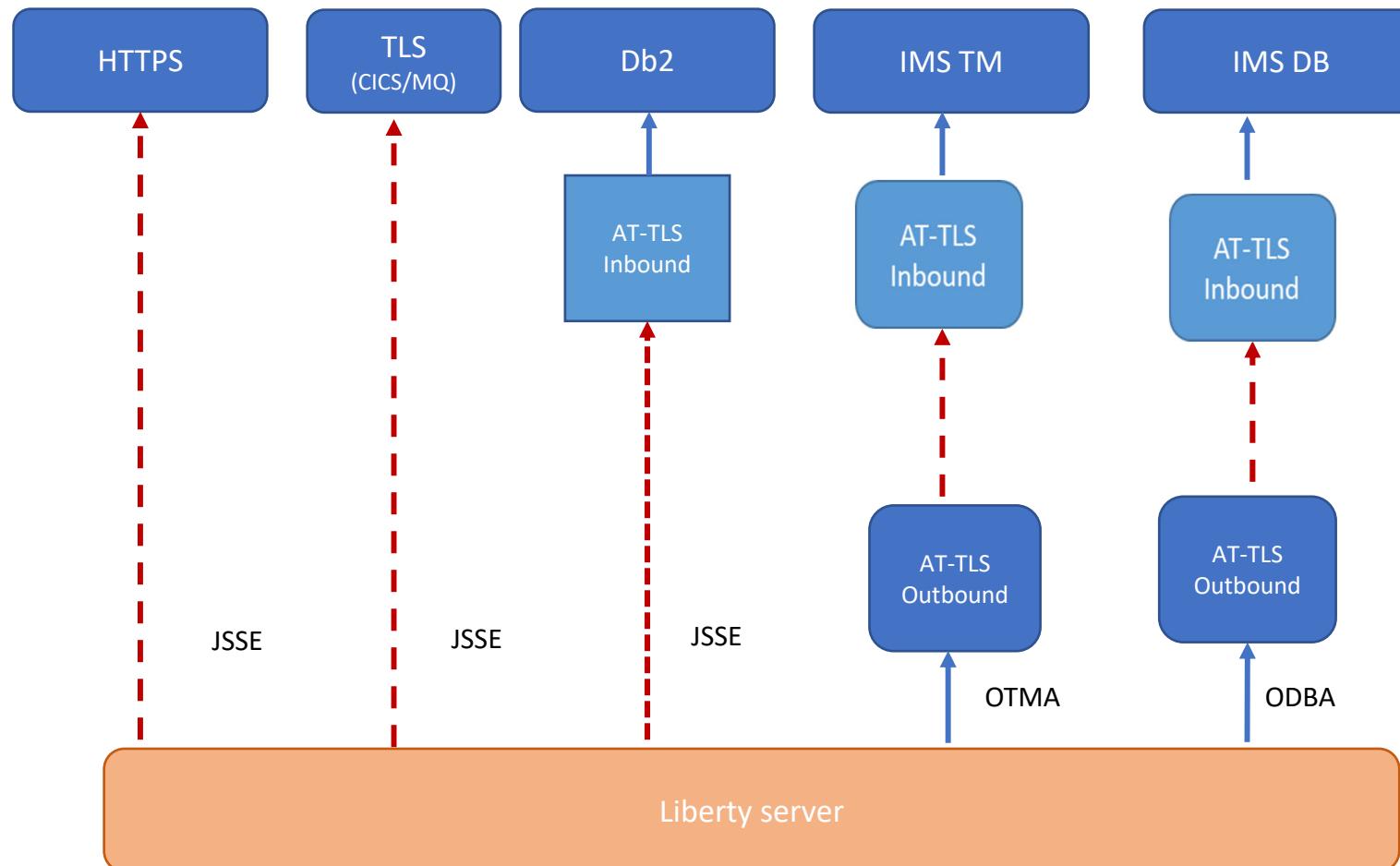
Capability	Description	JSSE	AT-TLS
Server authentication	Verification of z/OS Connect server certificate by client	Yes	Yes
Mutual authentication	Verification of client certificate by z/OS Connect	Yes	Yes
TLS client authentication	Use of client certificate for authentication	Yes	No
Support for requireSecure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

# TLS client encryption to a Liberty server scenarios





# TLS encryptions from a Liberty server (HTTPS/native TLS/OTMA/ODBA) scenarios



Inbound Policies Provide  
TLS support for incoming  
requests to Db2 and IMS  
Connect

Outbound Policies  
provide TLS support for  
request outbound from  
Liberty

**Let's explore using TLS for  
encryption and data integrity  
using samples in various scenarios**

# Using the contents of these key rings

Digital ring information for user LIBSERV:				
Ring: >zCEE.KeyRing<				
Certificate Label Name	Cert Owner	USAGE	DEFAULT	
<b>zCEE CA</b>	CERTAUTH	CERTAUTH	NO	
<b>Liberty CA</b>	CERTAUTH	CERTAUTH	NO	
<b>zCEE Client Cert</b>	<b>ID (LIBSERV)</b>	<b>PERSONAL</b>	<b>YES</b>	
<b>DB2 CA</b>	CERTAUTH	CERTAUTH	NO	
<b>MQ CA</b>	CERTAUTH	CERTAUTH	NO	
<b>CICS CA</b>	CERTAUTH	CERTAUTH	NO	
Ring: >Liberty.KeyRing<				
Certificate Label Name	Cert Owner	USAGE	DEFAULT	
<b>Liberty Server Cert</b>	<b>ID (LIBSERV)</b>	<b>PERSONAL</b>	<b>YES</b>	
<b>Liberty CA</b>	CERTAUTH	CERTAUTH	NO	
<b>zCEE CA</b>	CERTAUTH	CERTAUTH	NO	
<b>CICS CA</b>	CERTAUTH	CERTAUTH	NO	
Digital ring information for user CICSSTC:				
Ring: >CICS.KeyRing<				
Certificate Label Name	Cert Owner	USAGE	DEFAULT	
<b>CICS CA</b>	CERTAUTH	CERTAUTH	NO	
<b>CICS Client Cert</b>	<b>ID (CICSSTC)</b>	<b>PERSONAL</b>	<b>YES</b>	
<b>Liberty CA</b>	CERTAUTH	CERTAUTH	NO	
<b>zCEE CA</b>	CERTAUTH	CERTAUTH	NO	
Digital ring information for user DB2USER:				
Ring: >Db2.KeyRing<				
Certificate Label Name	Cert Owner	USAGE	DEFAULT	
<b>DB2 CA</b>	CERTAUTH	CERTAUTH	NO	
<b>zCEE CA</b>	CERTAUTH	CERTAUTH	NO	
<b>DB2USER</b>	<b>ID (DB2USER)</b>	<b>PERSONAL</b>	<b>YES</b>	

Tech-Tip: when more than one personal certificate is connected to a key ring. Use the SSL repertoire *serverKeyAlias* or *clientKeyAlias* attributes to select the personal certificate to be used in a handshake.



# Using this Liberty JSSE server XML configuration

```
<!-- Enable features -->
<featureManager>
    <feature>transportSecurity-1.0</feature>
</featureManager>

<sslDefault sslRef="DefaultSSLSettings"
    outboundSSLRef="OutboundSSLSettings" />

<ssl id="DefaultSSLSettings"
    keyStoreRef="CellDefaultKeyStore"
    trustStoreRef="CellDefaultKeyStore"
    clientAuthenticationSupported="true"
    clientAuthentication="true"
    serverKeyAlias="Liberty Server Cert"/>

<keyStore id="CellDefaultKeyStore"
    location="safkeyring:///Liberty.KeyRing"
    password="password" type="JCERACFKS"
    fileBased="false" readOnly="true" />

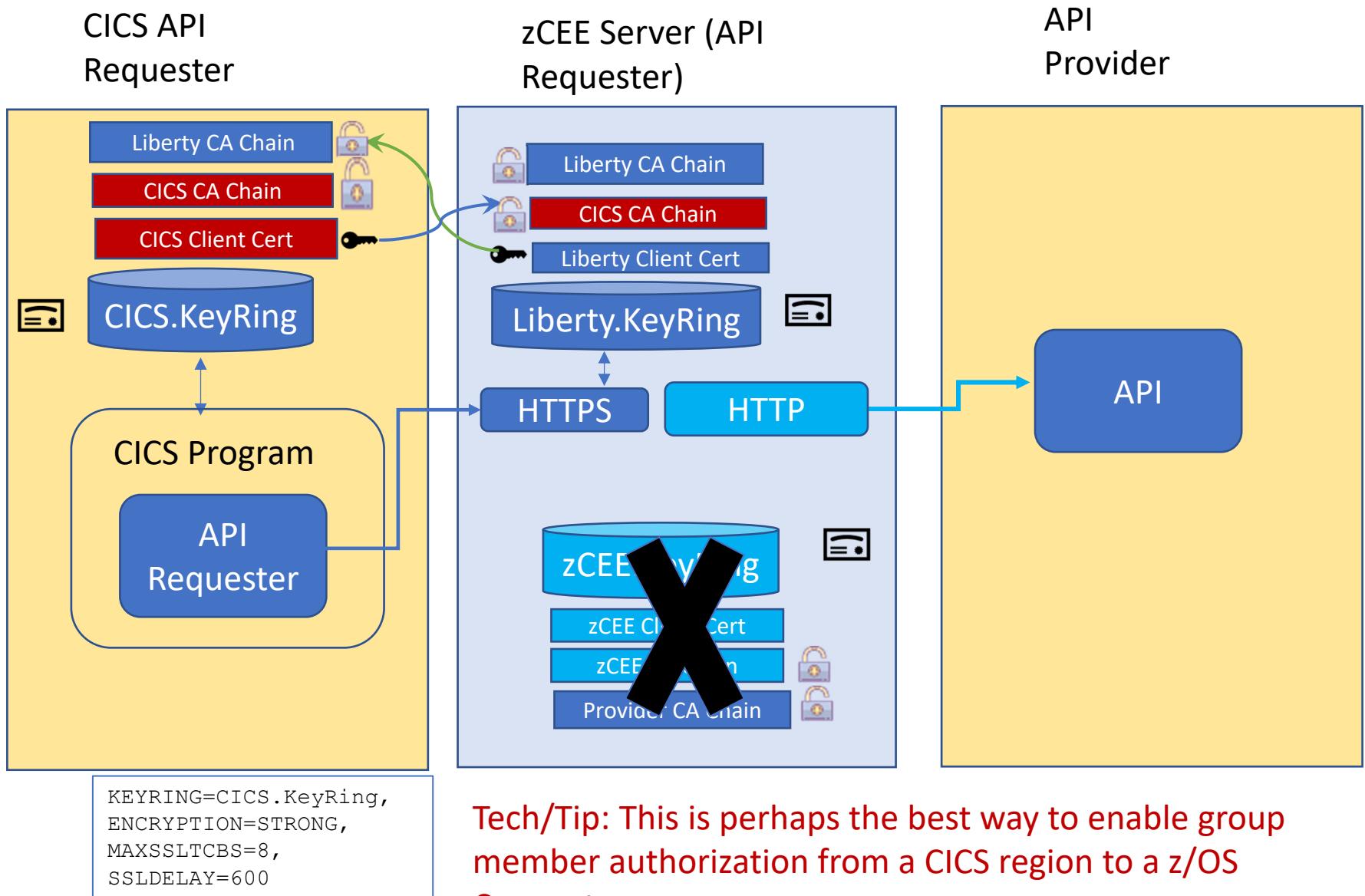
<ssl id="OutboundSSLSettings"
    keyStoreRef="OutboundKeyStore"
    trustStoreRef="OutboundKeyStore"/>

<keyStore id="OutboundKeyStore"
    location="safkeyring:///zCEE.KeyRing"
    password="password" type="JCERACFKS"
    clientKeyAlias="Liberty Client Cert"
    fileBased="false" readOnly="true" />
```

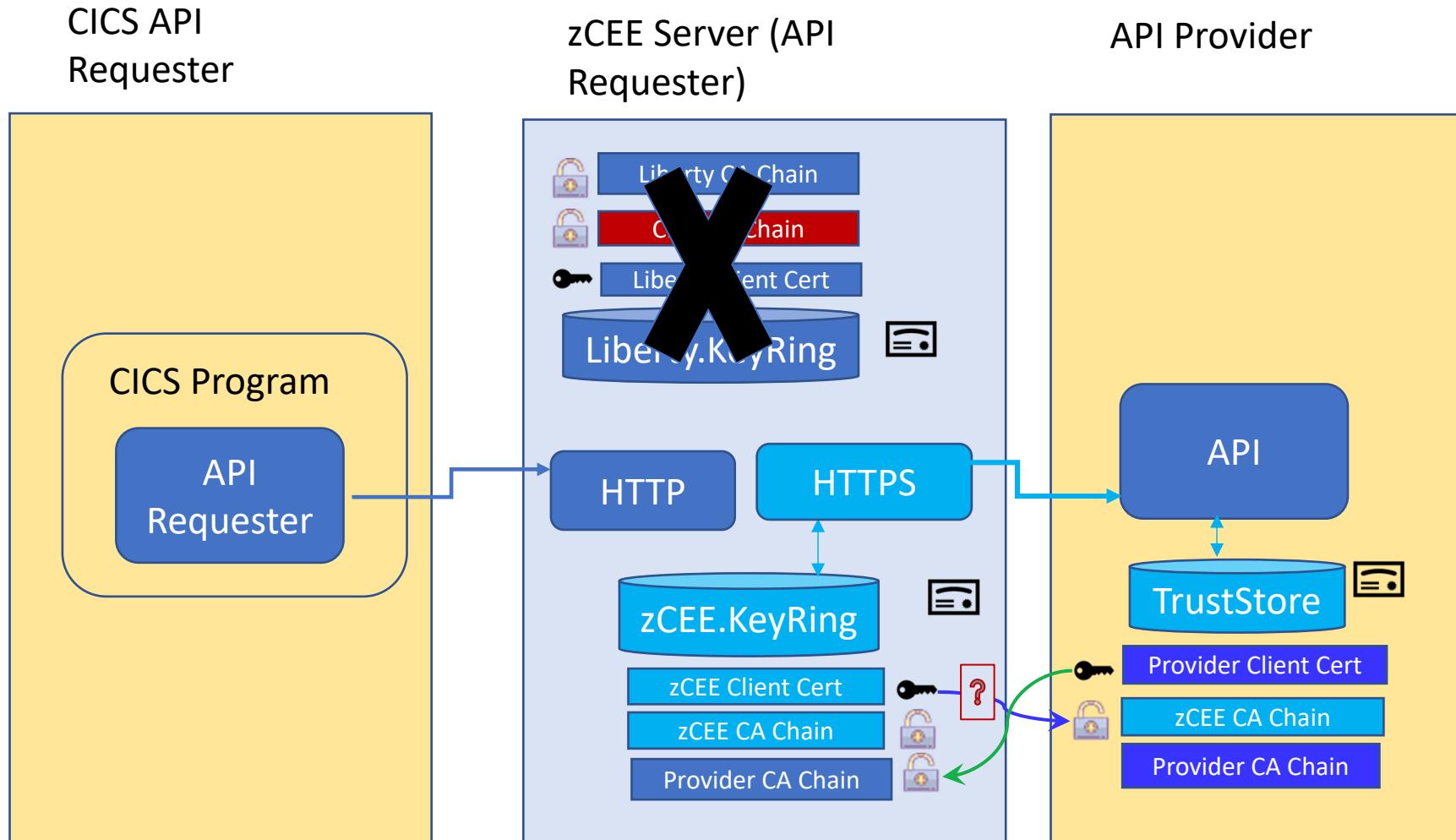
SSL repertoires

```
<zosconnect_authorizationServer sslCertRef="SSL repertoire"/>
<zosconnect_cicsIpicConnection sslCertRef="SSL repertoire"/>
<zosconnect_endpointConnect sslCertRef="SSL repertoire"/>
<zosconnect_zosConnectRestClient sslCertRef="SSL repertoire"/>
<zosconnect_zosConnectServiceRestClientConnection sslCertRef="SSL repertoire"/>
```

# TLS handshake scenario (CICS inbound handshake)

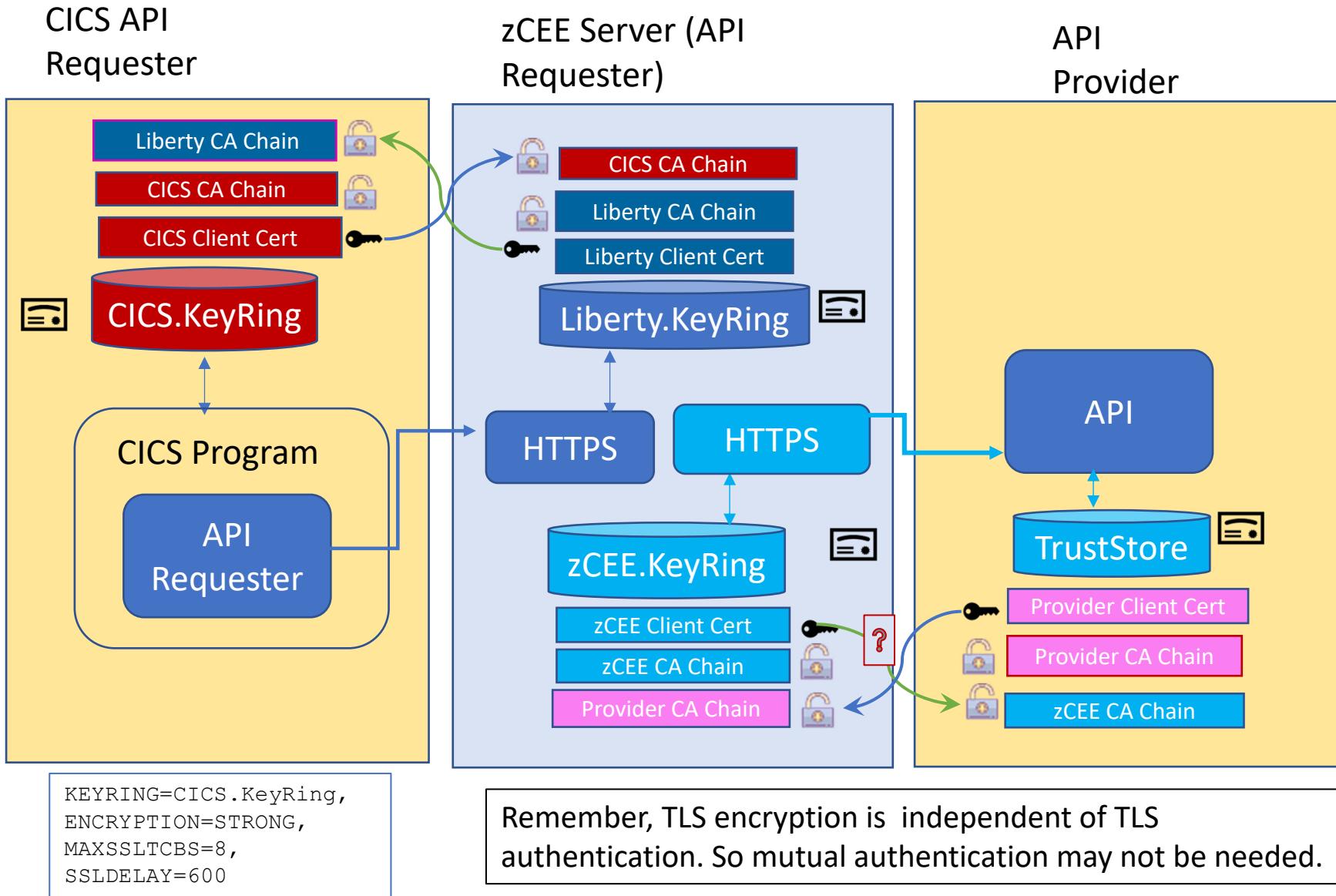


# TLS handshake scenario (outbound handshake)



Question if this really needed, TLS encryption is independent of TLS authentication.

