ZCADMIN – IBM z/OS Connect Administration

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

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

Notes and Disclaimers

- Additional information included in this presentation was distilled from years of experience implementing security using RACF with z/OS products like CICS, IMS, Db2, MQ, etc. as well as Java runtimes environments like WebSphere Application Server and WebSphere Application Server Liberty which is commonly called Liberty.

- There will be additional information on slides that will be designated as Tech/Tips. These contain information that at perhaps at least interesting and hopefully, useful to the reader.

- **IBM z/OS Connect (OpenAPI 2)** refers the z/OS Connect EE product prior to service level V3.0.55. **IBM z/OS Connect (OpenAPI 3)** refers to the additional functions and features added with service level V3.0.55. Important - servers configured for OpenAPI 2 can will continue to operate as is with service level V3.0.55 and later.

- A z/OS or a Java or a Liberty or a z/OS Connect OpenAPI 2, or a z/OS Connect OpenAPI 3 icon will appear on slides where the information is specific to these products. 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.

mitchj@us.ibm.com

Agenda

- **OMVS, Liberty, z/OS Connect configuration**
- **RACF, Liberty and z/OS Connect Security**
- **Connecting z/OS Connect servers to other z/OS subsystems**
- **Useful Liberty features and MVS commands**
- **Where do I look when things go wrong?**
- **Managing and Monitoring Liberty and z/OS Connect**
- **Additional Material - sample administrative JCL**

mitchj@us.ibm.com

Let’s start by reviewing some of the basic Liberty, OMVS, z/OS Connect configuration details and options

mitchj@us.ibm.com

Begin by verifying the Java and OMVS environments are ready*

Basic system configuration settings which have more than once caused issues

- Prevent out-of-memory or other storage issues:
  - Verify the Java runtime is not being limited by system parameters, e.g., *MAXASSIZE (2 147 483 647*), *MAXTHREADS*, etc., for details see *BPXPRM setting* at URL https://www.ibm.com/docs/en/sdk-java-technology/8?topic=SSYKE2_8.0.0/com.ibm.java.vm.80.doc/docs/j9_configure_zos_bpxprm.html
  - Check the value of *ASSIZEMAX* in the OMVS segments of the identifies involved and ensure it is adequate, see *MAXASSIZE* above.
  - Exclude OMVS from any IEFUSI exit, SUBSYS(OMVS,NOEXITS) in PARMLIB member *SMFRPMxx*.
  - Verify the JCL MEMLIMIT parameter is within reason for your system.
- Start an OMVS shell session and verify that Java is fully operational by entering command ***java –version***, you see should results like this:

```
java version "1.8.0_301"
Java(TM) SE Runtime Environment (build 8.0.6.35 - pmz6480sr6fp35-20210714_01(SR6 FP35))
IBM J9 VM (build 2.9, JRE 1.8.0 z/OS s390x-64-Bit Compressed References 20210622_7763 (JIT enabled, AOT
enabled)
OpenJ9   - b1f3adb
OMR      - c2f4a18
IBM      - c24a144)
JCL - 20210625_01 based on Oracle jdk8u301-b09
```

- Verify that RACF identities associated with started tasks have OMVS segments with UIDs and GIDs and valid HOME directories and that the identities can invoke Java commands.

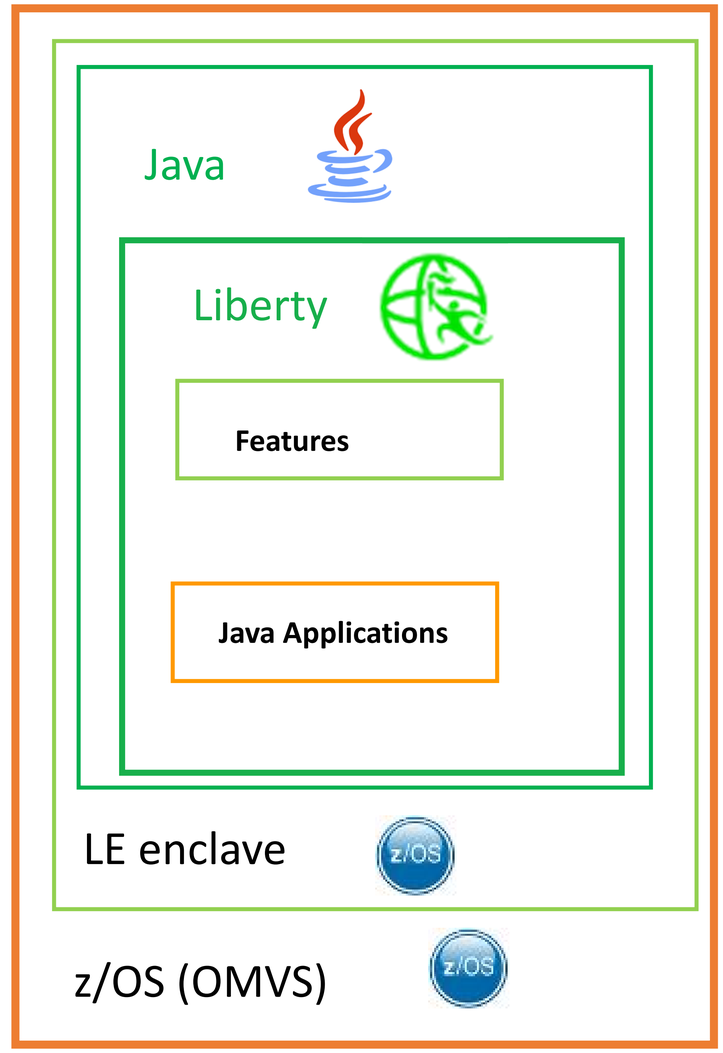
- Verify the *zconsetup* script has been executed. My recommendation is to execute this script in the SMP/E target environment, otherwise it will be lost when service is applied and propagated to other images.

mitchj@us.ibm.com

*be sure the power plug is plugged into the outlet

Think of a Liberty server consisting as a stack of software layers

- That is started by an OMVS script that starts the Java environment (in an OMVS process).
- Runs in a Language Environment (LE) enclave configured to support OMVS and Java processes.
- Liberty is a Java application and on z/OS, there are features that provide access to z/OS services and facilities (e.g., SAF, WLM, RRS, SMF, JCL, started tasks, etc.).
- The Liberty Java application provides an exeuction environment for multiple concurrent Java applications.



Knowing the different layers and their relationship is important regarding

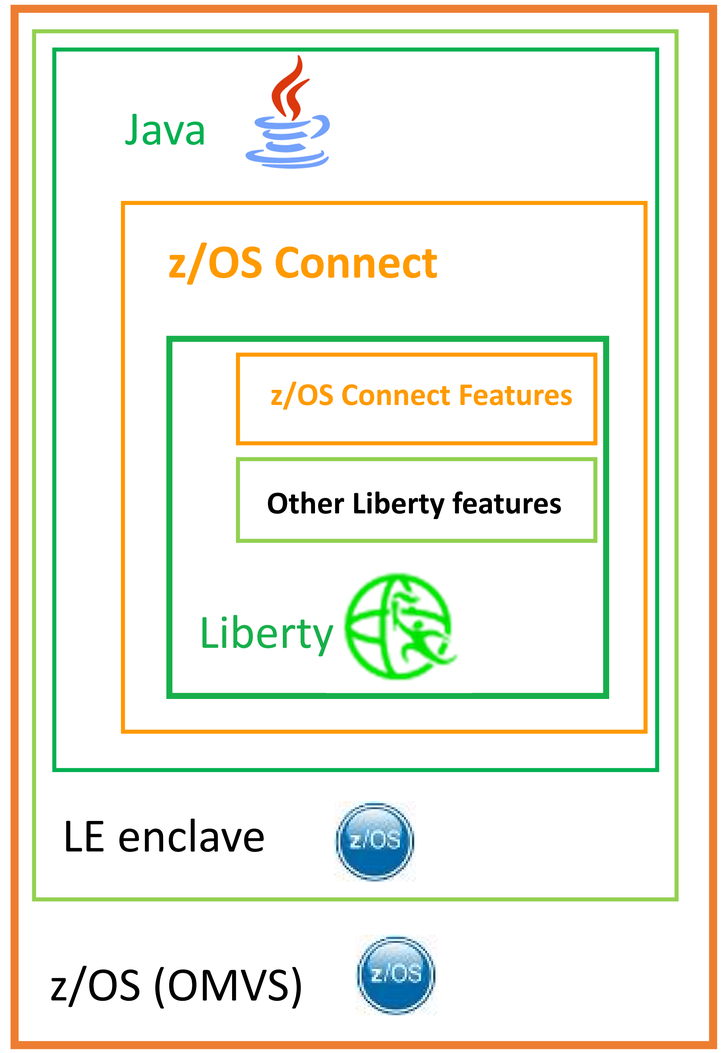
- Understanding which layer a configuration options, e.g., environment variables, etc., applies.
- Monitoring and understanding the health of the server
- Performing problem determination and performance tuning

mitchj@us.ibm.com

Adding a z/OS Connect Liberty server adds an additional layer

z/OS Connect is both a Java application and a set of Liberty features

- The z/OS Connect Java application provides code that spawns a Liberty process *
- The z/OS Connect Liberty features provides a REST interface to user application artifacts that access to common z/OS subsystems, e.g., CICS, Db2, IMS, MQ, etc. and external REST endpoints.



* z/OS Connect starts a Liberty process using a system programming interface (SPI). See the Note regarding environment variables and jvm.options and server.env files at URL https://www.ibm.com/docs/en/was-liberty/zos?topic=liberty-embedding-server-in-your-applications regarding restrictions in this environment.

mitchj@us.ibm.com

Liberty servers are created by using the Liberty *server* command

To create a new Liberty server, use the *server create* command, as in:

***server create* serverName**

- Where ***serverName*** is any value you wish, such as ***wlpopsrv*** or ***wlpOpenIDAuthServer*** and this value will be the name of the server instance. The default value is *defaultServer*
- Environment variable ***WLP_USER_DIR*** must be set to determine the location of the configuration directory and files created by this command. The constant **servers** is appended to the value of this variable, e.g., **{$WLP_USER_DIR}/servers** and the server's name is appended to this root directory and full directory path is the location where the server's configuration files, and default directories are created, e.g., **{$WLP_USER_DIR}/servers/serverName.** The WLP_USER_NAME variables is required when starting a server and must be the same value used when the server was created. There is no default value for a Liberty server.