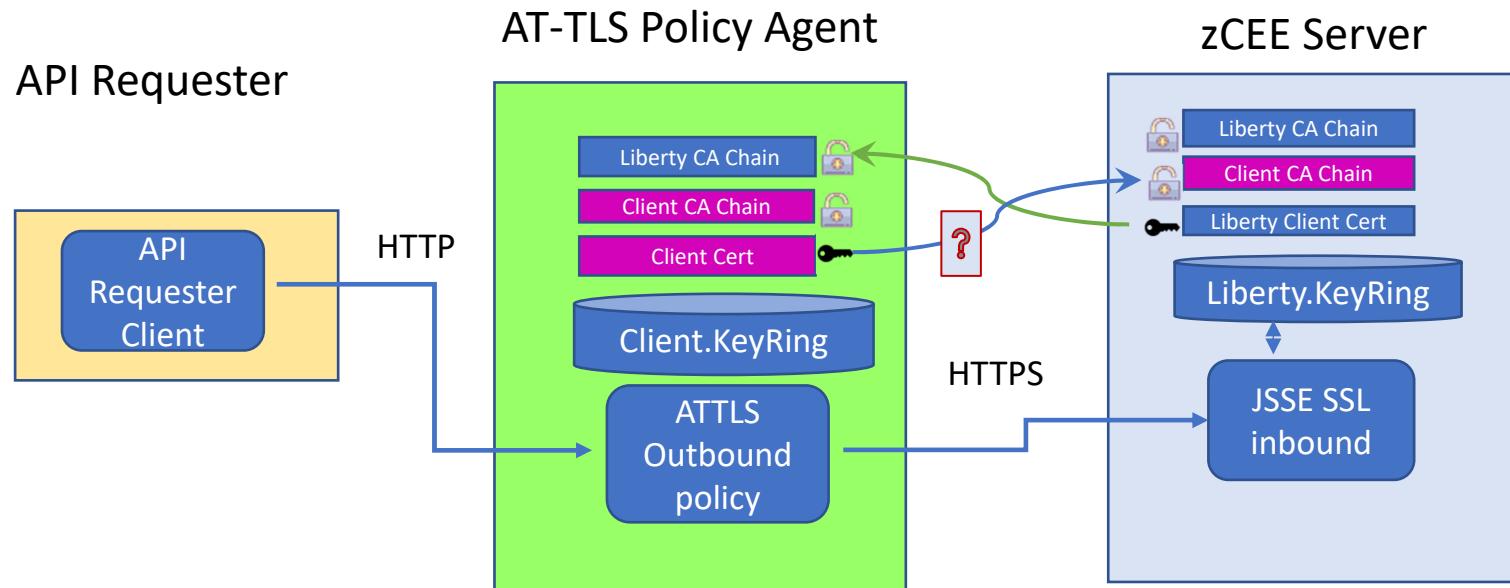
# TLS handshake scenario (multiple handshakes)





# AT-TLS - outbound policy handshake scenarios

Policy Agent uses an outbound policy and acts a surrogate TLS client



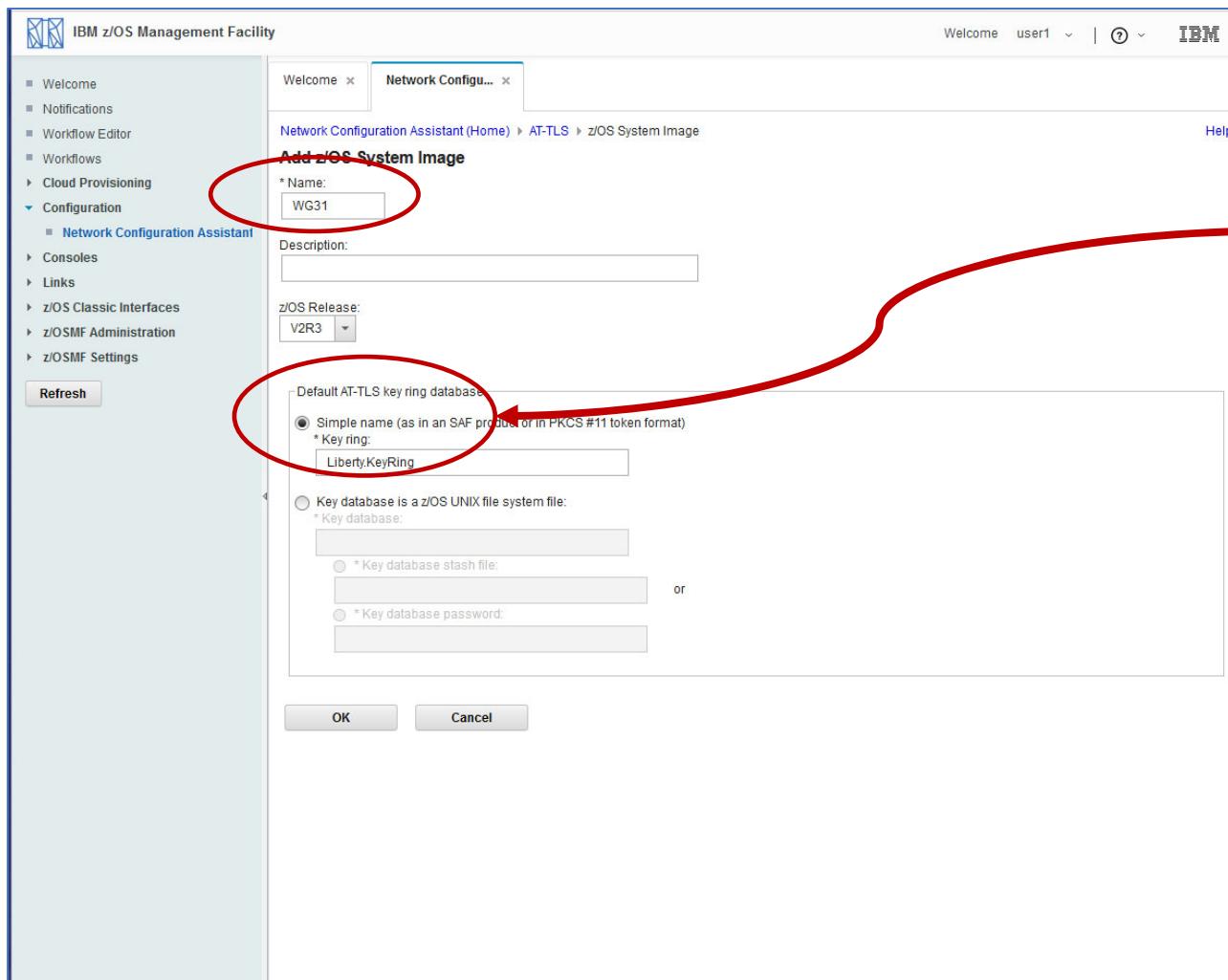
```
<zosconnect_apiRequesters idAssertion="ASSERT_ONLY">  
</zosconnect_apiRequesters>
```

- Certificate with a private key
- Certificate Authority (CA) certificate

Question if this really needed, remember TLS encryption is independent of TLS authentication.

# AT-TLS – setting a default key ring

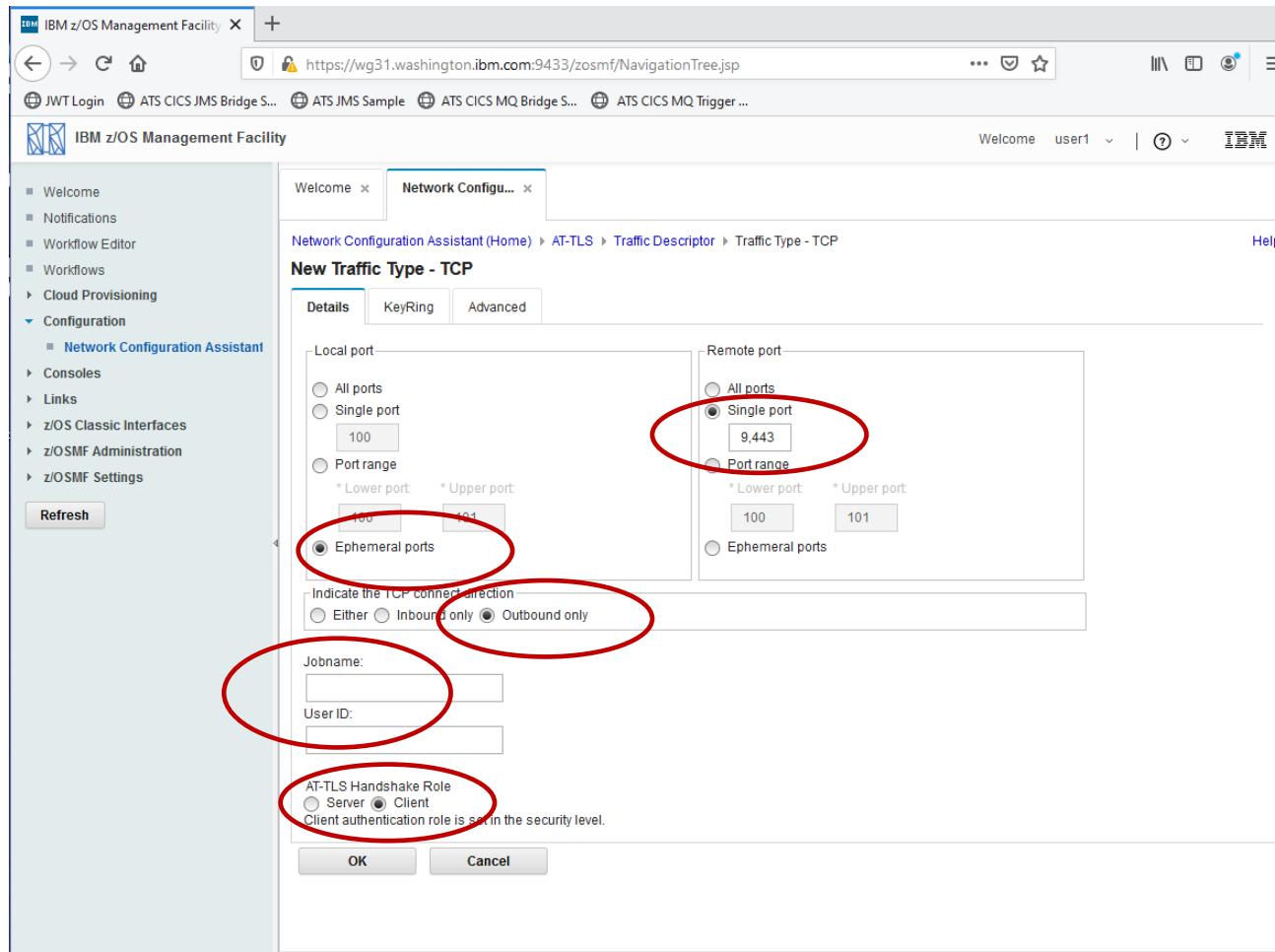
Using zOSMF Network Configuration Assistant to provide default Key Ring name



An SAF key ring name is specified as "identity/keyring". The current identity is used if the identity field is omitted.

# AT-TLS – configuring an outbound policy

Using zOSMF Network Configuration Assistant to configure traffic descriptor



# AT-TLS - setting a policy key ring

Using zOSMF Network Configuration Assistant to traffic descriptor Key Ring name

The screenshot shows the 'Network Configuration Assistant' interface under the 'AT-TLS' section for 'Traffic Descriptor'. The 'KeyRing' tab is active. A red oval highlights the 'Key ring:' dropdown, which contains three options:

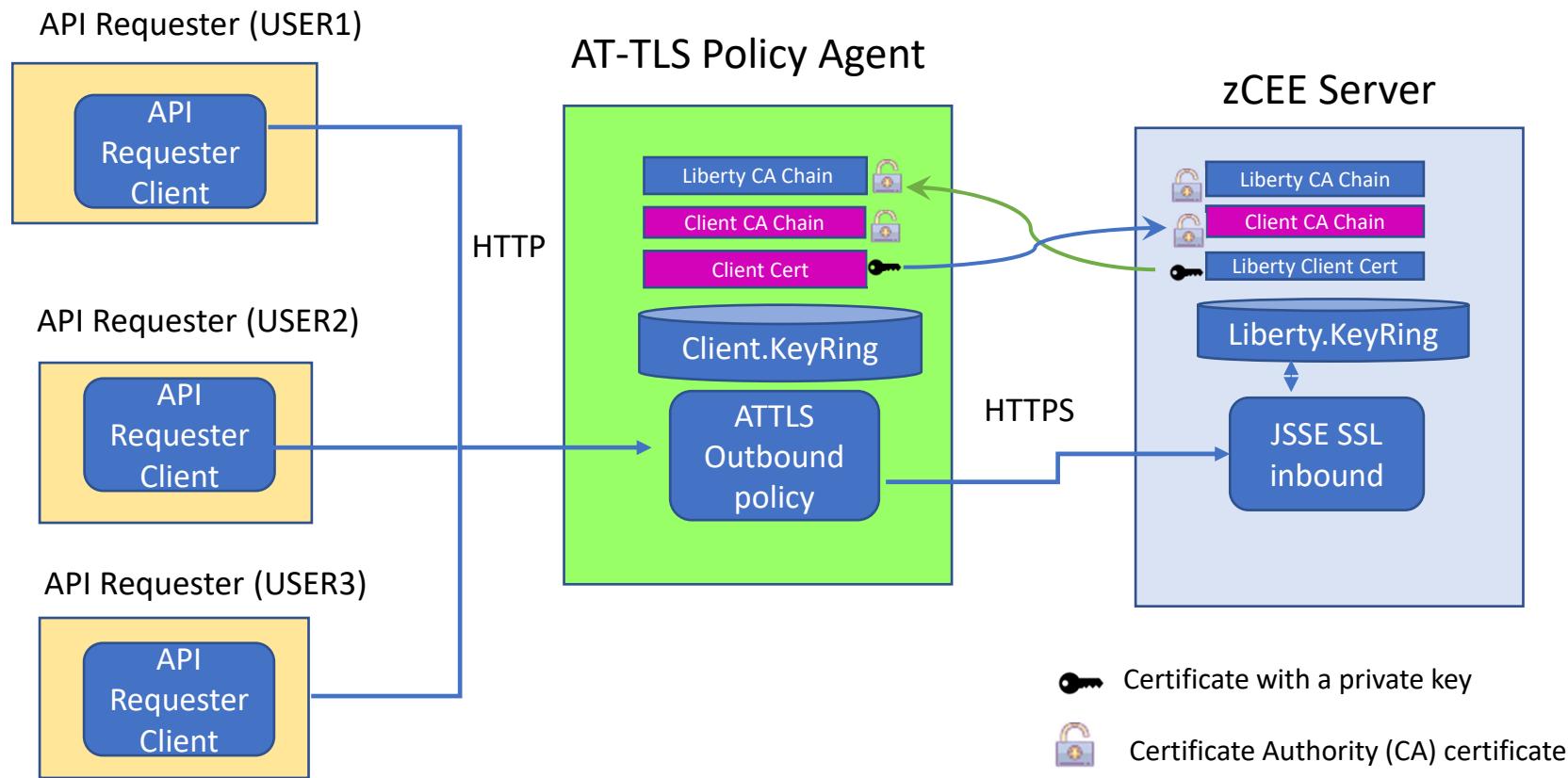
- Use the key ring database defined for the z/OS system image.
- Use a Simple name (as in a SAF product or in PKCS #11 Token format):  
\* Key ring:  
Liberty.KeyRing
- Use this z/OS UNIX file system key database:  
Key database:  
\* Key database stash file:  
\* Key database password:

A callout box to the right contains the following text:

An SAF key ring name is specified as "identity/keyring". The current identity is used if the identity is omitted.

# AT-TLS - outbound policy handshake scenario

Use of a common key ring name with multiple client identities

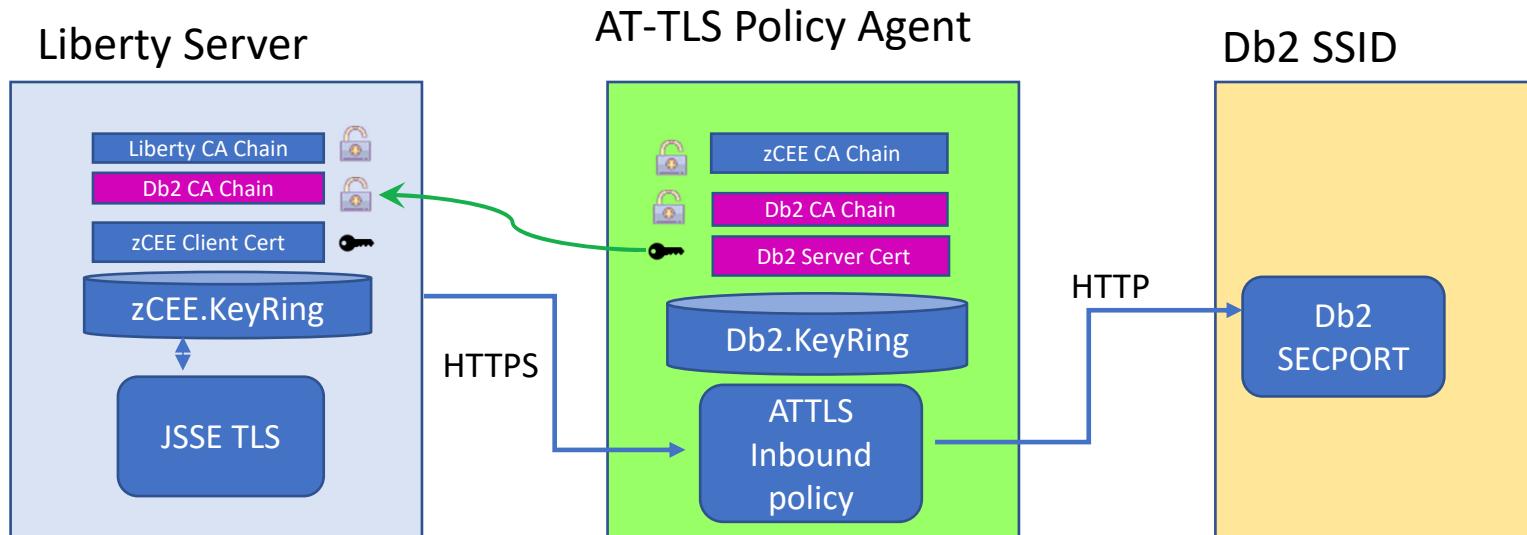


- Each user owns a keyring with the name Liberty.KeyRing.
- Each key ring has a different default client certificate for mutual authentication purposes.

This is a situation when AT-TLS mutual authentication has a benefit.

# AT-TLS - inbound policy handshake scenario (Db2)

Policy Agent uses an inbound policy and acts a surrogate TLS server

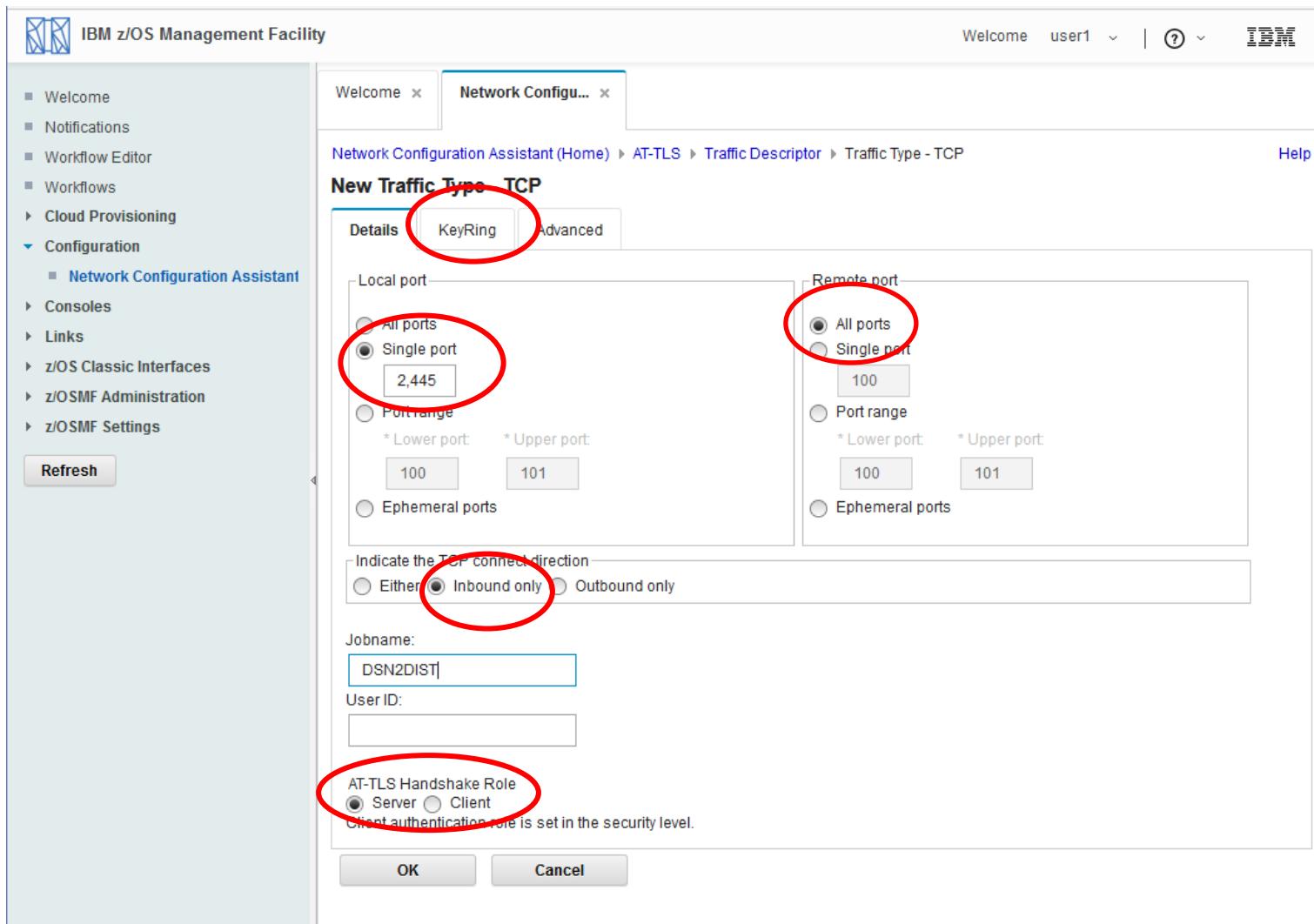


Note that DB2 is AT-TLS aware

- 🔑 Certificate with a private key
- 🔒 Certificate Authority (CA) certificate

# AT-TLS – configuring an inbound policy

Using zOSMF Network Configuration Assistant



# AT-TLS – configuring client authentication

Using zOSMF Network Configuration Assistant (mutual authentication)

The screenshot shows the IBM z/OS Management Facility interface. The left sidebar contains navigation links such as Welcome, Notifications, Workflow Editor, Workflows, Cloud Provisioning (Resource Management, Software Services), Configuration (Network Configuration Assistant, which is selected and highlighted in blue), Consoles, and Links (Shopz, Support for z/OS, WSC Flashes & Techdocs, z Systems, z/OS Basics Information Center, z/OS Home Page, z/OS Internet Library). Below these are sections for z/OS Classic Interfaces, z/OSMF Administration, and z/OSMF Settings. A Refresh button is at the bottom of the sidebar.

The main content area has tabs at the top: Welcome, Network Configu..., SSL Version 2 Ciphers, Client Authentication (which is selected and highlighted in blue), Tuning, Signature, Renegotiation, and Other. The Network Configuration Assistant tab is also highlighted in blue in the breadcrumb trail below the tabs.

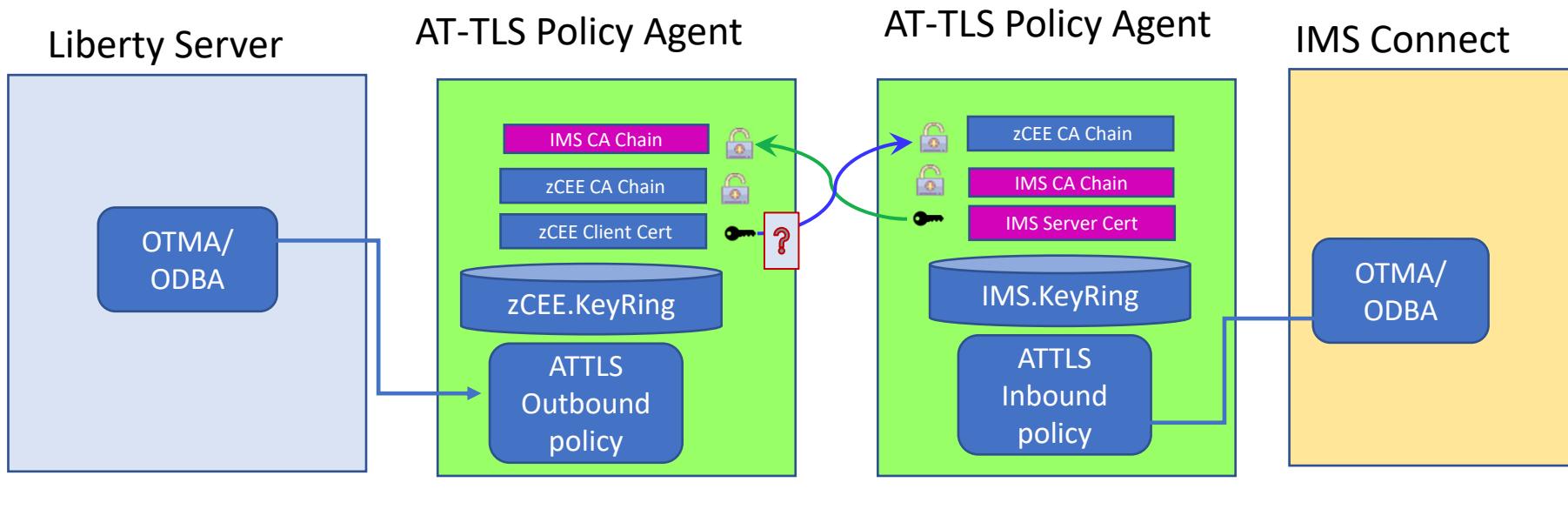
The main content area displays the "Advanced AT-TLS Settings" page. It includes sections for "Client authentication handling" (checkbox for "Use client authentication" is checked), "Indicate the level of client authentication" (radio buttons for Pass through, Full, Required, and SAF check, where Required is selected), and "Certificate revocation status checking" (checkbox for "Enabled" is checked, with a link to "Configure certificate revocation status checking"). At the bottom are OK and Cancel buttons.

© 2017, 2021 IBM Corporation

mitchj@us.ibm.com

# AT-TLS multiple policies handshake scenario (IMS)

Policy Agent uses both inbound and outbound policies and acts a surrogate TLS client with one and a TLS server with the other



🔑 Certificate with a private key

🔒 Certificate Authority (CA) certificate



Question if this really needed, TLS encryption is independent of TLS authentication. Use PassTickets to assert identity.

# Configuring TLS Encryption with JSSE

# Ciphers



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

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

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

For cipher details, see IBM SDK Java 8.0.0 Cipher Suites at URL

[https://www.ibm.com/support/knowledgecenter/SSYKE2\\_8.0.0/com.ibm.java.security.component.80.doc/security-component/jsse2Docs/ciphersuites.html](https://www.ibm.com/support/knowledgecenter/SSYKE2_8.0.0/com.ibm.java.security.component.80.doc/security-component/jsse2Docs/ciphersuites.html)

# Tech/Tip: Cipher Suite names

A CipherSuite is a suite of cryptographic algorithms used by a TLS connection. A suite comprises three distinct algorithms:

- The key exchange and authentication algorithm, used during the handshake
- The encryption algorithm, used to encipher the data
- The MAC (Message Authentication Code) algorithm, used to generate the message digest

There are several options for each component of the suite, but only certain combinations are valid when specified for a TLS connection. The name of a valid CipherSuite defines the combination of algorithms used. For example, the CipherSuite ***TLS\_RSA\_WITH\_AES\_128\_CBC\_SHA*** specifies:

- The RSA key exchange and authentication algorithm
- The AES encryption algorithm, using a 128-bit key and cipher block chaining (CBC) mode
- The SHA-1 Message Authentication Code (MAC)

Note				
To use some CipherSuites, the 'unrestricted' policy files need to be configured in the JRE. For more details of how policy files are set up in an SDK or JRE, see the <i>IBM SDK Policy files</i> topic in the <i>Security Reference for IBM SDK, Java Technology Edition</i> for the version you are using.				
Table 1. CipherSpecs supported by IBM MQ and their equivalent CipherSuites				
CipherSpec	Equivalent CipherSuite (IBM JRE)	Equivalent CipherSuite (Oracle JRE)	Protocol	FIPS 140-2 compatible
ECDHE_ECDSA_3DES_EDE_CBC_SHA256	SSL_ECDHE_ECDSA_WITH_3DES_EDE_CBC_SHA	TLS_ECDHE_ECDSA_WITH_3DES_EDE_CBC_SHA	TLS 1.2	yes
ECDHE_ECDSA_AES_128_CBC_SHA256	SSL_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256	TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256	TLS 1.2	yes
ECDHE_ECDSA_AES_128_GCM_SHA256	SSL_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256	TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256	TLS 1.2	yes
ECDHE_ECDSA_AES_256_CBC_SHA384	SSL_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384	TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384	TLS 1.2	yes
ECDHE_ECDSA_AES_256_GCM_SHA384	SSL_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384	TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384	TLS 1.2	yes
ECDHE_ECDSA_NULL_SHA256	SSL_ECDHE_ECDSA_WITH_NULL_SHA	TLS_ECDHE_ECDSA_WITH_NULL_SHA	TLS 1.2	no
ECDHE_ECDSA_RC4_128_SHA256	SSL_ECDHE_ECDSA_WITH_RC4_128_SHA	TLS_ECDHE_ECDSA_WITH_RC4_128_SHA	TLS 1.2	no
ECDHE_RSA_3DES_EDE_CBC_SHA256	SSL_ECDHE_RSA_WITH_3DES_EDE_CBC_SHA	TLS_ECDHE_RSA_WITH_3DES_EDE_CBC_SHA	TLS 1.2	yes
ECDHE_RSA_AES_128_CBC_SHA256	SSL_ECDHE_RSA_WITH_AES_128_CBC_SHA256	TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256	TLS 1.2	yes
ECDHE_RSA_AES_128_GCM_SHA256	SSL_ECDHE_RSA_WITH_AES_128_GCM_SHA256	TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256	TLS 1.2	yes
ECDHE_RSA_AES_256_CBC_SHA384	SSL_ECDHE_RSA_WITH_AES_256_CBC_SHA384	TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384	TLS 1.2	yes
ECDHE_RSA_AES_256_GCM_SHA384	SSL_ECDHE_RSA_WITH_AES_256_GCM_SHA384	TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384	TLS 1.2	yes
ECDHE_RSA_NULL_SHA256	SSL_ECDHE_RSA_WITH_NULL_SHA	TLS_ECDHE_RSA_WITH_NULL_SHA	TLS 1.2	no
ECDHE_RSA_RC4_128_SHA256	SSL_ECDHE_RSA_WITH_RC4_128_SHA	TLS_ECDHE_RSA_WITH_RC4_128_SHA	TLS 1.2	no

[https://www.ibm.com/support/knowledgecenter/SSFKSJ\\_9.1.0/com.ibm.mq.dev.doc/q113210.htm](https://www.ibm.com/support/knowledgecenter/SSFKSJ_9.1.0/com.ibm.mq.dev.doc/q113210.htm)

© 2017, 2021 IBM Corporation

mitchj@us.ibm.com

# CICS IPIC using TLS



The server.xml file is the key configuration file:

inquireSingle Service Configuration

Required Configuration

Enter the required configuration for this service.

Coded character set identifier (CCSID): 37

Connection reference: catalog

Optional Configuration

Enter the optional configuration for this service.

Transaction ID:

Transaction ID usage:

Overview Configuration

WG31

File Edit Settings View Communication Actions Window Help

I TCPIPS  
RESULT - OVERTYPE TO MODIFY  
Tcpipservice(CSCVINC)  
Openstatus( Open )  
Port(01493)  
Protocol(ipic)  
Ssltype(Ssl)  
Transid(CIIS)  
Authenticate(shaauthentic)  
Connections(00000)  
Backlog( 01024 )  
Maxdatalen( 000000 )  
Lrm( DFHISAPI )  
Privacy(Supported)  
Ciphers(3538392F3232)  
Host(ANY)  
Ipaddress(192.168.17.201)  
Hosttype(Any)  
Ipresolved(192.168.17.201)  
+ Ipfamily(Ipv4family)

SYSID=CICS APPLID=CICS53Z  
TIME: 13.12.07 DATE: 02/22/21  
PF 1 HELP 2 HEX 3 END 5 VAR 7 SBH 8 SFH 10 SB 11 SF  
01/012  
MA D  
Connected to remote server/host wg31 using lu/pool TCP00137 and port 23

Liberty Admin Center https://wg31.washington.ibm.com:944 ...

IBM Verse Coastal Federal Credit ... z/OS IBM Support Other Bookmarks

Server Config ipicSSLIDProp.xml Read only Close

Design Source

```
<server description="CICS IPIC ID propagation connections">
<!-- Enable features -->
<featureManager>
<feature>zosconnect:cicsService-1.0</feature>
</featureManager>
<zosconnect_cicsIpicConnection id="catalog"
host="wg31.washington.ibm.com"
port="1493"
zosConnectNetworkid="CSCVINC"
zosConnectApplid="CSCVINC"
transid="M10"
transidUsage="EIB_AND_MIRROR"
sslCertsRef="cicsSSLSettings"/>
</server>
```

Define IPIC/TLS connections to CICS

# Tech/Tip: Cipher Suite numbers (CICS TCPIPSERVICE):

Table 2. 2-character and 4-character cipher suite definitions for SSL V3, TLS V1.0, TLS V1.1, TLS V1.2, and TLS V1.3

2-character cipher number	4-character cipher number	Short name	Description <sup>1</sup>	FIPS 140-2	Base security level	Security level 3
					FMID >  HCPT440  <	JCPT441  <
00	0000	TLS_NULL_WITH_NULL_NULL	No encryption or message authentication and RSA key exchange	X	X	
01	0001	TLS_RSA_WITH_NULL_MD5	No encryption with MD5 message authentication and RSA key exchange	X	X	
02	0002	TLS_RSA_WITH_NULL_SHA	No encryption with SHA-1 message authentication and RSA key exchange	X	X	
03	0003	TLS_RSA_EXPORT_WITH_RC4_40_MD5	40-bit RC4 encryption with MD5 message authentication and RSA (export) key exchange	X	X	
04	0004	TLS_RSA_WITH_RC4_128_MD5	128-bit RC4 encryption with MD5 message authentication and RSA key exchange			X
05	0005	TLS_RSA_WITH_RC4_128_SHA	128-bit RC4 encryption with SHA-1 message authentication and RSA key exchange		X	^

[https://www.ibm.com/support/knowledgecenter/SSLTBW\\_2.4.0/com.ibm.zos.v2r4.gska100/csdcwh.htm](https://www.ibm.com/support/knowledgecenter/SSLTBW_2.4.0/com.ibm.zos.v2r4.gska100/csdcwh.htm)

© 2017, 2021 IBM Corporation

mitchj@us.ibm.com



# MQ JMS using TLS

The server.xml file is the key configuration file:

\*twoway Service X

### Service Project Editor: Configuration

Required Configuration

Enter the required configuration for this service.

Connection factory JNDI name: jms/qmgrCf

Request destination JNDI name: jms/requestQueue

Reply destination JNDI name: jms(replyQueue)

Wait interval: 3000

MQMD format: MQSTR

Coded character set identifier (CCSID): 37

Is message persistent:

Reply selection: msgIDToCorrelID

Expiry: -1

LIBERTY.SSL.SVRCONN - Properties

General  
Extended  
MCA  
Exits  
**SSL**  
Statistics

**SSL**

CipherSpec  
Set message security for this end of the channel  
SSL Cipher Spec: **TLS\_RSA\_WITH\_AES\_256\_CBC\_SHA256**  
TLS 1.2, 256-bit Secure Hash Algorithm, 256-bit AES encryption

Accept only certificates with Distinguished Names matching these values:

SSL Authentication: Required  
Certificate label:

Apply OK Cancel

Server Config

mqClientTLS.xml

Design Source

```
<server description="MQ Service Provider">
  <featureManager>
    <feature>zosconnect:mqService-1.0</feature>
  </featureManager>
  <variable name="wmqJmsClient.rar.location"
            value="/u/johnson/jca/wmq.jmsra.rar"/>
  <wmqJmsClient nativeLibraryPath="/usr/lpp/mqm/V9R1M1/java/lib"/>
  <zosconnect_services>
    <service name="mqPutService">
      <property name="useCallerPrincipal" value="true"/>
    </service>
  </zosconnect_services>
  <connectionManager id="ConMgr1" maxPoolSize="5"/>
  <jmsConnectionFactory id="qmgrCF" jndiName="jms/qmgrCF"
    connectionManagerRef="ConMgr1">
    <properties.wmqJMS transportType="CLIENT"
      queueManager="ZMQ1"
      channel="LIBERTY.SSL.SVRCONN"
      hostName="wg31.washington.ibm.com"
      sslcipherSuite="SSL_RSA_WITH_AES_256_CBC_SHA256"
      port="1422" />
  </jmsConnectionFactory>
  <jmsQueue id="q1" jndiName="jms/default">
    <properties.wmqJms
      baseQueueName="ZCEE.DEFAULT.MQZCEE.QUEUE"
      CCSID="37"/>
  </jmsQueue>
</server>
```



# Tech/Tip: API Requester - HTTP v HTTPS

## MVS Batch and IMS with AT-TLS

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

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

## CICS URIMAPs

```
WG31
File Edit Settings View Communication Actions Window Help
OVERTYPE TO MODIFY
CEDA ALter UriMap( BAQRIMP )
UriMap      : BAQRIMP
Group       : SYSGRP
DEscription ==> URIMAP for z/OS Connect EE server
Status      ==> Enabled      Enabled | Disabled
Usage       ==> Client       Server | Client | Pipeline |
                         | Jvmserver
UNIVERSAL RESOURCE IDENTIFIER
SCHEME      ==> HTTP        HTTP | HTTPS
Port        ==> 09120       No | 1-65535
HOST        ==> wg31.washington.ibm.com
==>
==>
Path        ==> /
(Mixed Case) ==>
==>
==>
==>
+ OUTBOUND CONNECTION POOLING
SYSID=CICS APPL
PF 1 HELP 2 COM 3 END      6 CRSR 7 SBH 8 SFH 9 MSG 10 SB 11
MA C
Connected to remote server/host wg31 using lu/pool TCP00133 and port 23
```

```
WG31
File Edit Settings View Communication Actions Window Help
OVERTYPE TO MODIFY
CICS RELEASE = 0710
CEDA ALter UriMap( BAQRIMP )
UriMap      : BAQRIMP
Group       : SYSGRP
DEscription ==> URIMAP for z/OS Connect EE server
Status      ==> Enabled      Enabled | Disabled
Usage       ==> Client       Server | Client | Pipeline | Atom
                         | Jvmserver
UNIVERSAL RESOURCE IDENTIFIER
SCHEME      ==> HTTPS       HTTP | HTTPS
Port        ==> 09443       No | 1-65535
HOST        ==> wg31.washington.ibm.com
==>
Path        ==> /
(Mixed Case) ==>
==>
==>
==>
+ OUTBOUND CONNECTION POOLING
SYSID=CICS APPLID=CICSS53Z
PF 1 HELP 2 COM 3 END      6 CRSR 7 SBH 8 SFH 9 MSG 10 SB 11 SF 12 CNCL
MA C
Connected to remote server/host wg31 using lu/pool TCP00133 and port 23
13/022
```

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

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

mitchj@us.ibm.com

© 2017, 2021 IBM Corporation

# Persistent connections



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

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

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

# TLS sessions



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

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

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

# Tech/Tip: SSL issues

Some SSL related messages will occur in the spool output of the server, i.e.,

- **CWPKI0022E: SSL HANDSHAKE FAILURE: A signer with SubjectDN CN=USER3 D. Client, OU=LIBERTY, O=IBM was sent from the target host. The signer might need to be added to local trust store safkeyring://Liberty.KeyRing, located in SSL configuration alias DefaultSSLSettings. The extended error message from the SSL handshake exception is: PKIX path building failed: com.ibm.security.cert.IBMCertPathBuilderException: unable to find valid certification path to requested target**

This message is indicating a personal certificate was presented in a TLS handshake and there was no corresponding certificate authority certificate connect to the local trust store (e.g., key ring). This can occur either for client connecting to the server or an API requester request going to an API provider.

Identify the certificate authority which signed this personal certificate and connect it to the keyring with usage CERTAUTH.

# Tech/Tip: SSL issues

With a few exceptions, most of TLS errors may require a review of a trace.

I use the *traceSpecification* as shown below:

```
<logging traceSpecification="com.ibm.ws.security.*=all:SSLChannel=all:SSL=all:zosConnectSaf=all"/>
```

This will generate a *trace.out* file in the *logs* subdirectory. This trace will provide details about the key ring and certificates involved in the handshake. There is a wealth of information about the flow between the client and server endpoints. Review this trace for exceptions. The following exceptions are some of the most commonly experienced.

- ***Error occurred during a read, exception:javax.net.ssl.SSLHandshakeException: null cert chain***

This exception occurs when the server configuration set to require client certificates (*clientAuthentication="true"*) and the client had no certificate to provide and no alternative authentication method was available.

- ***Error occurred during a read, exception:javax.net.ssl.SSLException: Received fatal alert: bad\_certificate error (handshake), vc=1083934466***

***Caught exception during unwrap, javax.net.ssl.SSLException: Received fatal alert: bad\_certificate***

This is usually caused when the client certificate presented to the server did not have a valid CA certificate for the client's personal certificate in the server's trust store key ring.

# Tech/Tip: SSL issues

- ***FFDC1015I: An FFDC Incident has been created: "java.io.IOException: Failed validating certificate paths com.ibm.ws.ssl.config.WSKeyStore\$1 do\_getKeyStore" at ffdc\_19.12.04\_20.51.47.0.log***

This can occur when the CA certificate used to sign the server's personal certificate was not connected to the server's local trust store (key ring on z/OS).

- ***java.io.IOException: IOException invoking https://132.25.33.351:9443/employees/John?validated=true: HTTPS hostname wrong: should be <132.25.33.351>***

In this situation the endpoint for the outbound API request was configured to use an IP address rather than a hostname. This should not be an issue unless an exchange of digital certificates is required.

The trace showed that during the handshake process the outbound API provider server's certificate had a common name (CN) which specified the hostname of the TCPIP stack where the API resided. This hostname was not known (e.g., DNS-resolvable) on the TCPIP stack where the z/OS Connect server was executing. This meant that communications back to the API requester's TCPIP stack based on the hostname was not possible which caused the IO exception. The best solution would be to use the host name in the server.xml configuration rather than the IP address and either add an entry to the local TCPIP stack's hostname (e.g., hosts) file for the IP address and hostname or add an entry to the DNS servers used by this TCPIP stack.