Note: the name of the server does not have to be same as the started task name, as shown in this example (note how the value for WLP_USER_DIR is provided by the PATH of the WLPUDIR DD statement):

```
//WLPOPID PROC PARMS='wlpOpenIDAuthServer'
//*
// SET INSTDIR='/usr/lpp/liberty_zos/18.0.0.1'
// SET USERDIR='/var/wlp'
//*
//STEP1   EXEC PGM=BPXBATSL,REGION=0M,TIME=NOLIMIT,
//  PARM='PGM &INSTDIR./lib/native/zos/s390x/bbgzsrv &PARMS'
//WLPUDIR  DD PATH='&USERDIR.'
//STDOUT   DD SYSOUT=*
//STDERR   DD SYSOUT=*
//*MSGLOG   DD SYSOUT=*
//*STDENV   DD PATH='/etc/system.env',PATHOPTS=(ORDONLY)
```

mitchj@us.ibm.com

Creating a server creates the required server directories and an initial server configuration file, e.g., server.xml

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

    <!-- Enable features -->
    <featureManager>
          <feature>apiDiscovery-1.0</feature>
          <feature>appSecurity-2.0</feature>
          <feature>beanValidation-1.0</feature>
          <feature>jaxb-2.2</feature>
          <feature>jaxrs-1.1</feature>
          <feature>jsf-2.0</feature>
          <feature>jsp-2.3</feature>
    </featureManager>

    <httpEndpoint id="defaultHttpEndpoint"
                  httpPort="29080"
                  host="*"
                  httpsPort="29443" />

    <!-- Automatically expand WAR files and EAR files -->
    <applicationManager autoExpand="true"/>

    <enterpriseApplication
         id="Trader_EAR"
         location="Trader_EAR.ear"
         name="Trader_EAR">
         <classloader delegation="parentLast"/>
     </enterpriseApplication>

</server>
```

A sample server.xml configuration file

The directory structure is located at {$WLP_USER_DIR}/servers/serverName.

Add or remove features as needed in the featureManager configuration elements.

Configure connectivity in httpEndpoint configuration elements

Add other configuration elements as needed.

mitchj@us.ibm.com

z/OS Connect servers are created using the z/OS Connect *zosconnect* command

To create a z/OS Connect server, use the *zosconnect* command using one of these templates, as in:

zosconnect create serverName --template=templateName

Where ***templateName*** can be:

- ***zosconnect:apiRequester*** for an OpenAPI2 z/OS Connect API requester enabled server
- ***zosconnect:default*** template for base OpenAPI2 z/OS Connect servers
- ***zosconnect:openApi3*** template for base OpenAPI3 z/OS Connect native servers
- ***zosconnect:openApi3Requester*** template for base OpenAPI3 z/OS Connect native API requester servers

- Where ***serverName*** is any value you wish, such as ***zceesrvr*** or ***zCEEServer***, and this value will be the name of the server instance. The templates can be found in directory */usr/lpp/IBM/zosconnect/v3r0/runtime/templates/servers.*

- Environment variable ***WLP_USER_DIR*** will be used to set the location of the configuration directory and files created by this command, default location is */var/zosconnect/servers* where */var/zosconnect* is default value for *WLP_USER_DIR* for z/OS Connect.

- The zosconnect:openApi3 template installs feature **zosconnect:zosConnect-3.0**. z/OS Connect service provider features, e.g., zosconnect:cics-1.0, zosconnect:mqService-1.0, zosconnect:dbService-1.0 and imsmobile:imsmobile-2.0 have dependencies on feature **zosconnect:zosconnect-2.0** and are not compatible with feature **zosconnect:zosConnect-3.0**. z/OS Connect XML configuration attributes other than **zosconnect_cicsIpicConnection** and **zosconnect_db2Connection** are not recognized in an z/OS Connect OpenAPI 3 server.

There are other templates, but they are essentially only useful as samples of service provider configuration options.

mitchj@us.ibm.com

Issues with permission bits and ownership and group access for directories and files is a common problem.

Differences between z/OS Connect OpenAPI2 and OpenAPI2 server.xml files

default template – OpenAPI 2 server.xml configuration file

```
<?xml version="1.0" encoding="UTF-8"?>
<server description="new server">
    <!-- Enable features -->
   <featureManager>
        <feature>zosconnect:zosConnect-2.0</feature>
        <feature>zosconnect:zosConnectCommands-1.0</feature>
        <feature>apiDiscovery-1.0</feature>  *
    </featureManager>

    <!-- To access this server from a remote client add a host attribute
    <httpEndpoint id="defaultHttpEndpoint"
                  host="*"
                  httpPort="9080"
                  httpsPort="9443" />
 <!-- add cors to allow cross origin access, e.g. when using swagger UI
to fetch swagger doc from zOS Connect Enterprise Edition  -->
    <cors id="defaultCORSConfig"
-  -  -  -  -  - 24 Line(s) not Displayed

 <!-- config requires updateTrigger="mbean" for REFRESH command support
-->
 <config updateTrigger="mbean" monitorInterval="500"/>

    <zosconnect_zosConnectManager setUTF8ResponseEncoding="true"/>

    <!-- zosConnect APIs -->
    <zosconnect_zosConnectAPIs updateTrigger="disabled" pollingRate="5s"
    <!-- zosConnect Services -->
 <zosconnect_services updateTrigger="disabled" pollingRate="5s"/>

 <!-- applicationMonitor is not applicable for z/OS Connect EE servers -
->
 <applicationMonitor updateTrigger="disabled" dropinsEnabled="false"/>

</server>
```

openApi3 template – OpenAPI 3 server.xml configuration file

```
<?xml version="1.0" encoding="UTF-8"?>
<server description="new server">
    <!-- Enable features -->
   <featureManager>
        <feature>zosconnect:zosConnect-3.0</feature>
        <feature>openapi-3.0</feature>
</featureManager>

    <!-- To access this server from a remote client add a host attribute
    <httpEndpoint id="defaultHttpEndpoint"
                  host="*"
                  httpPort="9080"
                  httpsPort="9443" />
-  -  -  -  -  -  -  -  -  -  -  -  -  -  -  - 12 Line(s) not Displayed
    <!-- config requires updateTrigger="mbean" for REFRESH command support
   config updateTrigger="mbean"/>

    <!-- applicationMonitor requires updateTrigger="mbean" for REFRESH command
support -->
    <applicationMonitor updateTrigger="mbean" dropinsEnabled="false"/>

    <!-- Automatic expansion of WAR files is required for z/OS Connect native
servers running the zosConnect-3.0 feature  -->
    <applicationManager autoExpand="true" />

    <!-- APIs are deployed as WAR files and a webApplication element must be
used to specify the location of the API WAR and optionally the name of the API
-->
    <webApplication id="My API" location="${server.config.dir}/apps/api.war"
name="MyAPI"/>

</server>
```

Note there are no *zosconnect* or *cors* configuration elements present with Open API 3.

mitchj@us.ibm.com

* Include these features if not already present.

A Liberty JCL procedure versus a z/OS Connect JCL procedure

```
//ZCEESRVR PROC PARMS='serverName'
//*
// SET ZCONHOME='/usr/lpp/IBM/zosconnect/v3r0'
// SET INSTDIR='/usr/lpp/liberty_zos/21.0.0.9'
//*
//ZCON     EXEC PGM=BPXBATSL,REGION=0M,MEMLIMIT=8G,
//  PARM='PGM &ZCONHOME./bin/zosconnect run &PARMS. --clean'
//  PARM='PGM &INSTDIR./lib/native/zos/s390x/bbgzsrv &PARMS'
//STDOUT   DD   SYSOUT=*
//STDERR   DD   SYSOUT=*
//STDIN    DD   DUMMY
//STDENV   DD   *
_BPX_SHAREAS=YES
_CEE_RUNOPTS=HEAPPOOLS(ON),HEAPPOOLS64(ON)
JAVA_HOME=/usr/lpp/java/J8.0_64
WLP_USER_DIR=/var/zosconnect
JVM_OPTIONS=-Dcom.ibm.ws.zos.core.angelName=zCEEAngel -Xmx512m
OPENJ9_JAVA_OPTIONS=-Xoptionsfile=/var/zcee/properties/myServer.property
```

OMVS
LE
JAVA
LIBERTY
z/OS Connect

mitchj@us.ibm.com

https://www.ibm.com/docs/en/zos/2.5.0?topic=bpxbatch-guidelines-defining-stdenv

Useful JCL parameters and environment variables

- **serverName** – This JCL parameter is used to provides the name of the server and used to set the value of several environment variables.
- **WLP_USER_DIR** – This environment variable is used when a server is created to determine where the server's working directories will be created and where the initial *server.xml* file will be created. This variable is also to by the runtime environment to locate the server's existing working directories and the *server.xml* file. Also, the WLP_USER_DIR is used to set the shared variables.
- **JAVA_HOME** – The OMVS directory where the Java executables (*/bin* directory) can be located.
- **JVM_OPTIONS – A z/OS Connect environment variables that provides Java options and/or system properties. The contents of *JVM_OPTIONS* is added to the *java* command line in the *zosconnect* startup script.**
- **IBM_JAVA_OPTIONS –** An IBM JAVA environment variable (deprecated and eventually will be replaced by *environment variable OPENJ9_JAVA_OPTIONS*). Environment variable *IBM_JAVA_OPTIONS* variable can be used to provide Java options and/or system properties.
- **OPENJ9_JAVA_OPTIONS –** An OpenJ9 environment variable (eventually will replace the deprecated environment variable *IBM_JAVA_OPTIONS*). Environment variable *OPENJ9_JAVA_OPTIONS* variable can be used to provide Java options and/or system properties.

Note: Any Java option or system property using JVM_OPTIONS supersedes the same Java non-standard options or system property when provided by IBM_JAVA_OPTIONS or OPEN9_JAVA_OPTIONS

The following environment variables are automatically set in a Liberty server and can be used as variables in the server XML configuration files.

- **server.config.dir** – whose value will automatically be set to the value of variable *WLP_USER_DIR* concatenated with the name of the server, e.g. */var/zosconnect/servers/serverName*
- **shared.config.dir** – whose value will automatically be set to the value of variable *WLP_USER_DIR* concatenated with /shared/config, e.g. */var/zosconnect/shared/config*
- **shared.app.dir** – whose value will automatically be set to the value of variable *WLP_USER_DIR* concatenated with /shared/apps, e.g. */var/zosconnect/shared/apps*
- **wlp.server.name -** whose value will automatically be set to the value of the server as provided in the PARMS value provided in the JCL procedure.

mitchj@us.ibm.com

Tech-Tip: The *zconsetup* script must be invoked once per LPAR per product install path

The *zconsetup* script creates a symbolic link from the WLP *../v3r0/wlp/etc* directory in the product path (normally R/O) to a local R/W directory (creating a default configuration and local extension directories). This provides a means for administrator to configure the image.