# Tech/Tip: Enabling hardware cryptography key rings

jvm.options

```
-Djava.security.properties=${server.config.dir}/java.security
```

java.security

```
security.provider.1=com.ibm.crypto.hdwrCCA.provider.IBMJCECCA  
security.provider.2=com.ibm.crypto.provider.IBMJCE  
security.provider.3=com.ibm.jsse2.IBMJSSEProvider2  
security.provider.4=com.ibm.security.jgss.IBMJGSSProvider  
.....
```

Enabling the IBMJCECCA provider

```
<keyStore id="CellDefaultKeyStore"  
         location="safkeyringhw://Liberty.KeyRing"  
         password="password" type="JCECCARACFKS"  
         fileBased="false" readOnly="true" />
```

Enabling the IBMJCEHYBRID provider

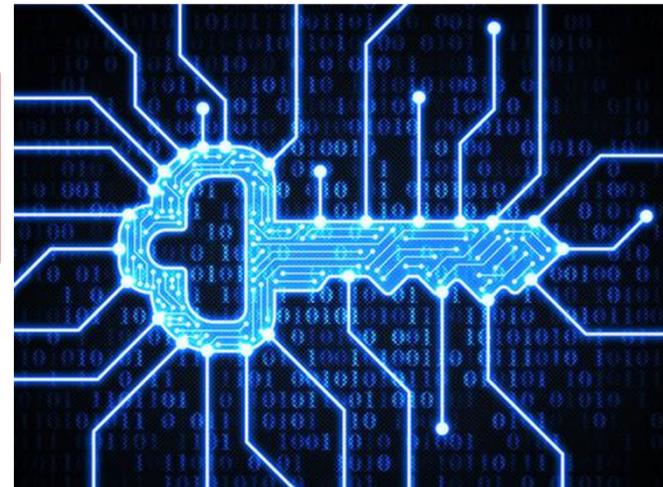
```
<keyStore id="CellDefaultKeyStore"  
         location="safkeyringhybrid://Liberty.KeyRing"  
         password="password" type="JCEHYBRIDRACFKS"  
         fileBased="false" readOnly="true" />
```

See URL <https://www.ibm.com/support/pages/node/6209109> for details on implementing IBMJCECCA and IBMJCEHYBRID hardware encryption providers

# General security terms or considerations

- Identifying who or what is requesting access (**Authentication**)
  - Basic Authentication
  - Mutual Authentication using TLS
  - 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 generating/sending a digital signature)

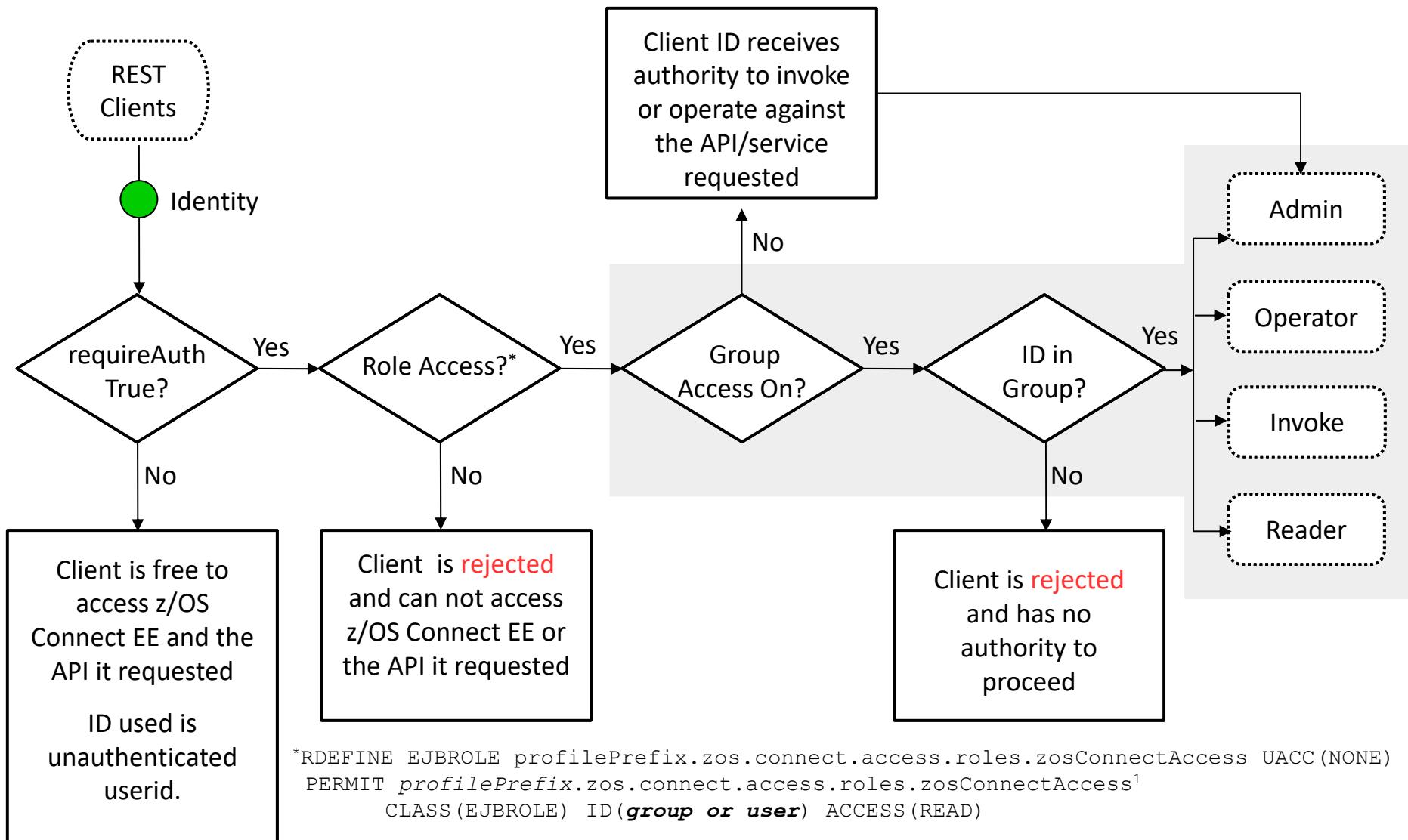
- Controlling access (**Authorization**)
  - Is the authenticated identity authorized to access to z/OS Connect
  - Is the authenticated identity authorized to access a specific API, Services, etc.



## Authorization

Once we have an identity, then what?

# Security flow with z/OS Connect EE



*\*where **profilePrefix** is value of the **profilePrefix** attribute in the **safCredentials** configuration element*

*<sup>1</sup>as specified in the **web.xml** file*

# **z/OS Connect Authorization Functions**



**Operations** - Ability to perform all z/OS Connect EE operations and actions except for function *Invoke*. The following operations/actions are allowed:

## **APIs:**

- *To obtain a list of all APIs (GET).*\*
- For a specific API, get its details and API Swagger document (GET) and *deploy (POST)*\*, update (PUT), start(PUT), stop(PUT), and delete(DELETE) it.

## **Services:**

- *To obtain a list of all services or statistics for all services (GET).*\*
- For a specific service, get its details, request and response schemas, statistics (GET) and *deploy(POST)*\*, update(PUT), start(PUT), stop(PUT), and delete(DELETE) it.

## **API Requesters:**

- *To obtain a list of all API requesters (GET).*\*
- For a specific API requester, get its details (GET) and *deploy (POST)*\*, update(PUT), start(PUT), stop(PUT), and delete(DELETE) it.

\*These APIs use either the POST or GET method to invoke the REST APIs whose URIs have no path parameter. Therefore, the name of the API, or service or API Requester is not available. For authorization, only the default or global groups list can be used since no specific group list can be determined (for deployment, the name is embedded in the archive file).

# **z/OS Connect Authorization Levels**



**Reader** - Ability for:

**APIs:**

- *To obtain a list of all APIs (GET) . \**
- For a specific API, get its details and API Swagger document (GET).

**Services:**

- *To obtain a list of all services (GET). \**
- For a specific service, get its details and request and response schemas (GET).

**API Requesters:**

- *To obtain a list of all API requesters (GET). \**
- For a specific API requester, get its details (GET) .

**Invoke** - Ability to invoke user APIs, services and/or API requesters (POST,PUT,GET,DELETE,+).

**Admin** - All z/OS Connect EE actions are allowed, including all corresponding *Operations*, *Invoke*, and *Reader* actions configured for the same z/OS Connect resource.

\*These APIs use either the POST or GET method to invoke the REST APIs whose URLs have no path parameter. Therefore, the name of the API, service or API Requester is not available. For authorization, only the default or global groups list since no specific group list can be determined (for deployment, the name is embedded in the archive file).



## z/OS Connect administration API

Interface providing meta-data and life-cycle operations for z/OS Connect services, APIs and API requesters.

### APIs : Operations for working with APIs

Show/Hide | List Operations | Expand Operations

GET	/apis	Returns a list of all the deployed z/OS Connect APIs
POST	/apis	Deploys a new API into z/OS Connect
DELETE	/apis/{apiName}	Undeploys an API from z/OS Connect
GET	/apis/{apiName}	Returns detailed information about a z/OS Connect API
PUT	/apis/{apiName}	Updates an existing z/OS Connect API

### Services : Operations for working with services

Show/Hide | List Operations | Expand Operations

GET	/services	Returns a list of all the deployed z/OS Connect services
POST	/services	Deploys a new service into z/OS Connect
DELETE	/services/{serviceName}	Undeploys a service from z/OS Connect
GET	/services/{serviceName}	Returns detailed information about a z/OS Connect service
PUT	/services/{serviceName}	Updates an existing z/OS Connect service
GET	/services/{serviceName}/schema/{schemaType}	Returns the request or response schema for a z/OS Connect service

### API Requesters : Operations that work with API Requesters.

Show/Hide | List Operations | Expand Operations

GET	/apiRequesters	Returns a list of all the deployed z/OS Connect API Requesters
POST	/apiRequesters	Deploys a new API Requester into z/OS Connect and invoke an API Requester call
DELETE	/apiRequesters/{apiRequesterName}	Undeploys an API Requester from z/OS Connect
GET	/apiRequesters/{apiRequesterName}	Returns the detailed information about a z/OS Connect API Requester
PUT	/apiRequesters/{apiRequesterName}	Updates an existing z/OS Connect API Requester

# **z/OS Connect RESTful Administrative APIs Security**



z/OS Connect uses group security for controlling authorization for accessing APIs. There are sets of default global groups for functional roles are configured in a `zosConnectManager` configuration element as shown below:

```
<zosconnect_zosConnectManager  
    globalInterceptorsRef="interceptorList_g"  
    globalAdminGroup="GMADMIN" globalOperationsGroup="GMOPERS"  
    globalInvokeGroup="GMINVOKE" globalReaderGroup="GMREADR"/>
```

There are four classes of groups available controlling z/OS Connect functions, administration, operations, invoking and reader in our server. An authenticated identity membership in one or more of these groups provides access to the corresponding function to that identity. There is also a way to provide an alternative set of groups for functional roles for specific APIs, services, and API requesters in subordinate configuration elements in our server.

```
<zoscConnectAPI name="cscvinc"  
    adminGroup="CSCADMIN" operationsGroup="CSCOPERS"  
    invokeGroup="CSCINVKE" readerGroup="CSCREADR"/>  
  
<service name="cscvincSelectService"  
    adminGroup="CSCADMIN" operationsGroup="CSCOPERS"  
    invokeGroup="CSCINVKE" readerGroup="CSCREADR"/>  
  
<apiRequester name="cscvinc_1.0.0"  
    adminGroup="CSCADMIN" operationsGroup="CSCOPERS"  
    invokeGroup="CSCINVKE" readerGroup="CSCREADR"/>
```

# Interceptor - server XML example



```
<zosconnect_zosConnectManager  
    globalInterceptorsRef="interceptorList_g"  
    globalAdminGroup="GMADMIN"  
    globalOperationsGroup="GMOPERS"  
    globalInvokeGroup="GMINVOKE"  
    globalReaderGroup="GMREADR"/>  
  
<zosconnect_authorizationInterceptor id="auth"/>  
<zosconnect_auditInterceptor id="audit"/>  
<zosconnect_zosConnectInterceptors id="interceptorList_g"  
    interceptorRef="auth"/>  
<zosconnect_zosConnectInterceptors id="interceptorList_a"  
    interceptorRef="auth,audit"/>  
  
<zosconnect_zosConnectAPIs>  
    <zosConnectAPI name="catalog"  
        runGlobalInterceptorsRef="true"  
        adminGroup="aapigrp1,aapigrp2"  
        operationsGroup="oapigrp1,oapigrp2"  
        invokeGroup="iapigrp1,oapigrp2"  
        readerGroup="rapigrp1,rapigrp2"/>  
</zosconnect_zosConnectAPIs>  
  
<zosconnect_apiRequesters>  
    <apiRequester name="cscvincapi_1.0.0"  
        runGlobalInterceptorsRef="false"  
        interceptorsRef="interceptorList_a"  
        adminGroup="aaprgrp1,aaprgrp2"  
        operationsGroup="oaprgrp1,oaprgrp2"  
        invokeGroup="iaprgrp1,oaprgrp2"  
        readerGroup="raprgrp1,raprgrp2"/>  
</zosconnect_apiRequesters>  
  
<zosconnect_services>  
    <service id="selectByEmployee" name="selectEmployee"  
        runGlobalInterceptorsRef="false"  
        interceptorsRef="interceptorList_a"  
        adminGroup="asrvgrp1,asrvgrp2"  
        operationsGroup="osrvgrp1,osrvgrp2"  
        invokeGroup="isrvgrp1,isrvgrp2"  
        readerGroup="rsrvgrp1,rsrvgrp2"/>  
</zosconnect_services>
```

ADDGROUP GMADMIN OMVS (AUTOGID)  
ADDGROUP GMINVOKE OMVS (AUTOGID)  
CONNECT FRED GROUP (GMADMIN)  
CONNECT USER1 GROUP (GMINVOKE)

Global interceptor list –  
authorization  
interceptor only

Alternative interceptor  
list – authorization and  
audit interceptors

Avoid duplication of  
interceptors

Note that these are  
**z/OS Connect**  
**configuration**  
**elements.**  
**Documented in the**  
**z/OS Connect KC**

# Tech/Tip: Server XML example – combining TLS/AUTH interceptor

```
<zosconnect_zosConnectManager  
    requireAuth="true"  
    requireSecure="true"  
    globalInterceptorsRef="interceptorList_g"  
    globalAdminGroup="GMADMIN"  
    globalOperationsGroup="GMOPERS"  
    globalInvokeGroup="GMINVOKE"  
    globalReaderGroup="GMREADR"/>  
  
<zosconnect_authorizationInterceptor id="auth"/>  
<zosconnect_zosConnectInterceptors id="interceptorList_g"  
    interceptorRef="auth"/>  
  
<zosconnect_apiRequesters>  
    <apiRequester name="cscvincapi_1.0.0"  
        requireSecure="false"  
        invokeGroup="iaprgrp1"/>  
</zosconnect_apiRequesters>
```

Global TLS security and authentication are enabled.

TLS security is disabled for this API requester archive artifact. Avoiding the HTTP 302 REDIRECT error.

This configuration would allow a MVS batch job to authenticate to z/OS Connect and use HTTP for the protocol. Only authorization identities which are members of groups identified as administrators or invokers would be authorized to invoke this API requester.

**F BAQSTRT,ZCON,CLEARSAFCACHE**

# Example of z/OS Connect Authorization Levels



```
<zosconnect_zosConnectManager
    globalInterceptorsRef="interceptorList_g"
    globalAdminGroup="GMADMIN" globalOperationsGroup="GMOPERS"
    globalInvokeGroup="GMINVOKE" globalReaderGroup="GMREADER"/>

<zosconnect_zosConnectAPIs>
    <zosConnectAPI name="cscvinc"
        adminGroup="CSCADMIN" operationsGroup="CSCOPERS"
        invokeGroup="CSCINVKE" readerGroup="CSCREADR"/>
    <zosConnectAPI name="db2employee"
        adminGroup="DB2ADMIN" operationsGroup="DB2OPERS"
        invokeGroup="DB2INVKE" readerGroup="DB2READR"/>
</zosconnect_zosConnectAPIs>

<zosconnect_services>
    <service name="cscvincSelectService"
        adminGroup="CSCADMIN" operationsGroup="CSCOPERS"
        invokeGroup="CSCINVKE" readerGroup="CSCREADR"/>
    <service name="selectEmployee"
        adminGroup="DB2ADMIN" operationsGroup="DB2OPERS"
        invokeGroup="DB2INVKE" readerGroup="DB2READR"/>
</zosconnect_services>

<zosconnect_apiRequesters>
    <apiRequester name="cscvincSelectService"
        adminGroup="CSCADMIN" operationsGroup="CSCOPERS"
        invokeGroup="CSCINVKE" readerGroup="CSCREADR"/>
    <apiRequester name="selectEmployee"
        adminGroup="DB2ADMIN" operationsGroup="DB2OPERS"
        invokeGroup="DB2INVKE" readerGroup="DB2READR"/>
</zosconnect_apiRequesters>
```

When groups are specified for zosConnectAPI, service, or apiRequester configuration elements, the global groups are ignored for certain functions. Other functions, e.g., deploy, get a list or service statistics, only use the global groups.

# **z/OS Connect Authorization Summary**



- Only members of groups GMADMIN, GMOPERS or GMREADER can connect to a z/OS server from the API toolkit.
- Only members of groups GMADMIN or GMOPERS can deploy new z/OS Connect API, service or API requester artifacts.
- Members of groups GMADMIN, GMOPERS, DB2ADMIN or DB2OPERS can not manage (e.g., change, stop or delete) z/OS Connect artifacts *managed* by group CSCOPERS or CSCADMIN.
- Members of groups GMADMIN, GMOPERS, CSCADMIN or CSCOPERS can not manage (e.g., change, stop or delete) z/OS Connect artifacts *managed* by group DB2OPERS or DB2ADMIN.
- Only members of group CSCADMIN, CSCINV, DB2ADMIN or DB2INVKE can invoke the artifacts defined in the subordinate element:
  - Members of group CSCADMIN or CSCVINKE can invoke artifacts managed by CSCINVKE
  - Members of group DB2ADMIN or DB2INVKE can invoke artifacts managed by DB2INVKE
  - Members of groups GMADMIN or GMINVOKE can not invoke any artifacts protected by these specific subordinate groups.

# Tech-Tips: z/OS Connect Authorization Levels



Identity	Group
USER11	GMADMIN
USER12	GMOPERS
USER13	GMINVOKE
USER14	GMREADER
USER21	CSCADMIN
USER22	CSCOPERS
USER23	CSCINVKE
USER24	CSCREADR
USER31	DB2ADMIN
USER32	DB2OPERS
USER33	DB2INVKE
USER34	DB2READR

# Tech-Tip: z/OS Connect Authorization Deploy Sample



## Deploy a new service as USER11

```
c:\z\passticket>curl -X POST --user user11:Z4GG7V4H --data-binary @cscvincSelectService.sar --header "Content-Type: application/zip" --insecure https://wg31.washington.ibm.com:9443/zosConnect/services {"zosConnect": {"serviceName": "cscvincSelectService", "serviceDescription": "", "serviceProvider": "CICS-1.0", "version": "1.0.0", "serviceURL": "https://wg31.washington.ibm.com:9443/zosConnect/services/cscvincSelectService", "serviceInvokeURL": "https://wg31.washington.ibm.com:9443/zosConnect/services/cscvincSelectService?action=invoke", "dataXformProvider": "zosConnectWVXform-1.0", "serviceStatus": "Started"}, "cscvincSelectService": {"transidUsage": "EIB_ONLY", "transid": "CSMI", "program": "CSCVINC", "connectionRef": "cscvinc", "ccsid": "37"}}
```

This service is now managed by members of CSCADMIN and CSCOPERS

```
c:\z\passticket>curl -X PUT --user user11:V4BTUHV5 --insecure https://wg31.washington.ibm.com:9443/zosConnect/services/cscvincSelectService?status=stopped {"errorMessage": "BAQR0435W: The zosConnectAuthorization interceptor encountered an error while processing a request.", "errorDetails": "BAQR0409W: User user11 is not authorized to perform the request."}
```

```
c:\z\passticket>curl -X DELETE --user user11:V4BTUHV5 https://wg31.washington.ibm.com:9443/zosConnect/services/cscvincSelectService {"errorMessage": "BAQR0434W: The zosConnectAuthorization interceptor encountered an error while processing a request for service cscvincSelectService. The user user11 is not authorized to perform the request."}
```

```
<zosconnect_services>
  <service name="cscvincSelectService"
    adminGroup="CSCADMIN" operationsGroup="CSCOPERS"
    invokeGroup="CSCINVKE" readerGroup="CSCREADR"/>
</zosconnect_services>
```

sing a  
o perform

```
c:\z\passticket>curl -X PUT --user user21:QLTNYHS7 --insecure https://wg31.washington.ibm.com:9443/zosConnect/services/cscvincSelectService?status=stopped {"zosConnect": {"serviceName": "cscvincSelectService", "serviceDescription": "", "serviceProvider": "CICS-1.0", "version": "1.0.0", "serviceURL": "https://wg31.washington.ibm.com:9443/zosConnect/services/cscvincSelectService", "serviceInvokeURL": "https://wg31.washington.ibm.com:9443/zosConnect/services/cscvincSelectService?action=invoke", "dataXformProvider": "zosConnectWVXform-1.0", "serviceStatus": "Stopped"}}
```

```
c:\z\passticket>curl -X DELETE --user user21:QLTNYHS7 --insecure https://wg31.washington.ibm.com:9443/zosConnect/services/cscvincSelectService {"name": "cscvincSelectService"}
```

# Tech-Tip: z/OS Connect Authorization Invoke Sample



```
c:\z\passticket>curl -X GET --user user11:MPSIJ0F2 --header "Content-Type: application/json"  
--insecure https://wg31.washington.ibm.com:9443/zosConnect/apis/cscvinc  
{"errorMessage":"BAQR0436W: The zosConnectAuthorization interceptor encountered an error  
while processing a request for API cscvinc.", "errorDetails":"BAQR0409W: User user11 is not  
authorized to perform the request."}
```

```
c:\z\passticket>curl -X GET --user user13:LSHWBUAK --header "Content-Type: application/json"  
--insecure https://wg31.washington.ibm.com:9443/cscvinc/employee/000100  
{"errorMessage":"BAQR0436W: The zosConnectAuthorization interceptor encountered an error  
while processing a request for API cscvinc.", "errorDetails":  
"BAQR0409W: User user13 is not authorized to perform the request."}
```

```
c:\z\passticket>curl -X GET --user user21:C8CFG4KA --header "Content-Type: application/json"  
--insecure https://wg31.washington.ibm.com:9443/cscvinc/employee/000100  
{"cscvincSelectServiceOperationResponse": {"cscvincContainer": {"response": {"CEIBRESP2": 0, "USE  
RID": "CICSUSER", "filea": {"date": "26 11 81", "amount": "$0100.11", "address": "SURREY,  
ENGLAND", "phoneNumber": "32156778", "name": "S. D.  
BORMAN", "employeeNumber": "000100"}, "CEIBRESP": 0}}}}
```

```
c:\z\passticket>curl -X GET --user user31:VWIJW74F --header "Content-Type: application/json"  
--insecure https://wg31.washington.ibm.com:9443/cscvinc/employee/000100  
{"errorMessage":"BAQR0436W: The zosConnectAuthorization interceptor encountered an error  
while processing a request for API cscvinc.", "errorDetails":  
"BAQR0409W: User user33 is not authorized to perform the request."}
```

© 2017, 2021 IBM Corporation

# Tech-Tip: Solution for z/OS Connect Authorization Levels



```
<zosconnect_zosConnectManager
    globalInterceptorsRef="interceptorList_g"
    globalAdminGroup="GMADMIN" globalOperationsGroup="GMOPERS ,CSCOPERS ,DB2OPERS"
    globalInvokeGroup="GMINVOKE" globalReaderGroup="GMREADER"/>

<zosconnect_zosConnectAPIs>
    <zosConnectAPI name="cscvinc" operationsGroup="CSCOPERS" invokeGroup="CSCINV"/>
    <zosConnectAPI name="db2employee" operationsGroup="DB2OPERS" invokeGroup="DB2INVKE"/>
</zosconnect_zosConnectAPIs>

<zosconnect_services>
    <service name="cscvincSelectService" operationsGroup="CSCOPERS" invokeGroup="CSCINV"/>
    <service name="selectEmployee" operationsGroup="DB2OPERS" invokeGroup="DB2INVKE"/>
</zosconnect_services>

<zosconnect_apiRequesters>
    <apiRequester name="cscvincSelectService" operationsGroup="CSCOPERS" invokeGroup="CSCINV"/>
    <apiRequester name="selectEmployee" operationsGroup="DB2OPERS" invokeGroup="DB2INVKE"/>
</zosconnect_apiRequesters>
```

- Now members of groups GMADMIN, GMOPERS, **CSCOPERS**, **DB2OPERS** and GMREADER can connect to a z/OS server from the API toolkit.