```
JOHNSON:/usr/lpp/IBM/zosconnect/v3r0/wlp/etc: ls -al
total 32
drwxrwxr-x   2 OMVSKERN 0            8192 Jun 24 10:24 .
drwxrwxr-x  10 OMVSKERN 0            8192 Jun 24 10:24 ..
lrwxrwxrwx   1 990023   0              31 Jul 27  2020 extensions -> /var/zosconnect/v3r0/extensions
```

```
/var/zosconnect
  /servers
  /v3r0
    /extensions
      imsmobile.properties
      zosconnect.properties
      filemanager.properties
      omegamon.properties
```

com.ibm.websphere.productId=com.ibm.ims.mobile
com.ibm.websphere.productInstall=imsmobile/wlp-ext

com.ibm.websphere.productId=com.ibm.zosconnect
com.ibm.websphere.productInstall=runtime

Relative v. absolute path

- This directory structure and contents is created by by invoking the *zconsetup* script and **must be created on each LPAR** on which z/OS Connect will execute. This is how the z/OS Connect Liberty server locates service provider executables. Note: the *com.ibm.websphere.productInstall* directive value that is **relative** to directory /usr/lpp/IBM/zosconnect/v3r0.
- Not creating this link will cause messages *CWWKF0001E: A feature definition could not be found for zosconnect*:…. or *CWWKE0054E: Unable to open /usr/lpp/IBM/zosconnect/v3r0/wlp/etc/extensions/zosconnect.properties*

Tech-Tip: Invoke the *zconsetup* script in the SMP/E TLIB file system.

mitchj@us.ibm.com

Tech-Tip: OMVS security - A quick review of Unix permissions bits

	Owner	Group	Other
	Read Write Execute	Read Write Execute	Read Write Execute
Bit	1 1 1	1 0 1	0 0 0
Base-2 Value	[4] [2] [1]	[4] [2] [1]	[4] [2] [1]
	4 + 2 + 1 =	4 + 0 + 1 =	0 + 0 + 0 =
	7 The owner has READ, WRITE and EXECUTE	**5** The group has READ and EXECUTE, but not WRITE	**0** Others have nothing

The **owner** of the file or directory

chmod –R* u+rwx zceesrv1

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

chmod g+rwx server.xml

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

-R* indicates recursion

mitchj@us.ibm.com

A z/OS Connect server configuration directories and files

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

```
export JAVA_HOME=<path_to_64_bit_Java>
export WLP_USER_DIR=/var/zosconnect
./zosconnect create zceesrvr
--template= zosconnect:apiRequester
```

```
                                Owner   Group
/var/zosconnect          750    LIBSERV LIBGRP
  /servers               750    LIBSERV LIBGRP
    /zceesrvr            750    LIBSERV LIBGRP
      /apps              750    LIBSERV LIBGRP
      /configDropins     750    LIBSERV LIBGRP
        /overrides       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
      /security          777    LIBSERV LIBGRP
      server.xml         640    LIBSERV LIBGRP
/shared                  750    LIBSERV LIBGRP
   /apps                 750    LIBSERV LIBGRP
   /config               750    LIBSERV LIBGRP
```

The create command 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?

  Sharing passwords is a bad practice. Better to take advantage SAF SURROGAT so permitted users can switch to the owning ID so they can make changes. In fact, LIBSERV should be a PROTECTED identity with no password in the first place.

- If you have multiple people with a need to read or update configuration files, do you simply connect them to LIBGRP?

  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.

- **The *shared* directory structure is shared among all servers started with a common value for the *WLP_USER_DIR* environment variable. Each server can access common server configuration files using the *shared.config.dir* environment variable or access web applications using the *shared.app.dir* environment variable.**

* Only created when using the apiRequester template.

mitchj@us.ibm.com

Tech-Tip: Using permission bits to control access

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

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

```
/var/zosconnect          751  LIBSERV LIBGRP
  /servers               751  LIBSERV LIBGRP
    /zceesrv1            751  LIBSERV LIBGRP
      /apps              761  LIBSERV LIBGRP
      /configDropins     761  LIBSERV LIBGRP
         /overrides      761  LIBSERV LIBGRP
      /logs              771  LIBSERV LIBGRP
         messages.log    644  LIBSERV LIBGRP
      /resources         751  LIBSERV ADMGRP
        /security        777  LIBSERV LIBGRP
        /zosconnect      751  LIBSERV ADMGRP
         /apis           761  LIBSERV ADMGRP
         /apiRequesters  761  LIBSERV ADMGRP
         /rules          761  LIBSERV ADMGRP
         /services       761  LIBSERV ADMGRP
       server.xml        460  LIBSERV ADMGRP
/shared                  750  LIBSERV LIBGRP
   /apps                 750  LIBSERV LIBGRP
   /config               750  LIBSERV LIBGRP
```

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

Sample OMVS commands to manage permission bits

```
export WLP_USER_DIR=/var/zosconnect
cd $WLP_USER_DIR
chmod o+x –R servers
chmod o+x servers/zceesrvr/resources
chmod –R o+x servers/zceesrvr/resources/*
chmod g+r –R servers
chmod g+r servers/zceesrvr/resources
chmod –R g+r servers/zceesrvr/resources/*
chmod g+w server.xml
```

Warning: Access for Owner, Group(g), Others(o) depend on user ID (UID) and group ID (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 utilities like ADRDSSU.

mitchj@us.ibm.com

Tech Tip: Use multiple mount points and dedicated ZFS file systems

Create the mount points and mount file systems prior to running the *zconsetup* script

```
mkdir -p /var/zosconnect
mkdir -p /var/zosconnect/servers
mkdir -p /var/zosconnect/group1
mkdir -p /var/zosconnect/group2
mkdir -p /var/zosconnect/group3
```

SYS1.PARMLIB(BPXPRM##)

```
MOUNT FILESYSTEM('OMVS.ZCEEVAR.ZFS')
      MOUNTPOINT('/var/zosconnect')
      TYPE(ZFS) MODE(READ)
MOUNT FILESYSTEM('OMVS.ZCEE.SERVERS.ZFS')
      MOUNTPOINT('/var/zosconnect/servers')
      TYPE(ZFS) PARM('AGGRGROW') MODE(RDWR)
MOUNT FILESYSTEM('OMVS.ZCEE.GROUP1.ZFS')
      MOUNTPOINT('/var/zosconnect/group1')
      TYPE(ZFS) PARM('AGGRGROW') MODE(RDWR)
MOUNT FILESYSTEM('OMVS.ZCEE.GROUP2.ZFS')
      MOUNTPOINT('/var/zosconnect/group2')
      TYPE(ZFS) PARM('AGGRGROW') MODE(RDWR)
MOUNT FILESYSTEM('OMVS.ZCEE.GROUP.ZFS')
      MOUNTPOINT('/var/zosconnect/group3')
      TYPE(ZFS) PARM('AGGRGROW') MODE(RDWR)
```

- Create a dedicated filesystem for the root z/OS Connect /var directory, e.g., /var/zosconnect/v3r0/extensions. This directory structure can not be changed. This provides portability for migrations and system upgrades. Note: MODE(READ) will apply to /var/zosconnect/servers.

- Create a dedicated filesystem for each set or groups of servers. These filesystems will contain the server configuration directories for 1 or more servers.
- Each server's WLP_USER_DIR environment variable will be set to the mount point, e.g., *WLP_USER_DIR=/var/zosconnect/group1* when the server is created and in the server's startup JCL.

df -P | grep /var/zosconnect

```
Filesystem            512-blocks        Used  Available  Capacity Mounted on
OMVS.ZCEEVAR.ZFS           69120       68658        462       100% /var/zosconnect
OMVS.ZCEE.SERVERS.ZFS     159120       76455      82665        48% /var/zosconnect/servers
OMVS.ZCEE.GROUP1.ZFS      135360        1506     133854         2% /var/zosconnect/group1
OMVS.ZCEE.GROUP2.ZFS     4059360     2591284    1468076        64% /var/zosconnect/group2
OMVS.ZCEE.GROUP3.ZFS      135360       17858     117502        14% /var/zosconnect/group3
```

mitchj@us.ibm.com

Tech-Tip: Use JCL to make the creation and configuration of servers repeatable and portable

And take advantage of RACF SURROGAT and UNIXPRIV resources

Example of using **SURROGAT** privileges

```
//ZCEESRVR JOB 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='zceeSrvr'
// SET TEMPLATE='zosconnect:default'
// SET WLPUSER='/var/ats/zosconnect'
// SET USER='ATSSERV'
// SET GROUP='ATSGRP'
//*************************************************************
//*  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; +
```

Example of using **UNIXPRIV** privileges