When a partial list of subordinate groups are provided, the corresponding default global groups for the absence groups are used.

# Tech-Tip: z/OS Connect Authorization Get Sample



## Deploy a new API Requester

```
c:\z\passticket>curl -X POST --user user12:ID1V0B4C --data-binary @cscvinc.ara --header "Content-Type: application/zip" --insecure https://wg31.washington.ibm.com:9443/zosConnect/apiRequesters {"name":"cscvincapi_1.0.0","version":"1.0.0","description":"","status":"Started","apiRequesterUrl":"https://wg31.washington.ibm.com:9443/zosConnect/apiRequesters/cscvincapi_1.0.0","connection":"cscvincAPI"}
```

This API requester details is now accessible by members of GMREADER but not CSCREADR

```
c:\z\passticket>curl -X GET --user user14:XQIC6GB3 --header "Content-Type: application/json" --insecure https://wg31.washington.ibm.com:9443/zosConnect/apiRequesters/cscvincapi_1.0.0 {"name":"cscvincapi_1.0.0","version":"1.0.0","description":"","status":"Started","apiRequesterUrl":"https://wg31.washington.ibm.com:9443/zosConnect/apiRequesters/cscvincapi_1.0.0","connection":"cscvincAPI"}
```

```
c:\z\passticket>curl -X GET --user user24:UFX1XUKB --header "Content-Type: application/json" --insecure https://wg31.washington.ibm.com:9443/zosConnect/apiRequesters/cscvincapi_1.0.0 {"errorMessage":"BAQR1176W: The zosConnectAuthorization interceptor encountered an error while processing a request for API requester cscvincapi_1.0.0."} *
```

This API requester is now managed by members of CSCOPERS and GMADMIN

```
c:\z\passticket>curl -X PUT --user user12:N2AHSG71 --insecure https://wg31.washington.ibm.com:9443/zosConnect/apiRequesters/cscvincapi_1.0.0?status=stopped {"errorMessage":"BAQR1176W: The zosConnectAuthorization interceptor encountered an error while processing a request for API requester cscvincapi_1.0.0."}
```

```
c:\z\passticket>curl -X PUT --user user22:RVWKLNZV --insecure https://wg31.washington.ibm.com:9443/zosConnect/apiRequesters/cscvincapi_1.0.0?status=stopped {"name":"cscvincapi_1.0.0","version":"1.0.0","description":"","status":"Stopped","apiRequesterUrl":"https://wg31.washington.ibm.com:9443/zosConnect/apiRequesters/cscvincapi_1.0.0","connection":"cscvincAPI"}
```

```
c:\z\passticket>curl -X DELETE --user user22:WAHOKIGF --insecure https://wg31.washington.ibm.com:9443/zosConnect/apiRequesters/cscvincapi_1.0.0 {"name":"cscvincapi_1.0.0"}q
```

\*BAQR0430W: User user24 is not authorized to perform the request on resource /zosConnect/apiRequesters/cscvincapi\_1.0.0

© 2017, 2021 IBM Corporation

mitchj@us.ibm.com

# Tech-Tip: z/OS Toolkit and authorization status



Now members of CSCOPERS and DB2OPERS can connect to a server from the API Toolkit

CSCOPERS

The screenshot shows the 'z/OS Connect EE Servers' interface with the 'APIs' section expanded. Under 'APIs (9)', several services are listed, including 'cscvinc' (Started), 'db2employee' (Not Authorized), 'filemgr' (Started), 'imsPhoneBook' (Started), 'jwtlvpDemoApi' (Started), 'miniloancics' (Started), 'mqapi' (Started), 'phonebook' (Started), and 'restadmin' (Started). The 'db2employee' service is circled in red.

Type to filter by name or status

- ▼ wg31:9443 (wg31.washington.ibm.com:9443)
  - ▼ APIs (9)
    - cscvinc (Started)
    - db2employee (Not Authorized)
    - filemgr (Started)
    - imsPhoneBook (Started)
    - jwtlvpDemoApi (Started)
    - miniloancics (Started)
    - mqapi (Started)
    - phonebook (Started)
    - restadmin (Started)
  - ▼ Services (19)
    - cscvincDeleteService (Started)
    - cscvincInsertService (Started)
    - cscvincSelectService (Started)
    - cscvincService (Started)
    - cscvincUpdateService (Started)
    - deleteEmployee (Started)
    - displayEmployee (Started)
    - inquireCatalog (Started)
    - inquireSingle (Started)
    - insertEmployee (Started)
    - jwtlvpDemoService (Started)
    - miniloanCICSService (Started)
    - miniloanService (Started)
    - mqGetService (Started)
    - mqPutService (Started)
    - placeOrder (Started)
    - selectByDepartments (Started)
    - selectByRole (Started)
    - selectEmployee (Not Authorized)

DB2OPERS

The screenshot shows the 'z/OS Connect EE Servers' interface with the 'APIs' section expanded. Under 'APIs (9)', several services are listed, including 'cscvinc' (Not Authorized), 'db2employee' (Started), 'filemgr' (Started), 'imsPhoneBook' (Started), 'jwtlvpDemoApi' (Started), 'miniloancics' (Started), 'mqapi' (Started), 'phonebook' (Started), and 'restadmin' (Started). The 'cscvinc' and 'db2employee' services are circled in red.

Type to filter by name or status

- ▼ wg31:9443 (wg31.washington.ibm.com:9443)
  - ▼ APIs (9)
    - cscvinc (Not Authorized)
    - db2employee (Started)
    - filemgr (Started)
    - imsPhoneBook (Started)
    - jwtlvpDemoApi (Started)
    - miniloancics (Started)
    - mqapi (Started)
    - phonebook (Started)
    - restadmin (Started)
  - ▼ Services (19)
    - cscvincDeleteService (Started)
    - cscvincInsertService (Started)
    - cscvincSelectService (Not Authorized)
    - cscvincService (Started)
    - cscvincUpdateService (Started)
    - deleteEmployee (Started)
    - displayEmployee (Started)
    - inquireCatalog (Started)
    - inquireSingle (Started)
    - insertEmployee (Started)
    - jwtlvpDemoService (Started)
    - miniloanCICSService (Started)
    - miniloanService (Started)
    - mqGetService (Started)
    - mqPutService (Started)
    - placeOrder (Started)
    - selectByDepartments (Started)
    - selectByRole (Started)
    - selectEmployee (Started)

## **Flowing identities to back-end z/OS systems**

# Basic authentication - Identity and Password



Server XML Configuration elements where basic authentication can be provided.

```
<connectionFactory id="imsTM"> containerAuthDataRef="IMScredentials">
<authData id="IMScredentials" user= "identity" password= "password"/>

<connectionFactory id="imsDB">
<properties.imsudbJLocal databaseName="DFSIVPA" user="identity" password="password"/>
</connectionFactory>

<zosconnect_cicsIpicConnection id="CICS" authDataRef="CICScredentials"/>
<zosconnect_authData id="CICScredentials" user= "identity" password= "password"/>

<zosconnect_zosConnectServiceRestClientConnection id="Db2" basicAuthRef="db2Auth"/>
<zosconnect_zosConnectServiceRestClientBasicAuth id="db2Auth"
    userName="identity" password="password"/>

<jmsQueueConnectionFactory jndiName="MQ">
    <properties.wasJms userName="identity" password="password" />
</jmsQueueConnectionFactory>
```

The value of the password can be encoded in the server XML configuration file. Using the **securityUtility** shipped with WebSphere Liberty Profile.



# Using securityUtility to encrypt passwords

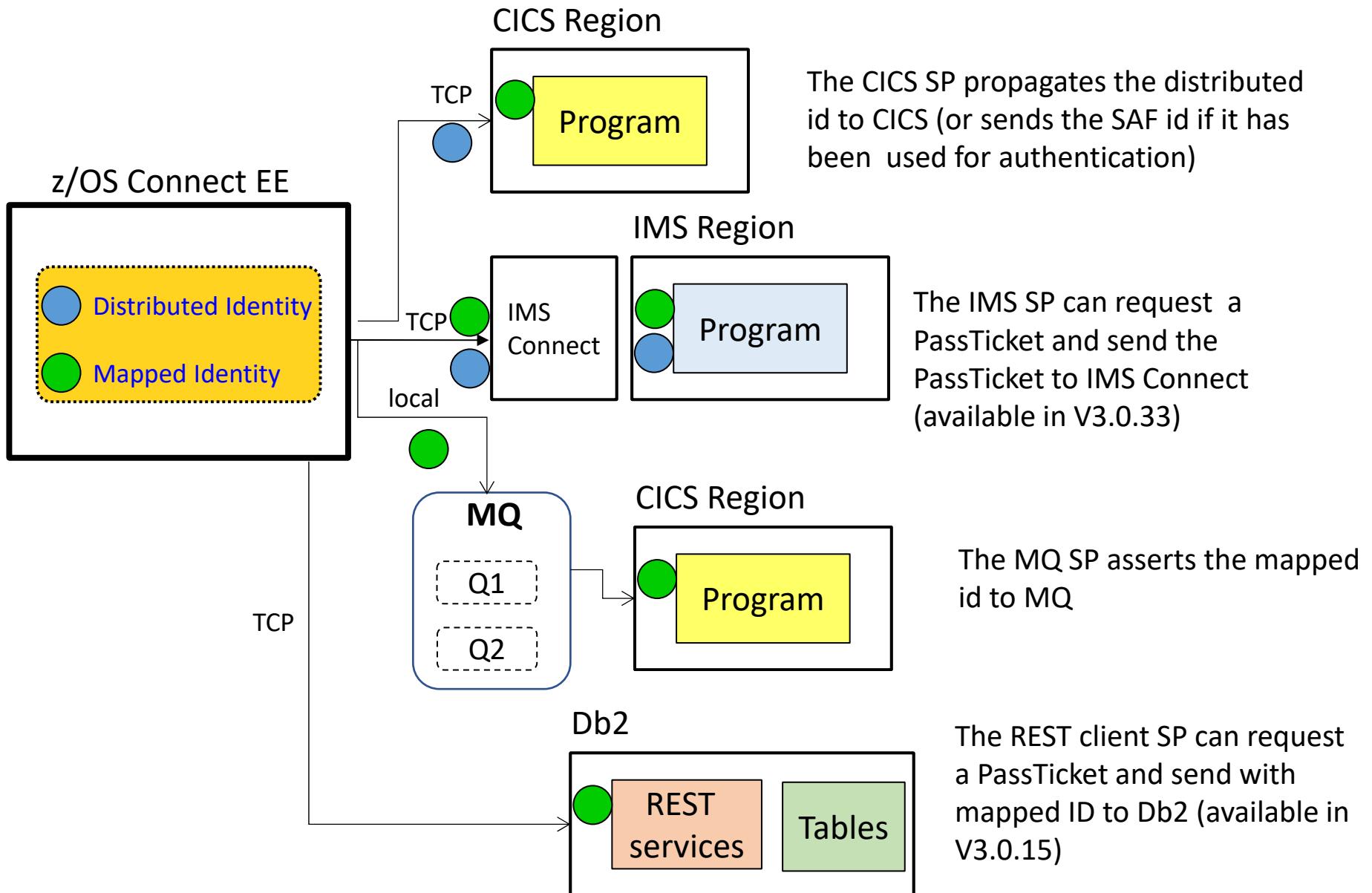
Best practice : use encryption for passwords instead of base64 encoding

- **securityUtility** – located in <wlp\_install\_dir>/wlp/bin
  - Usage: securityUtility {encode|createSSLCertificate|createLTPAKeys|help} [options]
  - For encryption, use encode --key=encryption\_key
    - Specifies the key to be used when encoding using AES encryption. This string is hashed to produce an encryption key that is used to encrypt and decrypt the password. The key can be provided to the server by defining the variable **wlp.password.encryption.key** whose value is the key. If this option is not provided, a default key is used.
- Also supports 1-way hash encoding – for passwords in server.xml with basicRegistry
  - For hash, use encode --encoding=hash
  - ./securityUtility encode --encoding=hash XXXXXXXX
  - {hash}ATAAAAAAIcqTmHn5qZahAAAAAIMjzy+hP8YFaIO6LiCreVe4etRLUS9a25eVuYtx6WKiv

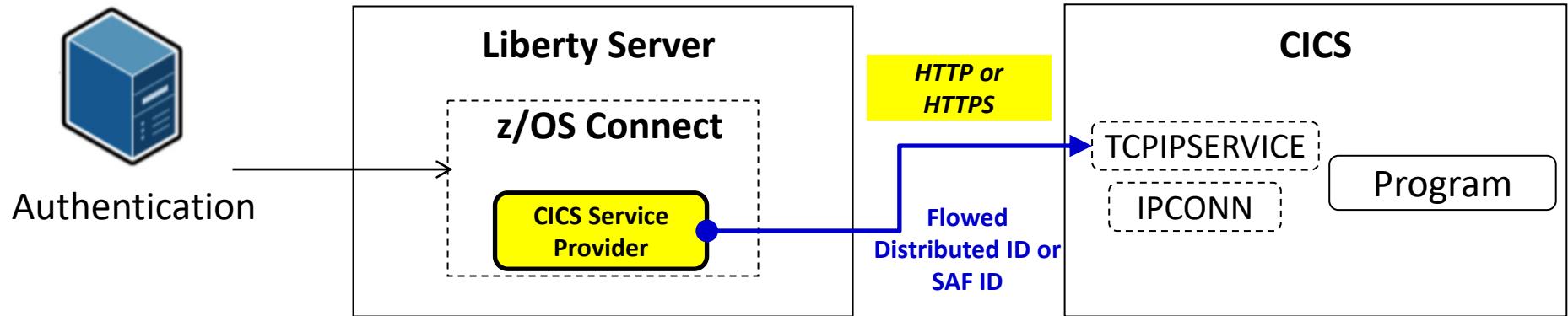
See the WebSphere Application Server for z/OS Liberty at URL:

[https://www.ibm.com/support/knowledgecenter/en/SS7K4U\\_liberty/com.ibm.websphere.wlp.zseries.doc/ae/rwlp\\_command\\_securityutil.html](https://www.ibm.com/support/knowledgecenter/en/SS7K4U_liberty/com.ibm.websphere.wlp.zseries.doc/ae/rwlp_command_securityutil.html)

# Flowing an identity to a back-end subsystem



# Flowing a user ID with CICS service provider



Distributed identities can be propagated to CICS and then mapped to a RACF user ID by CICS. You can then view the distinguished name and realm for a distributed identity in the association data of the CICS task. **Important:** If the z/OS Connect EE server is not in the same Sysplex as the CICS system, you must use an IPIC TLS (JSSE) connection that is configured with client authentication.

If a SAF ID is used for authentication (e.g., basic authentication with a SAF registry) then the SAF ID is passed to CICS.

# CICS IPIC (server.xml)



z/OS Connect EE

The server.xml file is the key configuration file:

The diagram illustrates the configuration process for CICS IPIC. It shows four main components:

- Top Left:** A screenshot of the "inquireSingle Service" window titled "Configuration". It displays "Required Configuration" (Coded character set identifier (CCSID): 37, Connection reference: catalog) and "Optional Configuration" (Transaction ID: [empty], Transaction ID usage: [dropdown]).
- Top Right:** A screenshot of the "Liberty Admin Center" titled "Server Config" for "ipicSSLIDProp.xml". The "Design" tab is selected, showing XML code for a server configuration. Two specific lines are circled in red:

```
1 <server description="CICS IPIC ID propagation connections">
2
3 <!-- Enable features -->
4 <featureManager>
5   <feature>zosconnect:cicsService-1.0</feature>
6 </featureManager>
```

```
8 <zosconnect_cicsIpicConnection id="catalog"
9   host="wg31.washington.ibm.com"
10  port=1493
11  zosConnectNetworkId="CSCVINC"
12  zosConnectApplid="CSCVINC"
13  transid="WT70"
14  transidUsage="EIB_AND_MIRROR"
15  sslCertsRef="cicsSSLSettings"/>
```
- Bottom Left:** Two terminal windows titled "WG31". The left window shows the output of the "IPCONN" command, where the "zosConnectApplid" and "zosConnectNetworkId" fields are highlighted in red. The right window shows the output of the "TCPIPS" command.
- Bottom Right:** A callout box with the text "Define identity propagation to CICS".

# CICS IPCONN



```
DEFINE IPCONN (ZOSCONN)
  GROUP (SYSPGRP)
  APPLID (ZOSCONN)
  NETWORKID (ZOSCONN)
  TCPIPSERVICE (ZOSCONN)
  LINKAUTH (SECUSER)
  USERAUTH (IDENTIFY)
  IDPROP (REQUIRED)
```

Must match `zosConnectApplid` set in  
`zosconnect_cicsIpicConnection`

Must match `zosConnectNetworkid` set in  
`zosconnect_cicsIpicConnection`

Specify the name of  
the TCPIPSERVICE

Requests run under  
the flowed user ID

```
<zosconnect_cicsIpicConnection id="cscvinc"
  host="wg31.washington.ibm.com"
  zosConnectNetworkid="ZOSCONN"
  zosConnectApplid="ZOSCONN"
  port="1491"/>
```

# Identity Propagation and CICS High Availability



The service archive files use the following *Connection reference* values:

- cscvinc
- catalog
- miniloan

baqsvr1's bootstrap.properties

```
ipicPort=1491
cicsHost=dvipa.washington.ibm.com
serverPrefix=baqsvr1
```

The CICS IPCONN resources that correspond to the `zosconnect_cicsIpicConnection` configuration elements must be dedicated to a z/OS Connect server and can not be reused.

baqsvr2's bootstrap.properties

```
cicsHost=dvipa.washington.ibm.com
ipicPort=1491
serverPrefix=baqsvr2
```

ipicIDProp.xml

```
<zosconnect_cicsIpicConnection id="cscvinc"
  host="${cicsHost}"
  zosConnectNetworkid="${wlp.server.name}"
  zosConnectApplid="${wlp.server.name}"
  sharedPort="true" port="${ipicPort}"/>
→ baqsvr1 or baqsvr2

<zosconnect_cicsIpicConnection id="catalog"
  host="${cicsHost}"
  zosConnectNetworkid="${serverPrefix}C"
  zosConnectApplid="${serverPrefix}C"
  sharedPort="true" port="${ipicPort}"/>
→ baqsvr1C or baqsvr2C

<zosconnect_cicsIpicConnection id="miniloan"
  host="${cicsHost}"
  zosConnectNetworkid="${serverPrefix}M"
  zosConnectApplid="${serverPrefix}M"
  sharedPort="true" port="${ipicPort}"/>
→ baqsvr1M or baqsvr2M
```

# CICS IPCONN and TCPIPSERVICE resources for HA



## CICS Specific TCPIPSERVICE - IPIC

```
TCpipservice : IPIC1
GROup       : SYSPGRP
Urm         ==> DFHISAIP
POrtnumber  ==> 01492
STatus      ==> Open
PROtocol    ==> IPic
TRansaction ==> CISS
Host        ==> ANY
Ipaddress   ==> ANY
SPeciftcp  ==>
```

## CICS Generic TCPIPSERVICE - IPICG

```
TCpipservice : IPICG1
GROup       : SYSPGRP
Urm         ==> DFHISAIP
POrtnumber  ==> 01491
STatus      ==> Open
PROtocol    ==> IPic
TRansaction ==> CISS
Host        ==> ANY
Ipaddress   ==> ANY
SPeciftcp  ==> IPIC
```

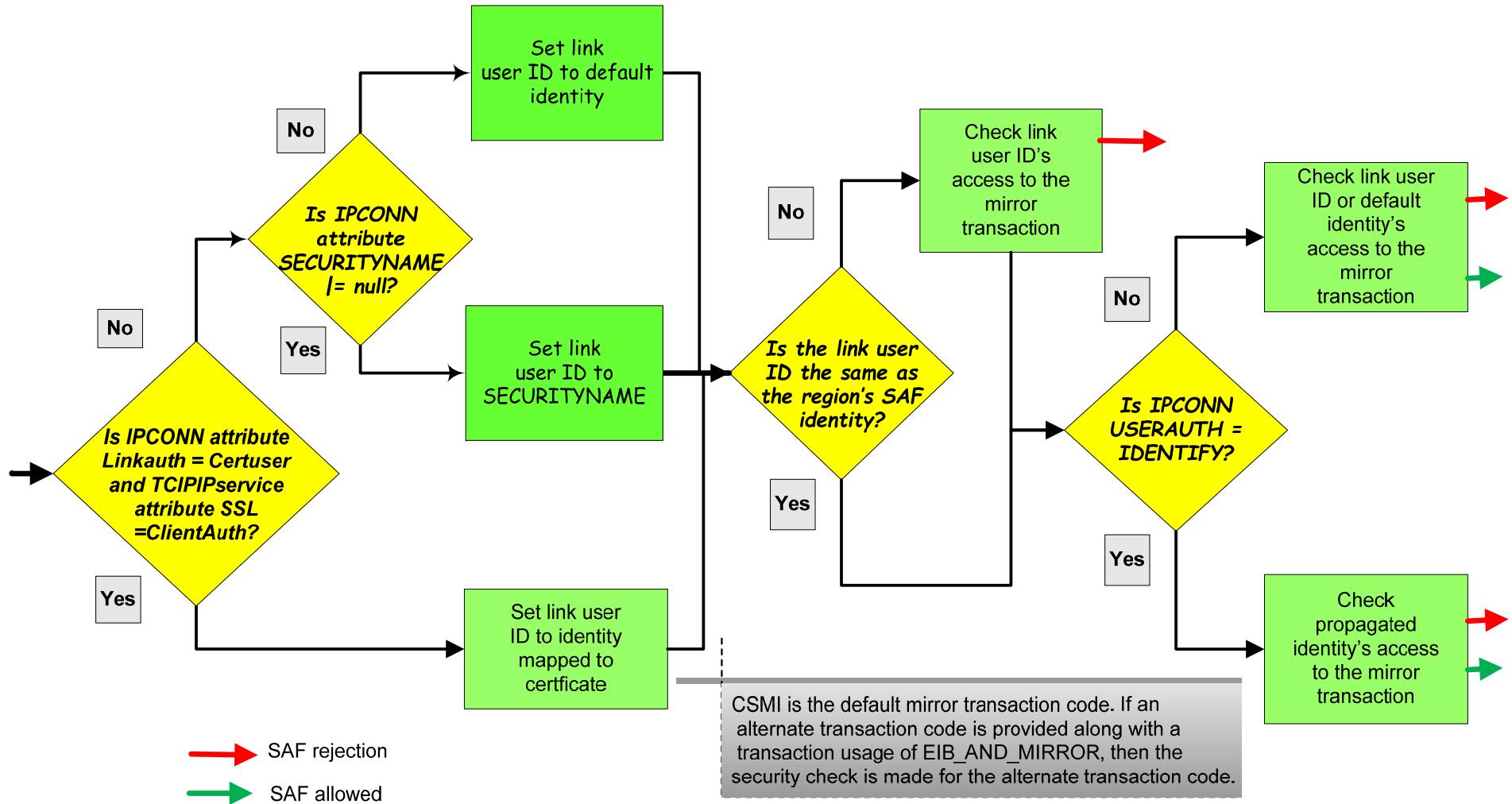
A client connects first to the CICS region's generic port (1491) and then the CICS region redirects the client to the region's specific port (1492).

## I IPCONN ACQ

```
STATUS: RESULTS - OVERTYPE TO MODIFY
Ipc(BAQSVR1 ) App(BAQSVR1) Net(BAQSVR1) Ins Acq Nos
          Rece(001) Sen(000) Tcp(IPIC)
Ipc(BAQSVR1C) App(BAQSVR1C) Net(BAQSVR1C) Ins Acq Nos
          Rece(001) Sen(000) Tcp(IPIC)
Ipc(BAQSVR1M) App(BAQSVR1M) Net(BAQSVR1M) Ins Acq Nos
          Rece(001) Sen(000) Tcp(IPIC)
Ipc(BAQSVR2 ) App(BAQSVR2) Net(BAQSVR2) Ins Acq Nos
          Rece(001) Sen(000) Tcp(IPIC)
Ipc(BAQSVR2C) App(BAQSVR2C) Net(BAQSVR2C) Ins Acq Nos
          Rece(001) Sen(000) Tcp(IPIC)
Ipc(BAQSVR2M) App(BAQSVR2M) Net(BAQSVR2M) Ins Acq Nos
          Rece(001) Sen(000) Tcp(IPIC)
```

<sup>1</sup>CICS requires the specific TCPIPSERVICE be installed before the corresponding generic TCPIPSERVICE resource. TCPIPServices are installed in alphabetically order, so the name of specific service must be alphabetically prior to the name of the generic TCPIPSERVICE.

# Tech/Tip: CICS IPIC Security



# PassTickets and IMS

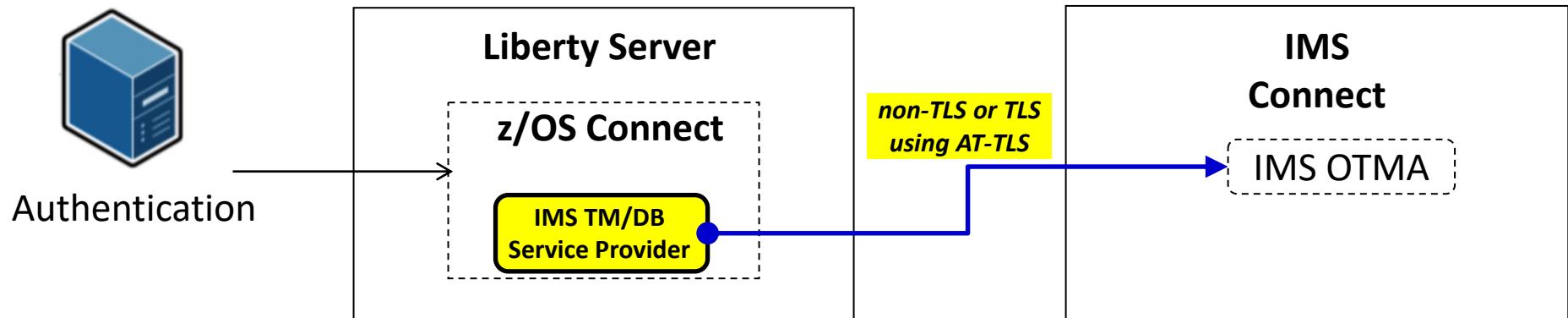
- ❑ Basic authentication to IMS Connect using a PassTicket depends on the APPL parameters configured in IMS Connect.

```
HWS= (ID=IMS15HWS, XIBAREA=100, RACF=Y, RRS=Y)
TCPIP= (HOSTNAME=TCPIP, PORTID= (4000, LOCAL) , RACFID=JOHNSON, TIMEOUT=5000)
DATASTORE= (GROUP=OTMAGRP, ID=IVP1, MEMBER=HWSMEM, DRU=HWSYDRU0,
TMEMBER=OTMAMEM, APPL=IMSTMPL)
ODACCESS= (ODBMAUTOCONN=Y, IMSPLEX= (MEMBER=IMS15HWS, TMEMBER=PLEX1) ,
DRDAPORT= (ID=5555, PORTTMOT=6000) , ODBMTMOT=6000, APPL=IMSDBAPL)
```

```
RDEFINE PTKTDATA IMSTMPL SSIGNON(0123456789ABCDEF) APPLDATA('NO REPLAY PROTECTION') UACC(NONE)
RDEFINE PTKTDATA IRRPTAUTH.IMSTMPL.* UACC(NONE)
PERMIT IRRPTAUTH.IMSTMPL ID(LIBSERV) CLASS(PTKTDATA) ACCESS(UPDATE)
```

```
RDEFINE PTKTDATA IMSDBAPL SSIGNON(0123456789ABCDEF) APPLDATA('NO REPLAY PROTECTION') UACC(NONE)
RDEFINE PTKTDATA IRRPTAUTH.IMSDBAPL.* UACC(NONE)
PERMIT IRRPTAUTH.IMSDBAPL ID(LIBSERV) CLASS(PTKTDATA) ACCESS(UPDATE)
```

# Flowing an identity to IMS Connect (TM)



```
HWS=(ID=IMS15HWS,XIBAREA=100,RACF=Y,RRS=Y)  
TCPIP=(HOSTNAME=TCPIP,PORTID=(4000,LOCAL),RACFID=JOHNSON,TIMEOUT=5000)  
DATASTORE=(GROUP=OTMAGRP,ID=IVP1, MEMBER=HWSMEM, DRU=HWSYDRU0,  
TMEMBER=OTMAMEM, APPL=IMSTMPL)
```

**Authentication options:**

1. User ID / password
2. PassTicket support

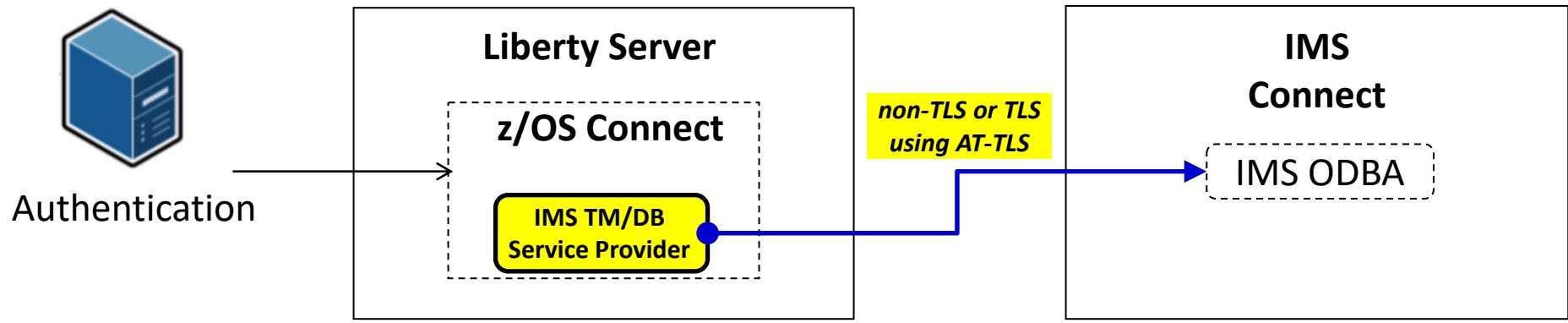
```
<connectionFactory containerAuthDataRef="Connection1_Auth" id="IVP1">  
<properties.gmoa hostName="wg31.washington.ibm.com" portNumber="4000"/>  
</connectionFactory>  
<authData id="Connection1_Auth" user="USER1" password="{xor}GhIPExAGDwg="/>
```

Specify a user identity and password to be used in the request to IMS Connect

```
<connectionFactory containerAuthDataRef="Connection1_Auth" id="IVP1">  
<properties.gmoa hostName="wg31.washington.ibm.com" portNumber="4000  
applicationName="IMSTMPL"/>  
</connectionFactory>
```

Request a PassTicket And use it in the request to IMS Connect

# Flowing an identity to IMS Connect (DB)



```
HWS=(ID=IMS15HWS,XIBAREA=100,RACF=Y,RRS=Y)  
TCPIP=(HOSTNAME=TCPIP,PORTID=(4000,LOCAL),RACFID=JOHNSON,TIMEOUT=5000)  
ODACCESS=(ODBMAUTOCONN=Y,IMSPLEX=(MEMBER=IMS15HWS,TMEMBER=PLEX1),  
DRDAPORT=(ID=5555,PORTTMOT=6000),ODBMTMOT=6000,APPL=IMSDBAPL)
```

**Authentication options:**

1. User ID / password
2. PassTicket support

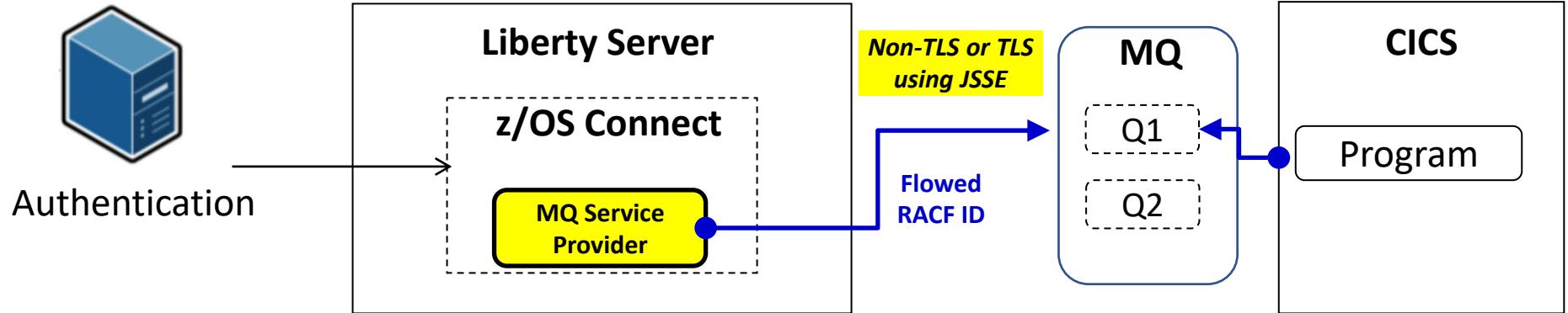
```
<connectionFactory id="DFSIVPACConn"> <properties.imsudbJLocal  
databaseName="DFSIVPA" datastoreName="IVP1" portNumber="5555"  
driverType="4" datastoreServer="wg31.washington.ibm.com" flattenTables="True"  
user="USER1" password="USER1" />  
</connectionFactory>
```

Specify a user identity and password to be used in the request to IMS Connect

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

Request a PassTicket  
And use it in the request to IMS Connect

# Flowing a user ID with MQ service provider



Set `useCallerPrincipal=true` to flow the authenticated RACF user ID

```
<zosconnect_services>
  <service name="mqPut">
    <property name="destination" value="jms/default"/>
    <property name="useCallerPrincipal" value="true"/>
  </service>
</zosconnect_services>
```

Define identity propagation to MQ

# PassTickets and Db2

- ❑ Basic authentication Db2 using a PassTicket depends on the Db2 configuration.

```
DSNL080I -DSN2 DSNLTDDF DISPLAY DDF REPORT FOLLOWS:  
DSNL081I STATUS=STARTD  
DSNL082I LOCATION          LUNAME          GENERICCLU  
DSNL083I DSN2LOC           USIBMWZ.DSN2APPL  USIBMWZ.DSNOAPPL  
DSNL084I TCPPORT=2446    SECPORT=2445    RESPORT=2447    IPNAME--NONE  
DSNL085I IPADDR=:192.168.17.201  
DSNL086I SQL    DOMAIN=WG31.WASHINGTON.IBM.COM  
DSNL105I CURRENT DDF OPTIONS ARE:  
DSNL106I PKGREL = COMMIT  
DSNL106I SESSIDLE = 001440  
DSNL099I DSNLTDDF DISPLAY DDF REPORT COMPLETE
```

```
DSNL080I -DSNC DSNLTDDF DISPLAY DDF REPORT FOLLOWS:  
DSNL081I STATUS=STARTD  
DSNL082I LOCATION          LUNAME          GENERICCLU  
DSNL083I DSN2LOC           -NONE          -NONE  
DSNL084I TCPPORT=2446    SECPORT=2445    RESPORT=2447    IPNAME=DB2IPNM  
DSNL085I IPADDR=:192.168.17.252  
DSNL086I SQL    DOMAIN=WG31.WASHINGTON.IBM.COM  
DSNL086I RESYNC DOMAIN=WG31.WASHINGTON.IBM.COM  
DSNL089I MEMBER IPADDR=:192.168.17.252  
DSNL105I CURRENT DDF OPTIONS ARE:  
DSNL106I PKGREL = COMMIT  
DSNL106I SESSIDLE = 001440  
DSNL099I DSNLTDDF DISPLAY DDF REPORT COMPLETE
```

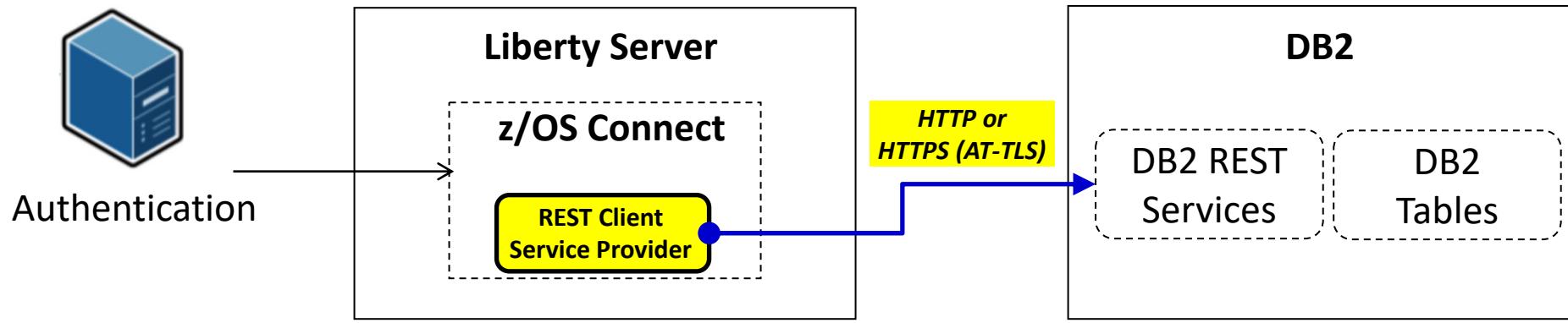
Which value should be used for *appName* is determined for use in RACF resources is determined as shown below.

- If **GENERICLU** is defined, use the second part of **GENERICLU** for *appName*, e.g., **DSN0APPL**
- If **GENERICLU** is not defined, use the second part of **LUNAME** for *appName*, e.g., **DSN2APPL**
- If neither **GENERICLU** or **LUNAME** is defined, use the value of the **IPNAME** for *appName*, e.g., **DB2IPNM**

```
RDEFINE PTKTDATA DSN2APPL SSIGNON(0123456789ABCDEF) APPLDATA('NO REPLAY PROTECTION') UACC(NONE)  
RDEFINE PTKTDATA IRRPTAUTH.DSN2APPL.* UACC(NONE)  
PERMIT IRRPTAUTH.DSN2APPL ID(LIBSERV) CLASS(PTKTDATA) ACCESS(UPDATE)
```

© 2017, 2021 IBM Corporation

# Flowing the identity for the REST client SP (Db2)



```
<zosconnect_zosConnectServiceRestClientConnection id="Db2Conn"  
host="wg31.washington.ibm.com"  
port="2446"  
basicAuthRef="dsn2Auth" />  
<zosconnect_zosConnectServiceRestClientBasicAuth id="dsn2Auth"  
userName="USER1"  
password="USER1"/>
```

## Authentication options:

1. User ID / password
2. TLS Client Certificate (JSSE)
3. PassTicket support

Specify a user identity and password to be used in the HTTP header with the Db2 REST Service

```
<zosconnect_zosConnectServiceRestClientConnection id="Db2Conn"  
host="wg31.washington.ibm.com"  
port="2446"  
basicAuthRef="dsn2Auth" />  
  
<zosconnect_zosConnectServiceRestClientBasicAuth id="dsn2Auth"  
appName="DSN2APPL"/>
```

z/OS Connect requests a PassTicket from RACF

# Tech/Tip: Db2 REST Security

- ❑ Access to Db2 REST services requires READ access to the Db2 subsystem DSNR REST resource. i.e., permit READ access to this resource to the identity in question, for example

```
PERMIT DSN2.REST CLASS(DSNR) ID(USER2) ACC(READ) where DSN2 is the Db2 subsystem ID  
SETROPTS RACLIST(DSNR) REFRESH
```

- ❑ Db2 package access is also required. If a user is not able to display a valid Db2 REST services in the z/OS Connect Db2 services development tooling or by using a **POST** to the Db2 provided REST interface URL of <http://wg31.washington.ibm.com:2446/services/DB2ServiceDiscover>, then they may not have sufficient access to the package containing the service.