```
//ZCEESRVR JOB CLASS=A,REGION=0M,NOTIFY=&SYSUID
//*************************************************************
//*  SET SYMBOLS
//*************************************************************
//EXPORT EXPORT SYMLIST=(*)
// SET JAVAHOME='/usr/lpp/java/J8.0_64'
// SET ZCEEPATH='/usr/lpp/IBM/zosconnect/v3r0'
// SET SERVER='openApi3'
// SET TEMPLATE='zosconnect:openApi3'
// SET WLPUSER='/var/ats/zosconnect'
// SET CONFIG='configDropins/overrides'
// SET USER='ATSSERV'
// SET GROUP='ATSGRP'
//*************************************************************
//*  Step ZCEEAPI3 - Use the zosconnect command to create a server
//*************************************************************
//ZCEEAPI3 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; +
chown -R &USER:&GROUP $WLP_USER_DIR/servers/&SERVER
```

mitchj@us.ibm.com

Tech/Tip: Use SAF SURROGAT resources for administration

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.

mitchj@us.ibm.com

Tech/Tip: z/OS : Also use 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) or the Enable superuser mode(SU) Setup option in ISHELL*
  PERMIT BPX.SUPERUSER CLASS(FACILITY) ID(adminUser) ACC(READ)

- *Refresh the UNIXPRIV and/or FACILITY instorage profiles*
  SETROPTS RACLIST(UNIXPRIV,FACILITY) REFRESH

https://www.ibm.com/support/knowledgecenter/en/SSLTBW_2.4.0/com.ibm.zos.v2r4.bpxb200/usspriv.htm

Use the power of these commands provide carefully and only when necessary

mitchj@us.ibm.com

A more detailed tour of a LPAR’s directories and files

```
/var/zosconnect/v3r0
  /extensions    (see previous slide)


${WLP_USER_DIR}
  /servers                (details on next page)
    /serverName
```

- The *extensions* subdirectory will always be in /var/zosconnect/v3r0

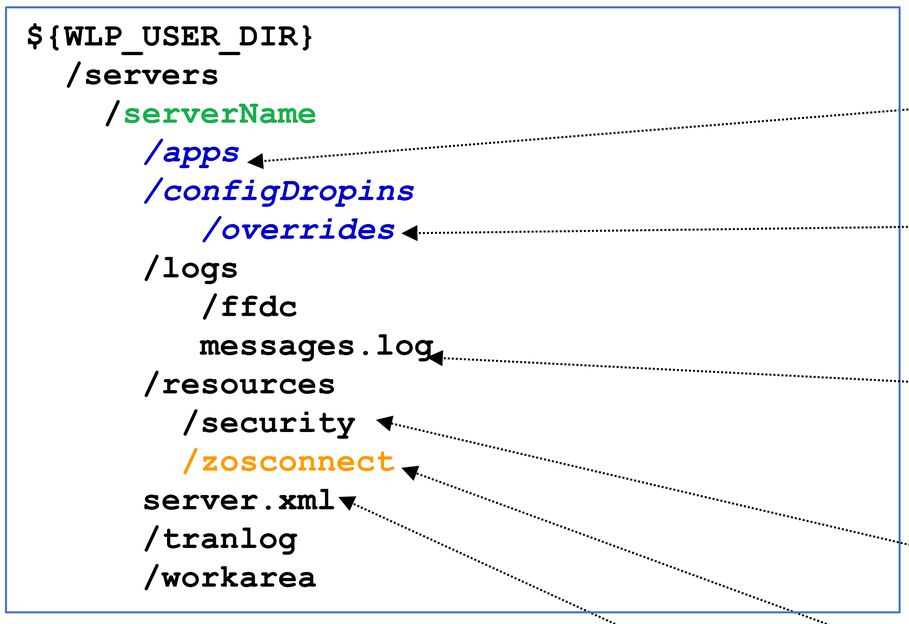
- There can be multiple $WLP_USER_DIR directory on an LPAR
- Each server (serverName) will have a unique subdirectory in the location specified by WLP_USER_DIR, which **defaults** to */var/zosconnect*.
- Important, use the same value for starting a server that was used when the server was created.

- The location of the *serverName* directory is based on the concatenation of the value of the *WLP_USER_DIR* environment variable with the constant *servers* and does not have to be in directory /var/zosconnect.
- The *serverName* directory structure and its initial contents are created by by invoking the *zosconnect create serverName* script.
- *serverName* can be a mount point with a dedicated file system mounted at this mount point (see above). This can be used to isolate servers to dedicated file systems.
- The number, size and output location of messages.log and trace files in the *logs* directory can be controlled with the Liberty <*logging*> configuration element or the output location controlled by using the *com.ibm.ws.logging.log.directory* Java directive as a JVM options override, more on this later.
- #These directories maintain state information and it is a good practice is to add the --*clean* parameter to the server startup JCL, e.g., PARMS='serverName --clean’, especially after service is applied.

mitchj@us.ibm.com

The contents of a server's configuration files

A server's configuration structure looks like this (N.B. OpenAPI 2 and OpenAPI 3 servers do not coexist as shown here):



The */apps* directory is the location to where OpenAPI 3 Web Archive (WAR) files are deployed.

The */configDropins/overrides* directory is the location where server XML configuration files are placed for OpenAPI 3 servers.

The ***messages.log*** file is the key output file for messages about Liberty and the processing taking place in the Liberty server. The output written to this file can be written to the SPOOL by including DD statement MSGLOG in the startup JCL, e.g., //MSGLOG DD SYSOUT=*,FREE=CLOSE,SPIN=(UNALLOC,1M)

The */security* directory contains files **ltpa.keys** and **key.p12**. **ltpa,keys** is the server specific LTPA token. **key.p12** is a self-signed certificate that expires in one year.

The */zosconnect* directory is where the deployed APIs, services, and API requester files will be placed for an OpenAPI 2 server.

The **server.xml** file is the key configuration file. It is here that z/OS Connect definitions go which define the essential configuration of the server and backend connectivity.

The *WLP_USER_DIR* environment variables sets the value of the root directory of the server's configuration files and directories, e.g., WLP_USER_DIR=/var/zosconnect

mitchj@us.ibm.com

Use “include” files to extend and manage a server’s configuration

- 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

server.xml (owned by ID ADMIN1)

```
<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”/>
<featureManager>
    <feature>zosconnect:zosConnect-2.0</feature>
   <feature>zosconnect:zosConnectCommands-1.0</feature>
   <feature>apiDiscovery-1.0</feature>
 <featureManager>
```

db2.xml (owned and managed 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 and managed by a CICS administrator)

```
<server description=“CICS">
<featureManager>  <feature>zosconnect:cicsService-1.0</feature>  </featureManager>
<zosconnect_cicsIpicConnection id="catalog"  host="wg31“ port="1491"/>
<zosconnect_cicsIpicConnection id="cscvinc"  host="wg31“ port="1493"/>
</server>
```

imsDB.xml (owned and managed by a IMS administrator)

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

Nesting of an include file within a include file is possible

mitchj@us.ibm.com

Tech-Tip: Review configuration conflicts

```
ÝAUDIT   ¨ CWWKG0102I: Found conflicting settings for cscvincAPI instance of zosconnect_endpointConnection configuration.
  Property port has conflicting values:
    Value 9443 is set in file:/var/zosconnect/servers/myServer/includes/apiRequesterHTTPS.xml.
    Value 9443 is set in file:/var/zosconnect/servers/myServer/includes/apiRequesterHTTPS.xml.
    Value 9463 is set in file:/var/zosconnect/servers/myServer/includes/oauth.xml.
  Property port will be set to 9463.
  Property host has conflicting values:
    Value https://dvipa.washington.ibm.com is set in file:/var/zosconnect/servers/myServer/includes/apiRequesterHTTPS.xml.
    Value https://dvipa.washington.ibm.com is set in file:/var/zosconnect/servers/myServer/includes/apiRequesterHTTPS.xml.
    Value https://mpz3.washington.ibm.com is set in file:/var/zosconnect/servers/myServer/includes/oauth.xml.
  Property host will be set to https://mpz3.washington.ibm.com.
  Property authenticationConfigRef has conflicting values:
    Value mySAFAuth is set in file:/var/zosconnect/servers/myServer/includes/apiRequesterHTTPS.xml.
    Value myoAuthConfig is set in file:/var/zosconnect/servers/myServer/includes/oauth.xml.
  Property authenticationConfigRef will be set to myoAuthConfig.
```

onConflict="MERGE"	Conflicting elements will be merged, and the last value encountered will be used.
onConflict="REPLACE"	When elements conflict, the element in the included file will be ignored
onConflict="IGNORE"	Conflicting elements in the included file are ignored.

mitchj@us.ibm.com

Use a *bootstrap.properties* file to help customize a server's XML configuration#

zceesrv1's bootstrap.properties

```
httpPort=9080
httpsPort=9443
ipicPort=1491
host=*
cicsHost=wg31.washington.ibm.com
network=ZOSCONN1
applid=ZOSCONN1
com.ibm.ws.zos.core.angelName=namedAngel
```

zceesrv2's bootstrap.properties

```
httpPort=9090
httpsPort=9453
ipicPort=1492
host=wg31.washington.ibm.com
cicsHost=wg31.washington.ibm.com
network=ZOSCONN2
applid=ZOSCONN2
com.ibm.ws.zos.core.angelName=namedAngel
```

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="${host}"
                  httpPort="${httpPort}"
                  httpsPort="${httpsPort}" />
```

Java directives can also be provided.