For example, if service *zCEEService.selectEmployee* is defined to Db2 but not visible in the z/OS Connect tooling or if a **GET** request to URL <http://wg31.washington.ibm.com:2446/services/zCEEService/selectEmployee> fails with message:

```
{  
  "StatusCode": 500,  
  "StatusDescription": "Service zCEEService.selectEmployee discovery failed due to  
  SQLCODE=-551 SQLSTATE=42501, USER2 DOES NOT HAVE THE PRIVILEGE TO PERFORM OPERATION EXECUTE  
  PACKAGE ON OBJECT zCEEService.selectEmployee. Error Location:DSNLJACC:35"  
}
```

The user needs to be granted execute authority on package *zCEEService.selectEmployee* with command:

```
GRANT EXECUTE ON PACKAGE "zCEEService"."selectEmployee" TO USER2  or  
GRANT EXECUTE ON PACKAGE "zCEEService".""*"" TO USER2
```

# WOLA Security



## ❑ MVS Batch

```
<zosLocalAdapters wolaGroup="ZCEESRVR"
    wolaName2="ZCEESRVR"
    wolaName3="ZCEESRVR"/>
```

```
RDEFINE CBIND BBG.WOLA.ZCEESRVR.ZCEESRVR.ZCEESRVR UACC(NONE) OWNER(SYS1)
PERMIT BBG.WOLA.ZCEESRVR.ZCEESRVR.ZCEESRVR CLASS(CBIND) ACCESS(READ) ID(USER1,START1)
SETROPTS RACLIST(CBIND) REFRESH
```

## ❑ Data Virtualization Manager

```
"DEFINE ZCPATH",
"  NAME(ZCEE)          '',
"  RNAME(ZCEEDVM)      '',
"  WNAME(ZCEEDVM)      '',
'''
```

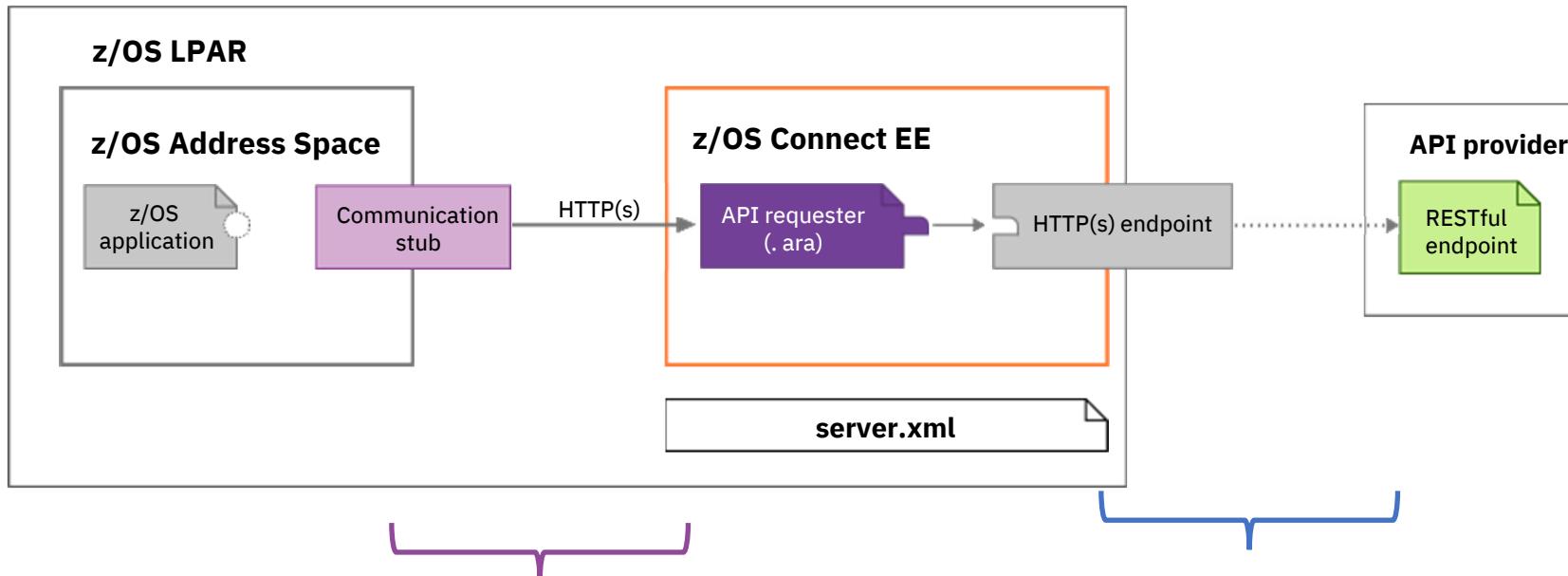
```
<!-- Adapter Details with WOLA Group Name (ZCEEDVM) -->
<zosLocalAdapters wolaName3="NAME3"
    wolaName2="NAME2"
    wolaGroup="ZCEEDVM"/>
```

```
RDEFINE CBIND BBG.WOLA.ZCEEDVM.** UACC(NONE)
PERMIT BBG.WOLA.ZCEEDVM.** CLASS(CBIND) ID(LIBSERV) ACC(READ)
SETROPTS RACLIST(CBIND) REFRESH
```

# **z/OS Connect API Requester Security**

## **Details**

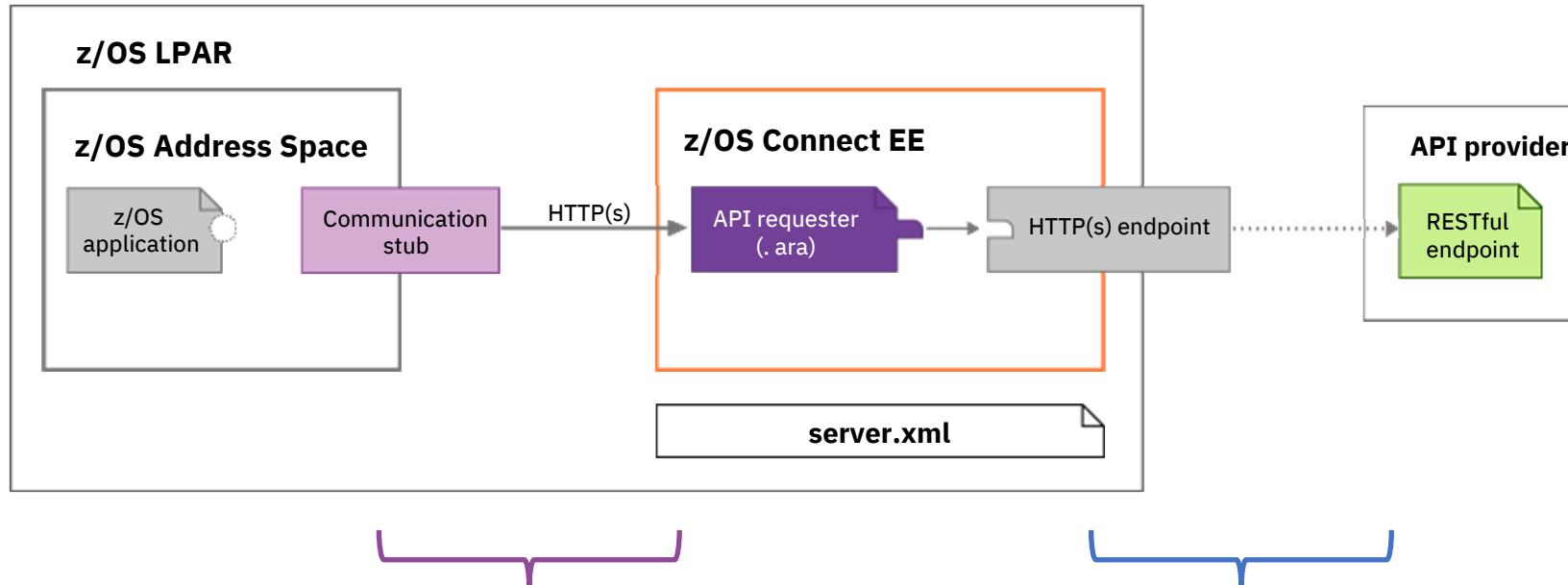
# Authentication



Options:

- 1. Basic Authentication
- 2. TLS Client/Server
- 1. Basic Authentication
- 2. TLS Client/Server
- 3. Third Party token

# Encryption

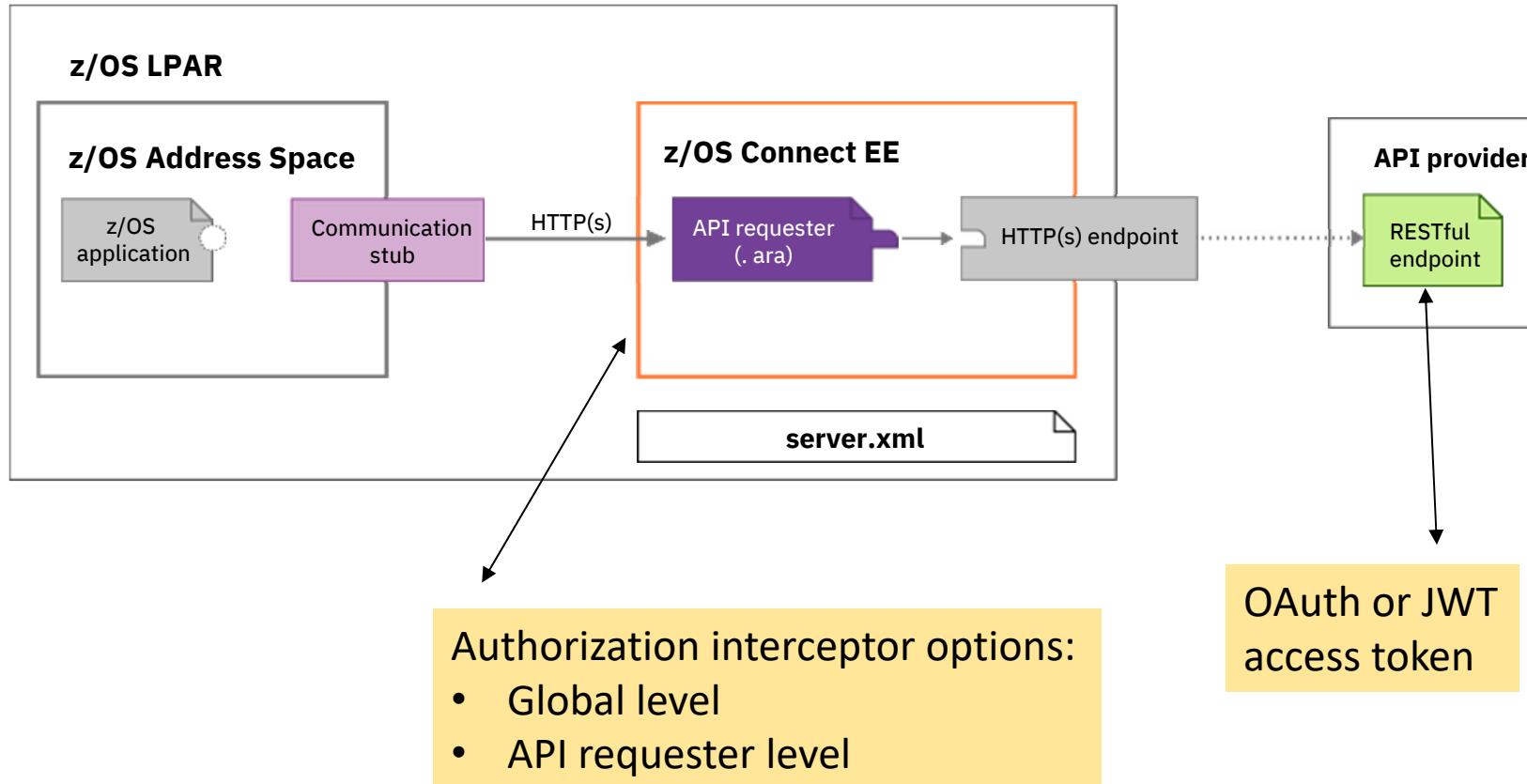


Options:

1. AT-TLS
2. CICS TLS (System TLS)

1. JSSE
2. AT-TLS

# Authorization



# API Requester- API Provider Authentication



Several different ways this can be accomplished:

## Basic Authentication



ID/PW

Okay!

zCEE  
Server

API  
Provider  
Client

**zCEE server supplies  
ID/PW or ID/PassTicket**

## Client Certificate



TLS  
Client  
Cert

zCEE  
Server

API  
Provider  
Client

**Server requests a  
client certificate**

**zCEE supplies a client  
certificate**

z/OS Connect

Okay!

Identity Mapping  
Green circle = 'FRED'

## Third Party Authentication



Cert

Token (JWT)

Auth

ID/PW

Okay

3<sup>rd</sup> Party

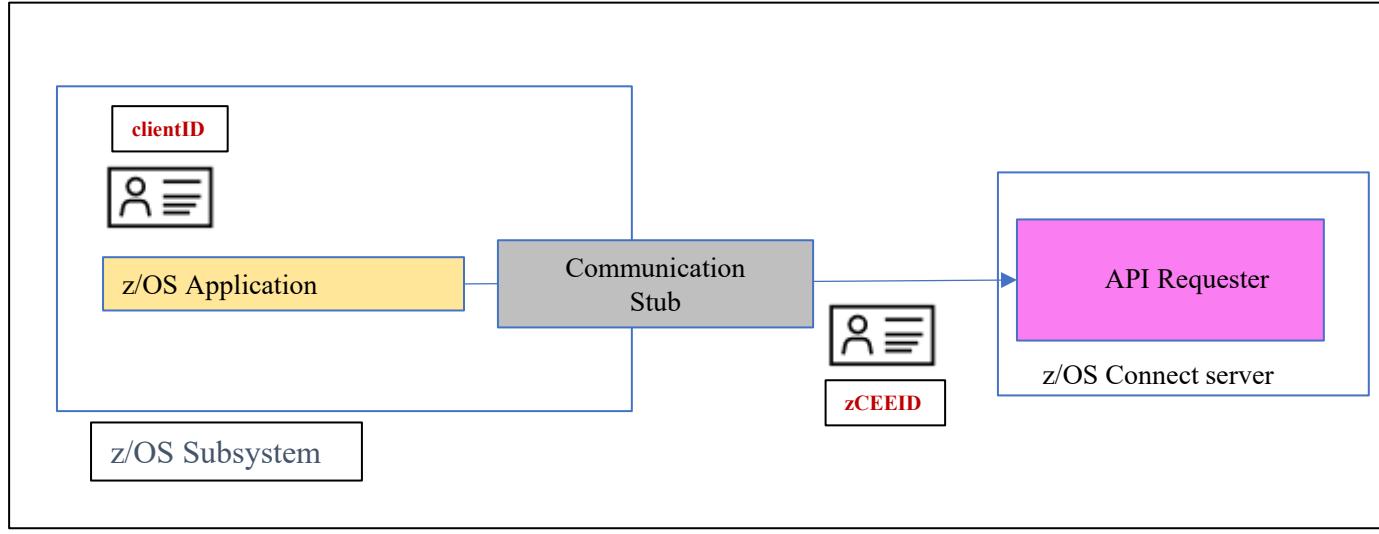
API  
Provider  
Client

**zCEE Server authenticates to 3<sup>rd</sup> party  
server**

**zCEE Server receives a trusted 3<sup>rd</sup> party  
token**

**Token flows to API Provider**

# API Requester - basic authentication and identity assertion



***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	OFF	Identity assertion is disabled. The zCEE server authenticates <b><i>zCEEID</i></b> and checks whether <b><i>zCEEID</i></b> has the authority to invoke an API requester.
	ASSERT_SURROGATE	Identity assertion is enabled. The zCEE server authenticates <b><i>zCEEID</i></b> and checks whether <b><i>zCEEID</i></b> is a surrogate of <b><i>clientID</i></b> . If <b><i>zCEEID</i></b> is a surrogate of <b><i>clientID</i></b> , the server further checks whether <b><i>clientID</i></b> has the authority to invoke an API requester; otherwise, a BAQR7114E message occurs.
	ASSERT_ONLY	Identity assertion is enabled. The zCEE server authenticates <b><i>zCEEID</i></b> and directly checks whether <b><i>clientID</i></b> 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 <b><i>clientID</i></b> 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 <b><i>clientID</i></b> has the authority to invoke an API requester

```
<zosconnect_apiRequesters idAssertion="ASSERT_ONLY">
</zosconnect_apiRequesters>
```

# SAF SURROGAT Resources for identity assertion



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

*su*

*cd /usr/lib/java\_runtime*

*extattr +p libifaedjreg64.so*

- *Permit the server 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***

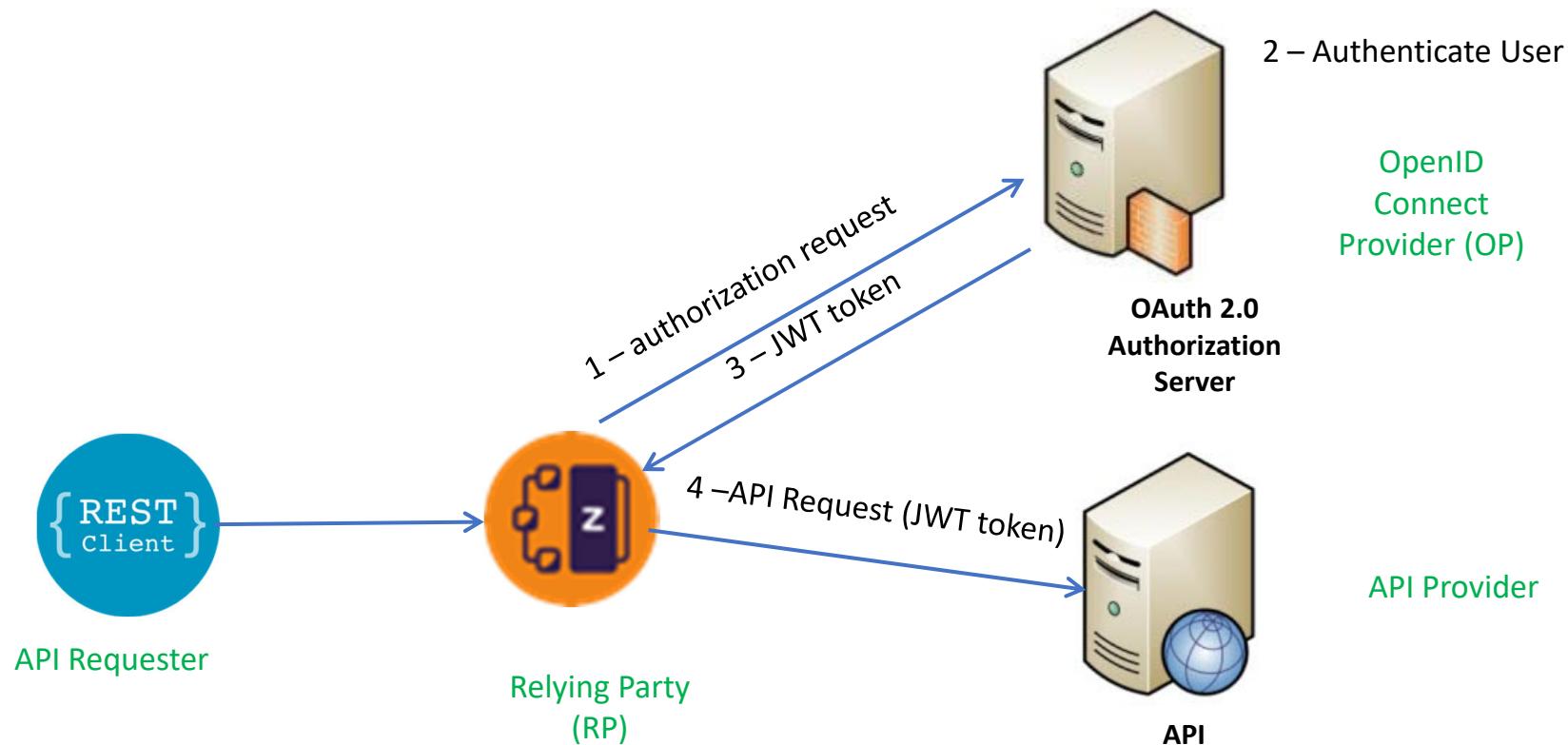
**RDEFINE SURROGAT \*.BAQASSRT UACC(NONE) OWNER(SYS1)  
PERMIT \*.BAQASSRT CLASS(SURROGAT) ACCESS(READ) ID(*zCEEID*)  
SETROPTS RACLIST(SURROGAT) REFRESH**

# **z/OS Connect API Requester - Token Support**

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

1. Use the OAuth 2.0 support when the request is part of an OAuth 2.0 flow. With OAAUTH configured, the token can be an opaque token or a JWT token.
2. 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 Flow for API requester

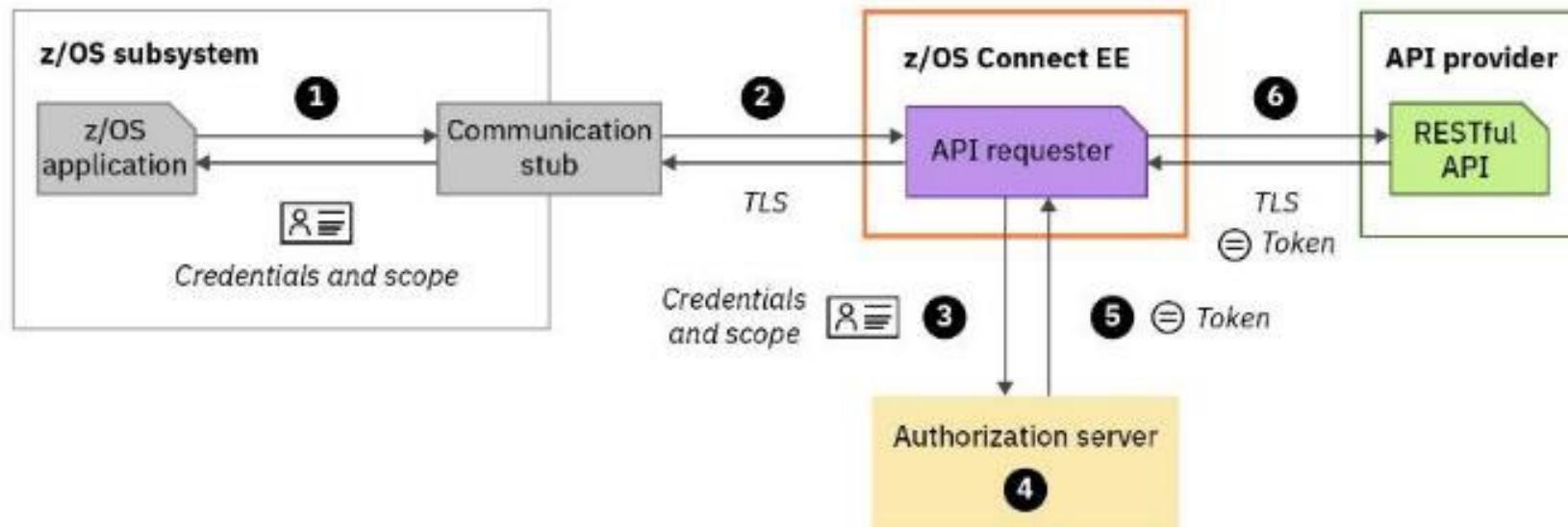


## Grant Types:

- client\_credentials
- password



# Calling an API with OAuth 2.0 support



# **z/OS Connect OpenID Connect terms**

**Resource owner** - An entity capable of granting access to a protected resource. When the resource owner is a person, it is referred to as an end user. *In a z/OS Connect EE API requester scenario, the resource owner might be the **user** of the CICS, IMS, or z/OS application.*

**Resource server** - The server that hosts the protected resources and accepts and responds to protected resource requests by using access tokens. *In a z/OS Connect EE API requester scenario, the resource server is the **request endpoint** for the remote RESTful API.*

**Client** - An application that makes protected resource requests on behalf of the resource owner and with its authorization. *In a z/OS Connect EE API requester scenario, the client is a 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.*

**Authorization server** - The server that issues access tokens to the client after authenticating the resource owner and obtaining authorization. *In a z/OS Connect EE API requester scenario, the authorization server is called by the z/OS Connect EE server to retrieve an access token.*