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}"/>
```

#Located in directory **${server.config.dir}** and uses EBCDIC encoding

mitchj@us.ibm.com

Tech-Tip: One suggestion for managing the server.xml configuration file

Default server.xml configuration file

```
<?xml version="1.0" encoding="UTF-8"?>
<server description="new server">
    <!-- Enable features -->
   <featureManager>
 - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
    </featureManager>

    <!-- To access this server from a remote client add a host attribute
    <httpEndpoint id="defaultHttpEndpoint"
                  host="*"
                  httpPort="9080"
                  httpsPort="9443" />
 <!-- add cors to allow cross origin access, e.g. when using swagger
    <cors id="defaultCORSConfig"
          domain="/"
          allowedOrigins="*"
          allowedMethods="GET, POST, PUT, DELETE, OPTIONS"
          allowedHeaders="Origin, Content-Type, Authorization, Cache-Con
          allowCredentials="true"
          maxAge="3600"/>
-  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  - 30 Line(s) not Displayed
</server>
```

Modified server.xml configuration file

```
<server description="zCEE Server">
<include location="${server.config.dir}/includes/safSecurity.xml"/>
<include location="${server.config.dir}/includes/ipicIDProp.xml"/>
<include location="${server.config.dir}/includes/keyring.xml”/>
<include location="${server.config.dir}/includes/groupAccess.xml"/>
<include location="${server.config.dir}/includes/shared.xml"/>
<include location="${server.config.dir}/includes/apiRequesterHTTPS.xml"/
<include location="${server.config.dir}/includes/imsDatabase.xml"/>
 <!-- Enable features -->
   <featureManager>
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
    </featureManager>
 <!-- To access this server from a remote client add a host attribute
    <httpEndpoint id="defaultHttpEndpoint"
                  host="${host}"
                  httpPort="${httpPort}"
                  httpsPort="${httpsPort}" />
-  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  - 36 Line(s) not Displayed
</server>
```

The simplifies administration by :

- Using a *bootstrap.properties* file to customize the ports in the *server.xml* file.
- Using “include” statements to make further changes such as adding additional features and additional XML configuration elements.
- Review https://www.ibm.com/docs/en/was-liberty/nd?topic=liberty-configuration-element-merging-rules to understand merging rules.
- **Consider providing configuration elements by placing server XML files in the *…/configDropins/original* subdirectory.**

mitchj@us.ibm.com

***Added in V3.0.48**

Providing STDENV input in a JCL Procedure

Use the STDENV DD statement to scale servers and share configuration properties horizontally

```
//ZCON      EXEC PGM=BPXBATSL,REGION=0M,MEMLIMIT=8G,
//            PARM='PGM &ZCONHOME./bin/zosconnect run &PARMS. --clean'
//STDERR    DD SYSOUT=*,FREE=CLOSE,SPIN=(UNALLOC,1M)
//STDOUT    DD SYSOUT=*
//STDIN     DD DUMMY
//STDENV    DD PATH='/var/zcee/properties/&PARMS..property',
//             PATHOPTS=ORDONLY
 or
//STDENV    DD DISP=SHR,DSN=JOHNSON.ZCEE.STDENV(COMMON)
//            DD DISP=SHR,DSN=JOHNSON.ZCEE.STDENV(ZCEESRVR)
```

Either one OMVS property file or multiple PDS members.

The last occurrence environment variable encountered determines the value of the environment variable.

```
Member COMMON
_BPX_SHAREAS=YES
_CEE_RUNOPTS=HEAPPOOLS(ON),HEAPPOOLS64(ON)
JAVA_HOME=/usr/lpp/java/J8.0_64
ZCON_ENV_DEBUG=TRUE
WLP_USER_DIR=/var/zosconnect
```

```
Member ZCEESRVR
OPENJ9_JAVA_OPTIONS=-Dcom.ibm.ws.zos.core.angelName=ZCEEANGL
JVM_OPTIONS=-Xoptionsfile=/var/zcee/properties/javaHCD.property -Xmx512m -verbose:sizes
JAVA_HOME=/u/johnson/java/J8.0_64
WLP_USER_DIR=/var/ats/zosconnect
```

Which value used for a Java option or property depends on which environment variable is used to specify the option or property.

Using //SYSIN DD * is discouraged because of the 80 character record limit.

mitchj@us.ibm.com

BTW, the DCB characteristics the STDENV PDS/PDSE data set were (LRECL=400,BLKSIZE=32400).

STDENV DD concatenation and environment variables precedence order

```
Member COMMON
_BPX_SHAREAS=YES
_CEE_RUNOPTS=HEAPPOOLS(ON),HEAPPOOLS64(ON)
JAVA_HOME=/usr/lpp/java/J8.0_64
ZCON_ENV_DEBUG=TRUE
WLP_USER_DIR=/var/alt/zosconnect
```

Green indicated the environment variable, Java option(-X) or system property(-D) that are used.
Red indicates the environment variable, Java option(-X) or system property(-D) that are ignored.

```
Member OPENJ9
OPENJ9_JAVA_OPTIONS=-verbose:sizes -Xms75m -Dcom.ibm.ws.zos.core.angelName=OPENJ9
-Dcom.ibm.ws.logging.message.file.name=openj9.log #
```

```
Member IBMOPTS
IBM_JAVA_OPTIONS=-verbose:jni -Xms80m -Dcom.ibm.ws.logging.message.file.name=ibmopts.log
-Dcom.ibm.ws.zos.core.angelName=IBMOPTS #
```

Aaron D Lewis <adlewis@us.ibm.com>; Neil Trapani <ntrapani@us.ibm.com>; Tony AVDIU <favdiu@us.ibm.com>;

```
Member JVMOPTHC
JVM_OPTIONS=-Xoptionsfile=/var/zcee/properties/javaHCD.property -Dcom.ibm.ws.zos.core.angelName= -Xmx256m -verbose:sizes
```

```
Member JAVAHOME
JAVA_HOME=/u/johnson/java/J8.0_64
```

```
Member ZCEEANGL
OPENJ9_JAVA_OPTIONS=-Dcom.ibm.ws.zos.core.angelName=ZCEEANGL -Dcom.ibm.ws.logging.message.file.name=zceeangl.log -Xmx16m -Xms60m
-verbose:gc #
```

```
Member WLPUSER
WLP_USER_DIR=/var/zosconnect
```

Default settings for the OpenJ9 VM https://www.ibm.com/docs/en/sdk-java-technology/8?topic=reference-default-settings

mitchj@us.ibm.com

These entries do not span multiple lines in the PDS member(the contents are continuous on one line).

Tech/Tip: Liberty Java Directives for controlling output

com.ibm.ws.logging.console.format (consoleFormat)- The required format for the console. Valid values are basic or json format.

com.ibm.ws.logging.console.log.level (consoleLogLevel) - This filter controls the granularity of messages that go to the console. The valid values are INFO, AUDIT, WARNING, ERROR, and OFF. By default, the console log level is set to AUDIT.

com.ibm.ws.logging.hideMessage (hideMessage) - Use this attribute to configure the messages that you want to hide from the *console.log* and *message.log* files. If the messages are configured to be hidden, then they are redirected to the *trace.log* file.

com.ibm.ws.logging.log.directory (logDirectory) - Use this attribute to set a directory for all log files, excluding the *console.log* file, but including *FFDC*. The default log location path is *WLP_OUTPUT_DIR/serverName/logs*

com.ibm.ws.logging.max.file.size (maxFileSize) - The maximum size (in MB) that a log file can reach before it is rolled. The Liberty runtime does only size-based log rolling. To disable this attribute, set the value to 0. The maximum file size is approximate. By default, the value is 20.

com.ibm.ws.logging.max.files (maxFiles) - If a maximum file size exists, this setting is used to determine how many of each of the log files are kept. This setting also applies to the number of exception logs that summarize exceptions that occurred on any day. So, if this number is 10, you might have 10 message logs, 10 trace logs, and 10 exception summaries in the *ffdc* directory. The default value is 2.

com.ibm.ws.logging.message.format (messageFormat) - The required format for the messages.log file. Valid values are basic or json format. By default, messageFormat is set to the environment variable WLP_LOGGING_MESSAGE_FORMAT (if set) or basic.

com.ibm.ws.logging.trace.file.name (traceFileName) - The *trace.log* file is only created if additional or detailed trace is enabled. *stdout* is recognized as a special value; and causes trace to be directed to the original standard out stream.

bootstrap.properties example:

```
com.ibm.ws.logging.message.file.name=basqstrtMessages.log
com.ibm.ws.logging.log.directory=/u/common/logs
```

N.B. *consoleFormat, logDirectory*, etc. can be specified in the <*logging/*> Liberty configuration element. Note the recommendation for the attributes in red is for them to be provided in Java directives.

mitchj@us.ibm.com

https://www.ibm.com/docs/en/was-liberty/zos?topic=liberty-logging-trace

Tech-Tip: Consider using symbolic links especially as an administrative shortcut

- Create an “administration” subdirectory, e.g., *LiberyConfig* in directory */var*
- Then create a symbolic link in the “administration” directory to each Liberty server’s configuration directory and other frequently accessed directories.

```
ls -al /var/LibertyConfig
drwxrwxrwx   4 JOHNSON  SYS1        8192 Aug 16 12:23 .
drwxrwxrwt  25 OMVSKERN SYS1        8192 Aug 16 11:56 ..
lrwxrwxrwx   1 JOHNSON  SYS1          57 Aug 16 12:22 CSCWLP -> /var/wlp/cics/CICS53Z/CSCWLP/wlp/usr/servers/defaultServer
lrwxrwxrwx   1 JOHNSON  SYS1          57 Aug 16 12:22 CICSWLP -> /var/wlp/cics/CICS53Z/CICSWLP/wlp/usr/servers/cicswlp
drwxrwxrwx   2 JOHNSON  SYS1        8192 Aug 16 15:30 hcd
lrwxrwxrwx   1 JOHNSON  SYS1          27 Jun 10 15:55 includes -> /global/zosconnect/includes
lrwxrwxrwx   1 JOHNSON  SYS1          28 Aug 16 10:12 mqweb -> /var/mqm/mqweb/servers/mqweb
lrwxrwxrwx   1 JOHNSON  SYS1          32 Jun  4 12:49 myServer -> /var/zosconnect/servers/myServer
drwxr-xr-x   2 JOHNSON  SYS1        8192 Aug 16 13:14 properties
lrwxrwxrwx   1 JOHNSON  SYS1          18 Aug 17 12:47 shared -> /var/shared/zosconnect/resources/zosconnect
lrwxrwxrwx   1 JOHNSON  SYS1          24 May 13  2020 walop3a -> /var/wlp/servers/walop3a
lrwxrwxrwx   1 JOHNSON  SYS1          24 May 13  2020 walrp3a -> /var/wlp/servers/walrp3a
lrwxrwxrwx   1 JOHNSON  SYS1          31 May 14  2020 wazs34a -> /var/zosconnect/servers/wazs34a
lrwxrwxrwx   1 JOHNSON  SYS1          24 Aug 16 10:32 wlphats -> /var/wlp/servers/wlphats
lrwxrwxrwx   1 JOHNSON  SYS1          36 Aug 16 10:31 zceeapir -> /var/ats/zosconnect/servers/zceeapir
lrwxrwxrwx   1 JOHNSON  SYS1          39 Aug 16 10:18 zceecics -> /var/cicsts/zosconnect/servers/zceecics
lrwxrwxrwx   1 JOHNSON  SYS1          35 Aug 16 10:31 zceedvm -> /var/ats/zosconnect/servers/zceedvm
lrwxrwxrwx   1 JOHNSON  SYS1          32 Jun 10 15:54 zceeopid -> /var/zosconnect/servers/zceeopid
lrwxrwxrwx   1 JOHNSON  SYS1          36 Aug 16 10:14 zceesrvr -> /var/ats/zosconnect/servers/zceesrvr
lrwxrwxrwx   1 JOHNSON  SYS1          44 Aug 16 11:57 zosmfServer -> /var/zosmf/configuration/servers/zosmfServer
```

Not all these directories are for z/OS Connect servers, there are CICS Liberty servers, a MQ Web Console Liberty server, a zOSMF Liberty server, a HATS Liberty server and a couple of standard Liberty servers for Java applications.

One administration directory to manage them all!

mitchj@us.ibm.com

Tech-Tip: Use dedicated ZFS filesystem at the mount points

- Create mount points in the “administrative” directory for shared r/w directories
- Avoid creating directories and files in the root file system.
- Use a common or shared mount point
  - Use /var mount point for local read/write file systems
  - Use /global for sharing a mount point across multiple LPARs
- Use ZFS filesystems and use AGGRGROW to allow R/W ZFS filesystems to automatically go into extents (>16).

```
SYS1.PARMLIB(BPXPRM##)
MOUNT FILESYSTEM('OMVS.LIBERTY.ZFS')
      MOUNTPOINT('/var/LibertyConfig')
      TYPE(ZFS) PARM('AGGRGROW') MODE(RDWR)
MOUNT FILESYSTEM('OMVS.ZCEEHCD.ZFS')
      MOUNTPOINT('/var/LibertyConfig/hcd')
      TYPE(ZFS) PARM('AGGRGROW') MODE(RDWR)
MOUNT FILESYSTEM('OMVS.ZCEE.SHARED.ZFS')
      MOUNTPOINT('/var/LibertyConfig/zosconnect')
      TYPE(ZFS) PARM('AGGRGROW') MODE(RDWR)
```

mitchj@us.ibm.com

Sharing XML configuration files between servers using ${shared.config.dir}

Add an "*includes"* subdirectories {shared.config.dir} with a symbolic links to a common location. This common directory an be accessed from multiple servers on a single or from multiple LPARs. Additions and updates to the "include" files are then made in one single administrative directory.

Symbolic links from servers shared configuration ${shared.config.dir} to common directory

Symbolic links to a shared local LPAR directory

```
ln -s /var/shared/zosconnect/includes /var/zosconnect/shared/config
ln -s /var/shared/zosconnect/includes /var/ats/zosconnect/shared/config
ln -s /var/shared/zosconnect/includes /var/wsc/zosconnect/shared/config
```

Symbolic links to a shared Sysplex directory *

```
ln -s /global/zosconnect/includes /var/zosconnect/servers/shared/config
ln -s /global/zosconnect/includes /var/ats/zosconnect/shared/config
ln -s /global/zosconnect/includes /var/wsc/zosconnect/shared/config
```

```
<include location="${shared.config.dir}/safSecurity.xml"/>
<include location="${shared.config.dir}/ipicIDProp.xml"/>
<include location="${shared.config.dir}/keyringOutboundMutual.xml"/>
<include location="${shared.config.dir}/groupAccess.xml"/>
<include location="${shared.config.dir}/shared.xml"/>
<include location="${shared.config.dir}/db2.xml"/>
<include location="${shared.config.dir}/oauth.xml"/>
```

/var/shared/*zosconnect/includes*

/global/zosconnect/*includes*

Contents of the common "includes" directory

```
basicSecurity.xml
db2.xml
db2TLS.xml
groupAccess.xml
ipic.xml
ipicIDProp.xml
keyringInbound.xml
keystore.xml
keyringMutual.xml
keyringOutboundMutual.xml
safSecurity.xml
```

The server.xml file contains these "include" statements and the server XML is portable regardless of the underlying filesystem infrastructure.

For example, changing *basicSecurity.xml* to *safSecurity.xml* and refreshing the configuration changes security from basic to SAF

F ZCEESRVR ,REFRESH,CONFIG

mitchj@us.ibm.com

*https://www.ibm.com/docs/en/zos/2.3.0?topic=zuapbfizv-zos-unix-add-global-directory-sysplex-root-file-system

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

July 2020	
V3.0.35 (APAR PH26291) Server code update	**Enhancements** • The text of messages BAQR0417W and BAQR0418W has been updated. For more information, see z/OS Connect EE Runtime Messages. **Fixes** • PH21761 A CICS region reports **`SOS DFHSM0133 WBSETBUF`** when z/OS Connect EE requester is in use. • PH25345 Passing user credentials in the request body to the authentication server to obtain a JWT causes a NPE in z/OS Connect EE. • PH21819 z/OS Connect EE sets some CORS headers automatically. **Attention** When this fix is applied, additional CORS configuration is required in `server.xml` to enable connections from the z/OS Connect EE API toolkit and JavaScript clients. For more information, see Configuring Cross-Origin Resource Sharing on a z/OS Connect Enterprise Edition Server

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}/cors.xml"/>
```

mitchj@us.ibm.com

Sharing XML configuration files – using '*variables*' files

"variables" files whose names are based on the name of the server

myServer.xml

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

zceeopid.xml

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

server.xml

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

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

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>
```

mitchj@us.ibm.com

Using the Liberty server's configuration drop-in's directories

Located in the same directory as the *server.xml* configuration file.

- Configuration files in the */overrides* directory adds to or replaces the configuration elements found in *server.xml*
- Configuration files in the */default* directory provides defaults for configuration elements not present in *server.xml*

```
${WLP_USER_DIR}
  /servers
    /serverName
      /apps
      /configDropins
         /overrides
         /default
      /logs
         /ffdc
         messages.log
      /resources
        /security
        /zosconnect
         /apis
         /apiRequesters
         /rules
         /services
       server.xml
      /tranlog
      /workarea
```

commonFeatures.xml

```
<server description="Common Server Features">

  <!-- Enable features -->
  <featureManager>
     <feature>adminCenter-1.0</feature>
     <feature>restConnector-2.0</feature>
  </featureManager>

  <remoteFileAccess>
     <readDir>/var/zcee/includes</readDir>
     <readDir>/global/zosconnect/includes</readDir>
     <writeDir>${server.config.dir}</writeDir>
  </remoteFileAccess>

</server>
```

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>
```

Another directory that must be manually created.

mitchj@us.ibm.com

Simplifying administration by combining include files and by using server variables

Default server.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<server description="new server">
<include location="${server.config.dir}/includes/${wlp.server.name}.xml"/>

    <!-- Enable features -->
    <featureManager>
        <feature>zosconnect:zosConnect-3.0</feature>
        <feature>openapi-3.0</feature>
    </featureManager>

    <!-- To access this server from a remote client add a host attribute
to the following element, e.g. host="*" -->
    <httpEndpoint id="defaultHttpEndpoint"
                  host="*"
                  httpPort="9080"
                  httpsPort="9443" />
```

${server.config.dir}/includes/${wlp.server.name}.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<server description="new server">
<variable name="httpPort" value="9081"/>
<variable name="httpsPort" value="9445"/>
<variable name="hostName" value="*"/>
<variable name="CICS_HOST" value="wg31.washington.ibm.com"/>
<variable name="CICS_PORT" value="1491"/>
<variable name="DB2_HOST" value="wg31.washington.ibm.com"/>
<variable name="DB2_PORT" value="2446"/>
<variable name="DB2_USERNAME" value="USER2"/>
<variable name="DB2_PASSWORD" value="USER2"/>
<include location="${shared.config.dir}/safSecurity.xml"/>
<include location="${shared.config.dir}/httpEndpoint.xml"/>
<include location="${shared.config.dir}/db2.xml"/>
<include location="${shared.config.dir}/cics.xml"/>
<include location="${shared.config.dir}/keystore.xml"/>
</server>
```

${server.config.dir}/includes/httpEndpoint.xml”/>

```
<server description="basic security">
   <httpEndpoint id="defaultHttpEndpoint"
                 host="${hostName}"
                 httpPort="${httpPort}"
                 httpsPort="${httpsPort}" />
</server>
```

${server.config.dir}/includes/db2.xml”/>

```
<?xml version="1.0" encoding="UTF-8"?>
<server description="Default server">
    <featureManager>
        <feature>zosconnect:db2-1.0</feature>
    </featureManager>
 <zosconnect_credential user="${DB2_USERNAME}"
       password="${DB2_PASSWORD}" id="commonCredentials" />
 <zosconnect_db2Connection id="db2Conn" host="${DB2_HOST}"
      port="${DB2_PORT}" credentialRef="commonCredentials" />
</server>
```

${server.config.dir}/includes/cics.xml”/>

```
<server description="CICS IPIC connections">
<!-- Enable features -->
<featureManager>
  <feature>zosconnect:cics-1.0</feature>
</featureManager>
 <zosconnect_cicsIpicConnection id="cicsConn" host="${CICS_HOST}"
    port="${CICS_PORT}" />
 </server>
```

mitchj@us.ibm.com

Use Symbolic links to simplify commands used in OMVS and JCL

```
Performing commands:
ln -s /global/zosconnect/includes /var/zcee/includes
ln -s /var/zosconnect/servers/zceesrv1 /var/zcee/zceesrv1
ln -s /var/zosconnect/servers/zceesrv2 /var/zcee/zceesrv2

Will change these OMVS commands from:
ln -s /global/zosconnect/includes /var/zosconnect/servers/zceesrv1/includes
ln -s /global/zosconnect/includes /var/zosconnect/servers/zceesrv2/includes

To simpler (and shorter) OMVS commands:
ln -s /var/zcee/includes /var/zcee/zceesrv1/includes
ln -s /var/zcee/includes /var/zcee/zceesrv2/includes
```

Directory Shortcuts

- Create a shortcut from the shared administrative *include* directory to the Sysplex or LPAR shared directory
- Create shortcuts from the server's administrative directories to each server's configuration directory.