# 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"/>
```

# Configuring OAuth support – BAQRINFO copy book



```
05 BAQ-OAUTH.  
 07 BAQ-OAUTH-USERNAME          PIC X(256).  
 07 BAQ-OAUTH-USERNAME-LEN      PIC S9(9) COMP-5 SYNC VALUE 0.  
 07 BAQ-OAUTH-PASSWORD          PIC X(256).  
 07 BAQ-OAUTH-PASSWORD-LEN      PIC S9(9) COMP-5 SYNC VALUE 0.  
 07 BAQ-OAUTH-CLIENTID          PIC X(256).  
 07 BAQ-OAUTH-CLIENTID-LEN      PIC S9(9) COMP-5 SYNC VALUE 0.  
 07 BAQ-OAUTH-CLIENT-SECRET     PIC X(256).  
 07 BAQ-OAUTH-CLIENT-SECRET-LEN PIC S9(9) COMP-5 SYNC VALUE 0.  
 07 BAQ-OAUTH-SCOPE-PTR         USAGE POINTER.  
 07 BAQ-OAUTH-SCOPE-LEN         PIC S9(9) COMP-5 SYNC.
```

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

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

Scope is always required.

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

# 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>
```

<sup>1</sup>See URL [https://www.ibm.com/support/knowledgecenter/SS7K4U\\_liberty/com.ibm.websphere.wlp.zseries.doc/ae/twlp\\_oidc\\_token\\_endpoint.html](https://www.ibm.com/support/knowledgecenter/SS7K4U_liberty/com.ibm.websphere.wlp.zseries.doc/ae/twlp_oidc_token_endpoint.html)

<sup>2</sup> These credentials can be specified by the 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 "ATSOAUTHPASSWORD" to envVariableName.  
PERFORM CALL-CEEENV THRU CALL-CEEENV-END  
MOVE VAR(1:valueLength) to BAQ-OAUTH-PASSWORD  
MOVE valueLength to BAQ-OAUTH-PASSWORD-LEN  
MOVE " " to BAQ-OAUTH-CLIENTID.  
MOVE 0 to BAQ-OAUTH-CLIENTID-LEN.  
MOVE " " 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.
```

## Execution JCL

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

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



# 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(EJBRROLE )  
INSUFFICIENT ACCESS AUTHORITY  
ACCESS INTENT(READ ) ACCESS ALLOWED(NONE )
```

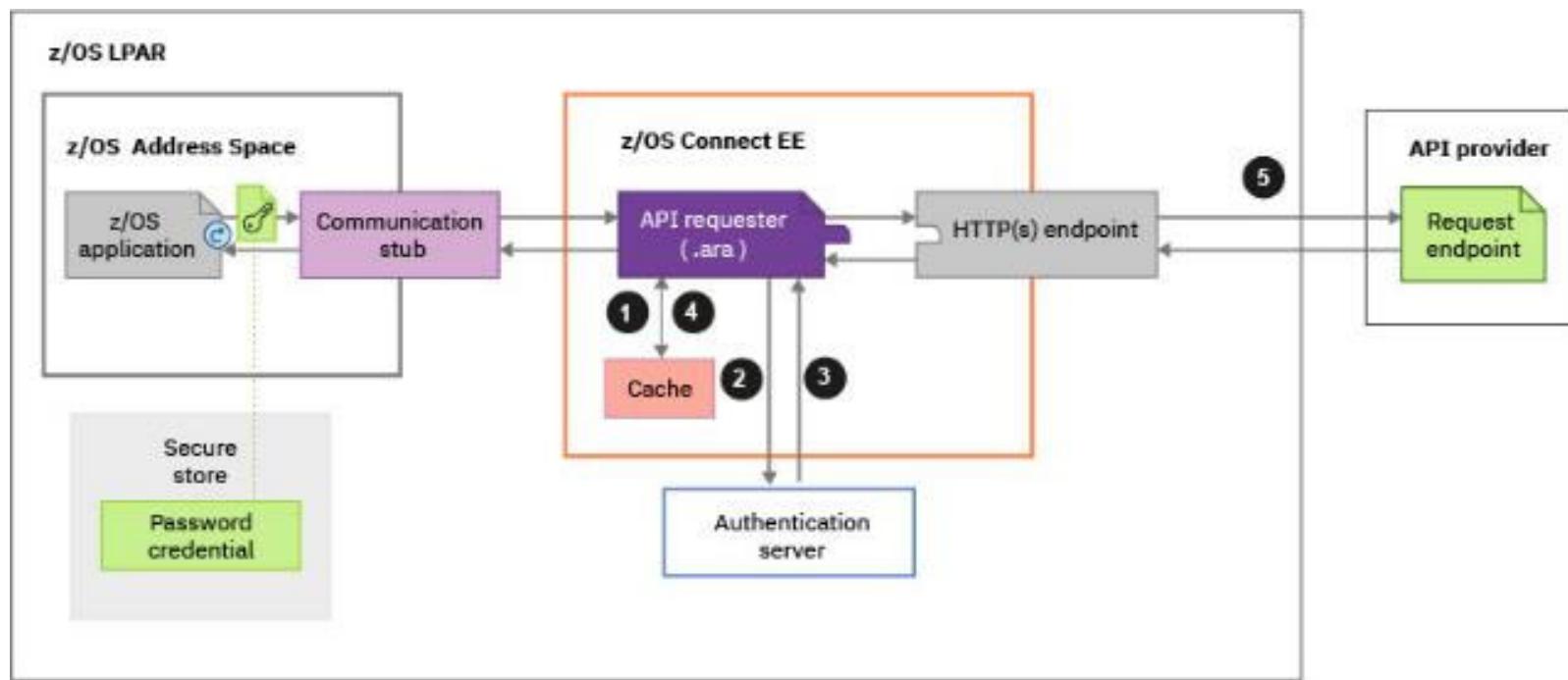
)  
*distuser2* is mapped to RACF identity USER2 which has no access to the EJBRROLE protecting z/OS Connect



# Calling an API with using a JWT custom flow

z/OS Connect provides two ways of calling an API secured with a JWT

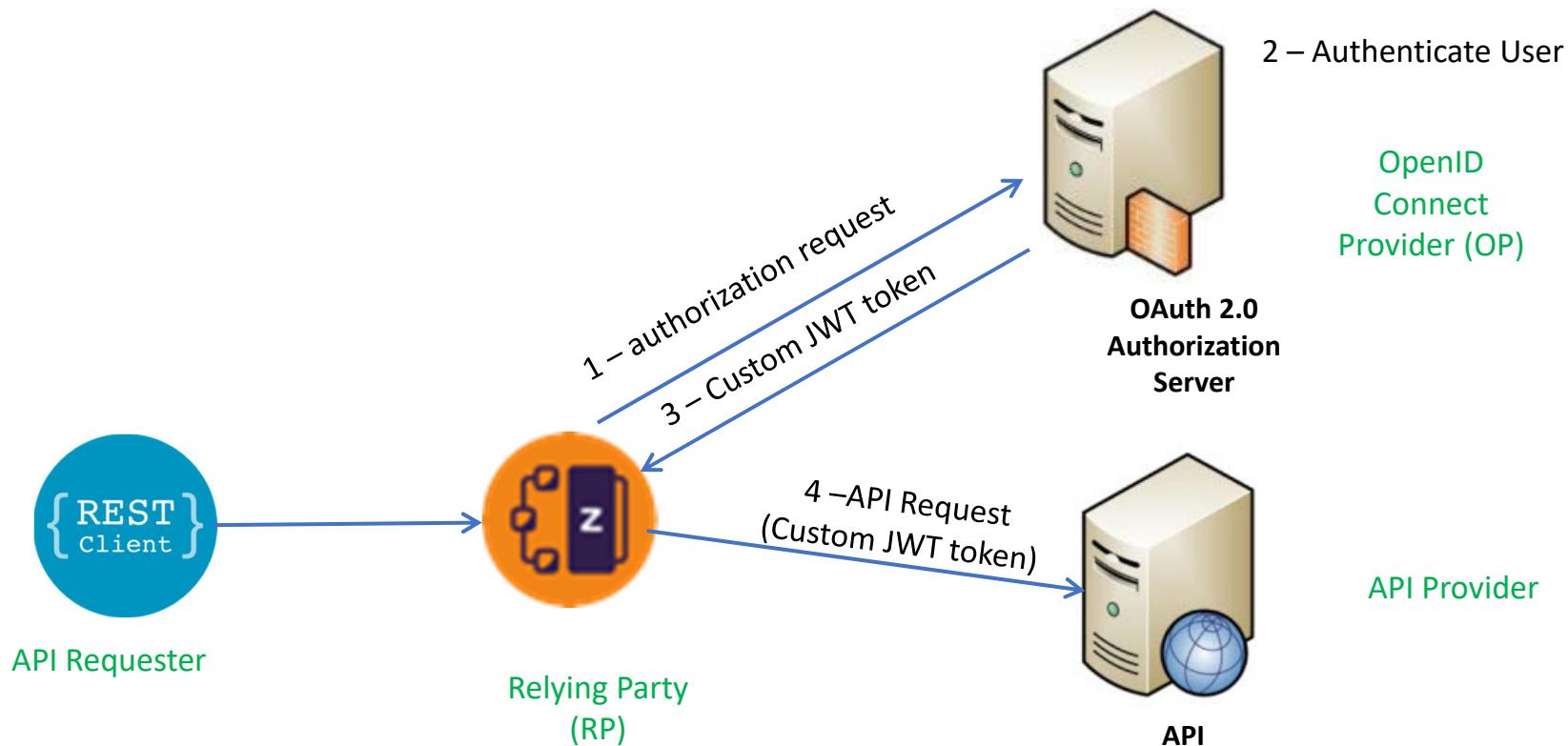
- ❑ Use the OAuth 2.0 support when the request is part of an OAuth 2.0 flow as just described.
- ❑ In a non-OAuth 2.0 scenario, a JWT token is used in a custom flow, for example:
  - When you need to specify the HTTP verb that is used in the request to the authentication server.
  - When you need to 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.



© 2017, 2021 IBM Corporation

mitchj@us.ibm.com

# z/OS Connect OAuth Customer Flow





# API Requester – JWT Custom flow

## BAQRINFO copy book

```
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.
```

## 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_authConfig 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/token"1
    basicAuthRef="tokenCredential" 2
    sslCertsRef="OutboundSSLSettings" />

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

<sup>1</sup>See URL [https://www.ibm.com/support/knowledgecenter/SS7K4U/liberty/com.ibm.websphere.wlp.zseries.doc/ae/twlp\\_oidc\\_token\\_endpoint.html](https://www.ibm.com/support/knowledgecenter/SS7K4U/liberty/com.ibm.websphere.wlp.zseries.doc/ae/twlp_oidc_token_endpoint.html)

<sup>2</sup> These credentials can be specified by the application



# Configuring Custom JWT flow

## Request Token Example 1

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

## Response Token

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

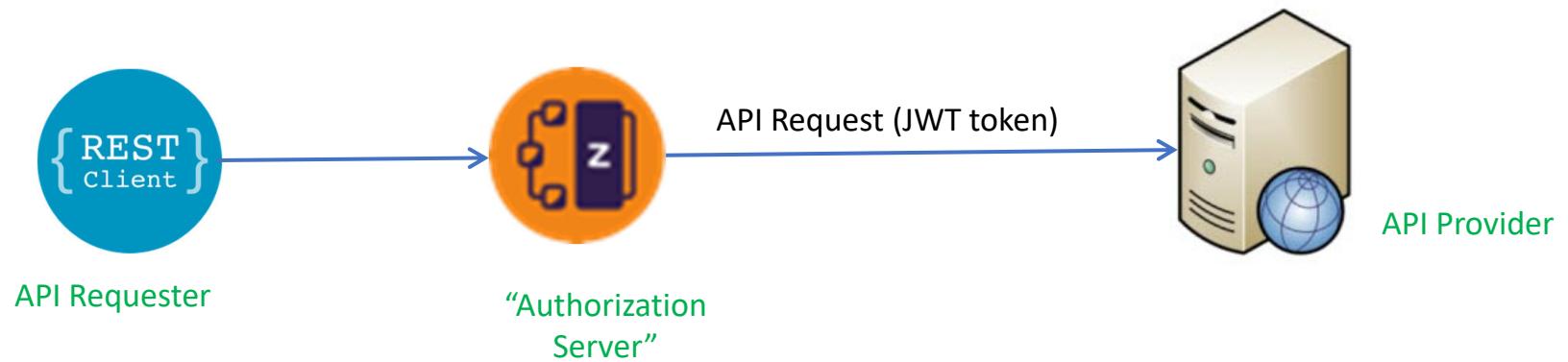
## Response Token Example 2

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

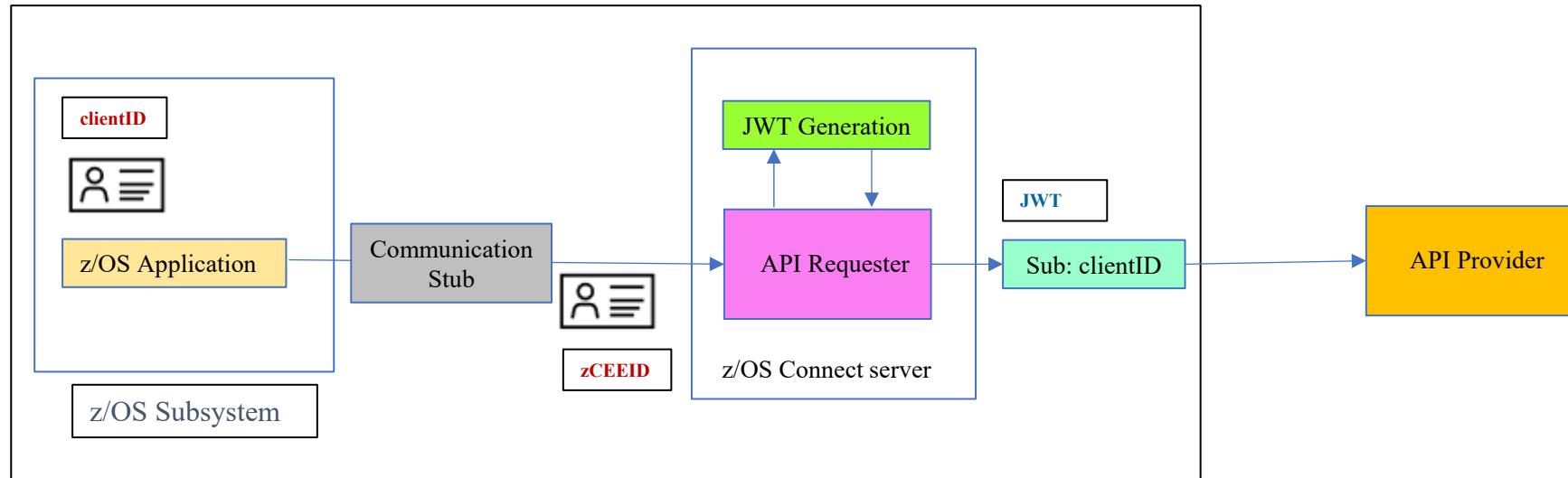
## Response Token

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

# z/OS Connect JWT Generation – V3.0.43



# API Requester – JWT Generation



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

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

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

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

# SAF SURROGAT Resources for identity assertion



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

- *su*
- *cd /usr/lib/java\_runtime*
- *extattr +p libifaedjreg64.so*

- *Permit the server's SAF 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 the connection identity*

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

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

# SAF SURROGAT Resources for JWT Generation



*Define a SURROGAT profile for the client identity and permit access to zCEEID*

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

*Define a SURROGAT profile for the client identity and permit access to zCEEID*

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

# 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" />
```

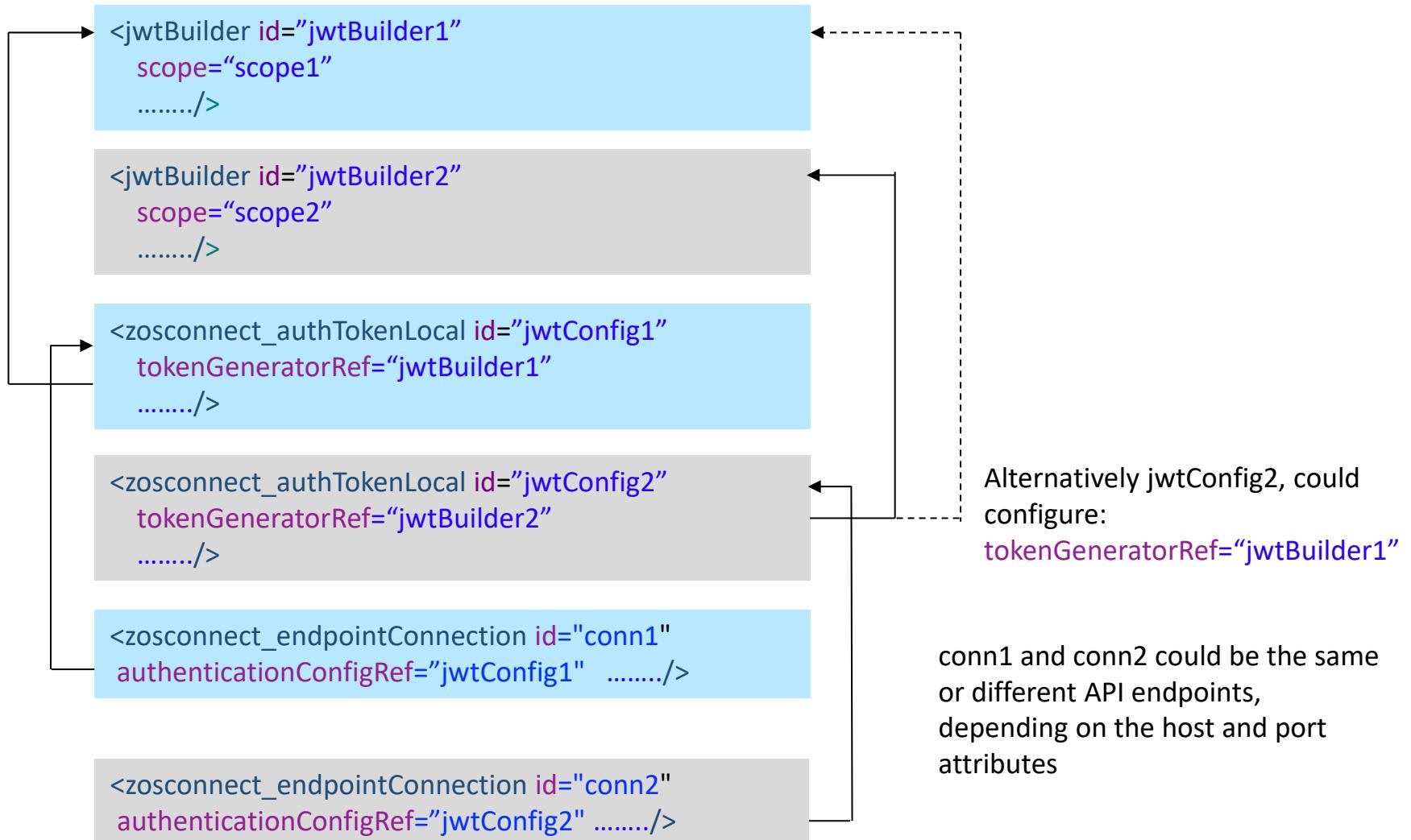
Configure the Liberty jwtBuilder element in server.xml.

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



## In this presentation we covered

- OMVS, Liberty, and RACF Security Review
- z/OS Connect Security Overview
- Authentication
  - Basic Authentication
  - Mutual Authentication using TLS
  - Third Party Tokens
- Encryption and Message Integrity using TLS
- Authorization
- Propagating identities to z/OS subsystems
- z/OS Connect API Requester and third-party tokens

# Summary

- Remember that because z/OS Connect EE is based on Liberty, it benefits from a wide range of Liberty security capabilities
- Security design needs to consider
  - Authentication
  - Encryption
  - Authorization
- Understand your security requirements, identify the waypoints and their security requirements.
- Consider security requirements from ending to beginning (not necessarily from beginning to end), e.g.
  - Do you need to flow the original authenticated identity all the way to the end?
  - Do you need to map individual identities to an application or server identity?

# Step by step exercises available on GitHub

The screenshot shows a web browser window displaying a GitHub repository. The URL in the address bar is <https://github.com/ibm-wsc/zCONNEE-Wildfire-Workshop/tree/master/security>. The repository name is **ibm-wsc / zCONNEE-Wildfire-Workshop**. The 'Code' tab is selected. The 'Security' section is visible in the navigation bar. The main content area shows a list of files under the 'security' branch. A file named 'Delete ZCEEGRPS.jcl' was deleted by user 'jd'. Other files listed include 'zCEE Customization Basic Configuration.pdf', 'zCEE Customization Basic Security.pdf', 'zCEE Customization Security and CICS.pdf', 'zCEE Customization Security and DB2.pdf', 'zCEE Customization Security and JWT Tokens.pdf', 'zCEE Customization Security and MQ.pdf', 'zCEE Customization Security when accessing an IMS Data...', 'zCEE Customization Security when accessing an IMS Tran...', 'zCEE Customization Security with MVS Batch.pdf', and 'zOS Connect EE V3 Advanced Topics Guide.pdf'. All files were added via upload by user 'emitchj'.

File	Added By	Time Ago
Delete ZCEEGRPS.jcl	jd	24 days ago
zCEE Customization Basic Configuration.pdf	emitchj	16 days ago
zCEE Customization Basic Security.pdf	emitchj	16 days ago
zCEE Customization Security and CICS.pdf	emitchj	16 days ago
zCEE Customization Security and DB2.pdf	emitchj	16 days ago
zCEE Customization Security and JWT Tokens.pdf	emitchj	16 days ago
zCEE Customization Security and MQ.pdf	emitchj	24 days ago
zCEE Customization Security when accessing an IMS Data...	emitchj	16 days ago
zCEE Customization Security when accessing an IMS Tran...	emitchj	16 days ago
zCEE Customization Security with MVS Batch.pdf	emitchj	16 days ago
zOS Connect EE V3 Advanced Topics Guide.pdf	emitchj	24 days ago