N.B. These are symbolic links to symbolic links.

ln –s oldname newname

These symbolic links can be used as JCL symbols

```
//EXPORT EXPORT SYMLIST=(*)
// SET SERVER= 'zceesrv1'
// SET SHARED='/var/zcee/includes'
// SET WLPUSER='/var/zosconnect'
//ZCEELN EXEC PGM=IKJEFT01,REGION=0M
//SYSTSPRT DD SYSOUT=*
//SYSERR   DD SYSOUT=*
//STDOUT   DD SYSOUT=*
//SYSTSIN  DD *,SYMBOLS=EXECSYS
  BPXBATCH SH +
  ln -s &SHARED /var/zcee/&SERVER/includes
    instead of entering the full directory names as in
  ln -s /global/zosconnect/includes  +
          &WLPUSER/servers/&SERVER/includes
```

And added as exports to /u/*home*/.profile or /etc/profile files

```
export serverName=zceesrv1
export shared=/var/zcee/includes
export WLP_USER_DIR=/var/zosconnect
```

```
//ZCEELN EXEC PGM=IKJEFT01,REGION=0M
//SYSTSPRT DD SYSOUT=*
//SYSERR   DD SYSOUT=*
//STDOUT   DD SYSOUT=*
//SYSTSIN  DD *
  BPXBATCH SH +
  ln -s $shared /var/zcee/$serverName/includes
instead of entering the full directory names as in
  ln -s /global/zosconnect/includes +
          $WLPUSER/servers/$serverName/includes
```

mitchj@us.ibm.com

Finally, use JCL to make the creation and configuration of servers repeatable and portable

```
//*************************************************************
//*  SET SYMBOLS
//*************************************************************
//EXPORT EXPORT SYMLIST=(*)
// SET JAVAHOME='/usr/lpp/java/J8.0_64'
// SET ZCEEPATH='/usr/lpp/IBM/zosconnect/v3r0'
// SET SERVER=`zceeSrvr'
// SET TEMPLATE='zosconnect:default'
// SET WLPUSER='/var/ats/zosconnect'
// SET USER='ATSSERV'
// SET GROUP='ATSGRP'
//*************************************************************
//*  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; +
 ln -s $WLP_USER_DIR/servers/&SERVER /var/zcee/&SERVER; +
 ln -s /var/shared/includes/commonFeatures.xml +
    /var/zcee/&SERVER/&CONFIG/commonFeatures.xml; +
 ln -s /var/shared/includes/cors.xml +
    /var/zcee/&SERVER/&CONFIG/cors.xml; +
 ln -s /var/shared/includes/safSecurity.xml +
    /var/zcee/&SERVER/&CONFIG/safSecurity.xml; +
 cp /var/zcee/properties/bootstrap.properties +
    /var/zcee/&SERVER; +
 cp /var/zcee/properties/server.xml +
    /var/zcee/&SERVER; +
 ln -s /var/shared/includes +
    /var/zcee/&SERVER/includes; +
 chown -R &USER:&GROUP $WLP_USER_DIR/servers/&SERVER
```

```
//*************************************************************
//*  SET SYMBOLS
//*************************************************************
//EXPORT EXPORT SYMLIST=(*)
// SET JAVAHOME='/usr/lpp/java/J8.0_64'
// SET ZCEEPATH='/usr/lpp/IBM/zosconnect/v3r0'
// SET SERVER='openApi3'
// SET TEMPLATE='zosconnect:openApi3'
// SET WLPUSER='/var/ats/zosconnect'
// SET CONFIG='configDropins/overrides'
// SET USER='ATSSERV'
// SET GROUP='ATSGRP'
//*************************************************************
//*  Step ZCEEAPI3 - Use the zosconnect command to create a server
//*************************************************************
//ZCEEAPI3 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; +
 ln -s $WLP_USER_DIR/servers/&SERVER /var/zcee/&SERVER; +
 ln -s /var/shared/includes/commonFeatures.xml +
    /var/zcee/&SERVER/&CONFIG/commonFeatures.xml; +
 ln -s /var/shared/includes/safSecurity.xml +
    /var/zcee/&SERVER/&CONFIG/safSecurity.xml; +
 cp /var/zcee/properties/bootstrap.properties +
    /var/zcee/&SERVER; +
 cp /var/zcee/properties/server.xml +
    /var/zcee/&SERVER; +
 ln -s /var/shared/includes +
    /var/zcee/&SERVER/includes; +
 chown -R &USER:&GROUP $WLP_USER_DIR/servers/&SERVER
```

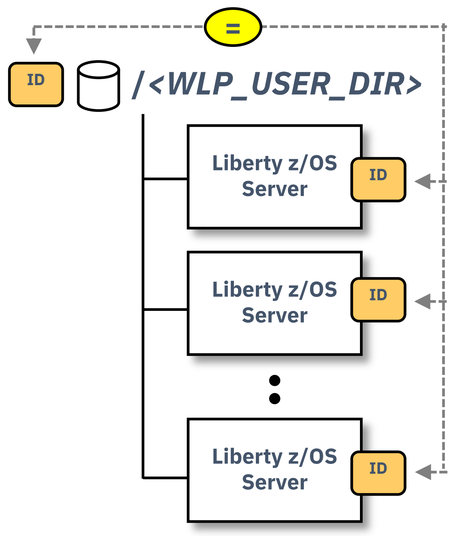
mitchj@us.ibm.com

RACF, Liberty and z/OS Connect Security Details and Options

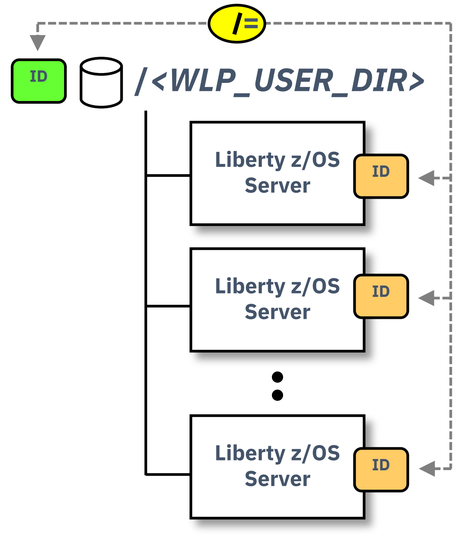
mitchj@us.ibm.com

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

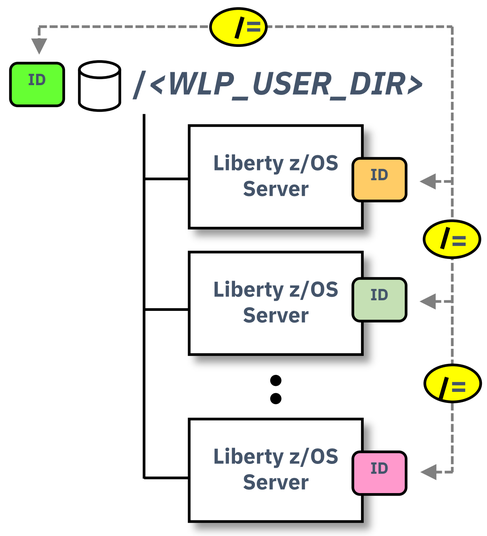
On z/OS, the best practice for Liberty servers in production is that they run as ‘Started Tasks’ (STCs).



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



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



- Multiple servers
- Different STC IDs
- STC IDs ǂ File Owner ID

Should all servers sharing WLP_USER_DIR share the same STC ID?
It is a matte of the degree of identity isolation that is required

mitchj@us.ibm.com

z/OS Security: Assigning ID to started tasks: SAF STARTED class

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

	Common Identity per task	*Unique Identities per task*

Common JCL Procedure	`RDEFINE STARTED ZCEEPROC.*`
	
	`S ZCEEPROC,JOBNAME=server1,PARMS='server1'`
	`S ZCEEPROC,JOBNAME=server2,PARMS='server2'`
	(1)
		`RDEFINE STARTED ZCEEPROC.server1`
		`RDEFINE STARTED ZCEEPROC.server2`
		
		`S ZCEEPROC,JOBNAME=server1,PARMS='server1'`
		`S ZCEEPROC,JOBNAME=server2,PARMS='server2'`
		(3)

Unique JCL Procedure per server	`RDEFINE STARTED ZCEE*.*`
	
	`S ZCEESRV1,JOBNAME=server1,PARMS='server1'`
	`S ZCEESRV2,JOBNAME=server2,PARMS='server2'`
	(2)
		`RDEFINE STARTED ZCEESRV1.*`
		`RDEFINE STARTED ZCEESRV2.*`
		
		`S ZCEESRV1,JOBNAME=server1,PARMS='server1'`
		`S ZCEESRV2,JOBNAME=server2,PARMS='server2'`
		(4)

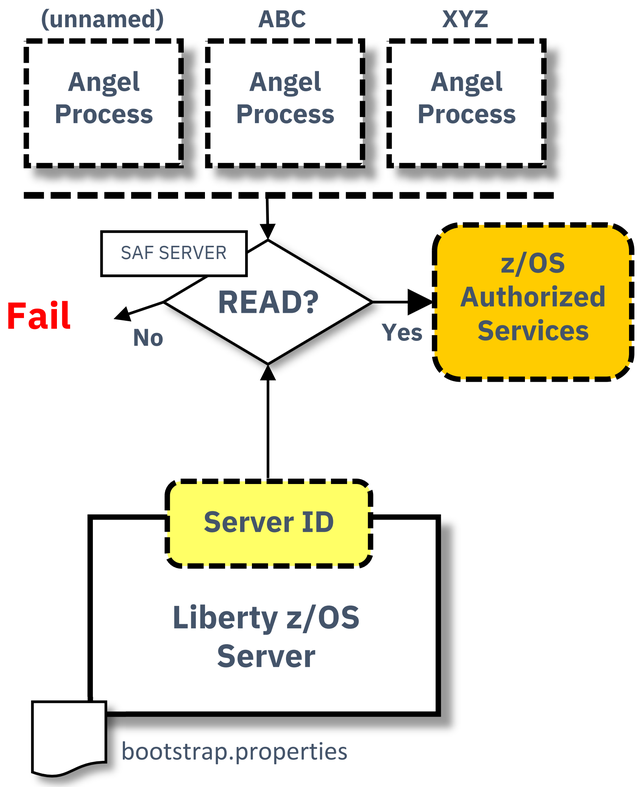
Note: Using unique JCL procedure eliminates the need to specify PARMS on the start commands

1. The same identity is used for all servers using a RACF STARTED class resource where the JCL procedure is discrete, and the job name is generic.
2. The same identity is used for all servers using a RACF STARTED class resource where both the JCL procedure and job names are generic
3. Different identities are used for each server using a RACF STARTED class resource where both the JCL procedure and job name are discrete.
4. Different identities are used for each server using a RACF STARTED class resource where the JCL procedure is discrete and job name is generic.

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.

mitchj@us.ibm.com

z/OS Security: 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**

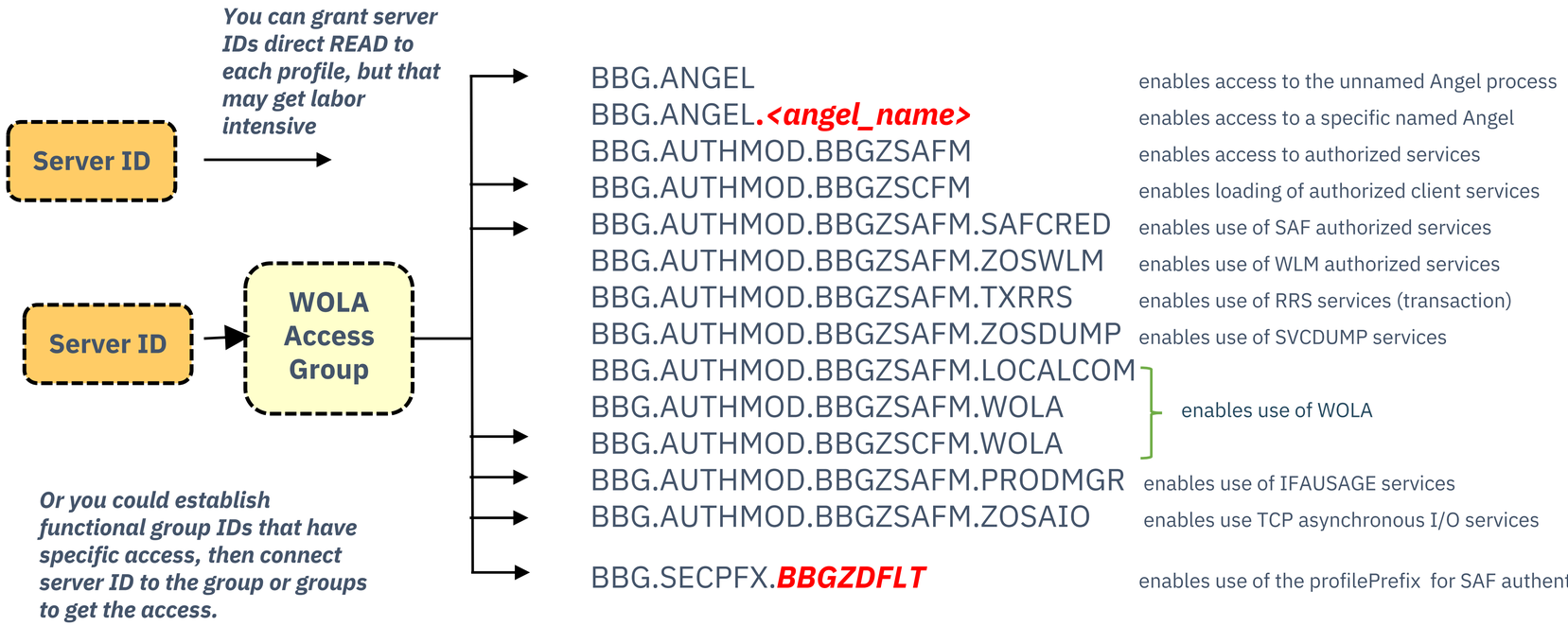
```
com.ibm.ws.zos.core.angelRequired=true
com.ibm.ws.zos.core.angelName=<name>
com.ibm.ws.zos.core.angelRequiredServices=SAFCRED,ZOSWLM,TXRRS,ZOSDUMP
```

List of current Liberty Features

https://www.ibm.com/support/knowledgecenter/SSEQTP_liberty/com.ibm.websphere.wlp.doc/ae/rwlp_feat.html

mitchj@us.ibm.com

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



Best practice:

- **Establish all the SERVER profiles ahead of time. Existence of profile does not grant access; READ access 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 service. 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
Liberty 21.0.6 add a new property to identity required services, com.ibm.ws.zos.core.angelRequiredServices, for more details see URL
https://www.ibm.com/docs/en/was-liberty/zos?topic=overview-process-types-zos

mitchj@us.ibm.com

z/OS Authorized service security: Angel Required Services

To use z/OS authorized services, you must have a Liberty Angel process and grant access for your Liberty server's SAF identity to use these services.

- LOCALCOM - Required to use *WebSphere Optimized Local Adapters* (WOLA).
- PRODMGR – Required to use IFAUSAGE services for SMF reporting.
- SAFCRED - Required to use SAF authorized user registry services and SAF authorization services.
- TXRRS - Required by the IBM® MQ resource adapter when the connection to IBM MQ is made in BINDINGS mode
- WOLA - Required to use *WebSphere Optimized Local Adapters* (WOLA).
- ZOSAIO - Required to use AsyncIO on z/OS.
- ZOSDUMP - Only required if asked to obtain an SVC dump by IBM service. It provides access to SVCDUMP services.
- ZOSWLM - Required to use WLM services. For more information, see Measuring API workloads with WLM at URL https://www.ibm.com/docs/en/zos-connect/zosconnect/3.0?topic=considerations-measuring-api-workloads-wlm

When a Liberty server connects to an angel process during server startup, it checks that the server's identity has access to the z/OS authorized services. By default, access checks are performed for all authorized services.
You can restrict the Liberty server to check and use only the authorized services it requires, which then makes other authorized services unavailable by using property, **com.ibm.ws.zos.core.angelRequiredServices**
The value for this property, **com.ibm.ws.zos.core.angelRequiredServices,** must be a comma-separated list of valid angel process services, as described above. This property must be specified with the **com.ibm.ws.zos.core.angelRequired** property set to **true**. Only these services, when properly specified, are the ones used by the server. **Lack of access to the angel process itself or any of these listed required services will cause a server startup failure.**

mitchj@us.ibm.com

Example: using the *bootstrap.properties* file to set required z/OS privileges

zceesrv1's bootstrap.properties

```
httpPort=9080
httpsPort=9443
ipicPort=1491
host=*
cicsHost=wg31.washington.ibm.com
network=ZOSCONN1
applid=ZOSCONN1
com.ibm.ws.zos.core.angelName=namedAngel
com.ibm.ws.zos.core.angelRequired=true
com.ibm.ws.zos.core.angelRequiredServices=SAFCRED,ZOSWLM,PRODMGR,ZOSAIO,TXRRS,LOCALCOM
```

zceesrv2's bootstrap.properties

```
httpPort=9090
httpsPort=9453
ipicPort=1492
host=wg31.washington.ibm.com
cicsHost=wg31.washington.ibm.com
network=ZOSCONN2
applid=ZOSCONN2
com.ibm.ws.zos.core.angelName=namedAngel
com.ibm.ws.zos.core.angelRequired=true
com.ibm.ws.zos.core.angelRequiredServices=SAFCRED,ZOSWLM,PRODMGR,ZOSAIO,TXRRS,LOCALCOM
```

mitchj@us.ibm.com

Tech-Tip: 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"/>
<safAuthorization racRouteLog=ASIS reportAuthorizationCheckDetails="true"/>
```

https://www.ibm.com/support/knowledgecenter/SS7K4U_liberty/com.ibm.websphere.wlp.zseries.doc/ae/twlp_config_security_saf.html
hhttps://www.ibm.com/docs/en/zos-connect/zosconnect/3.0?topic=registry-saf-unauthenticated-user-id

mitchj@us.ibm.com

z/OS Connect Security

Overview

mitchj@us.ibm.com

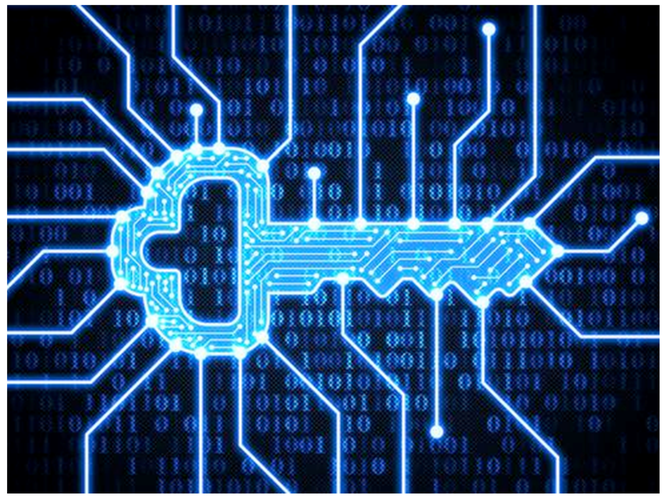
General security terms or considerations

Security involves

- Identifying who or what is requesting access (**Authentication)**
  - Basic Authentication
  - Mutual Authentication using Transport Layer Security (TLS), formerly known as SSL
  - Third Party Tokens

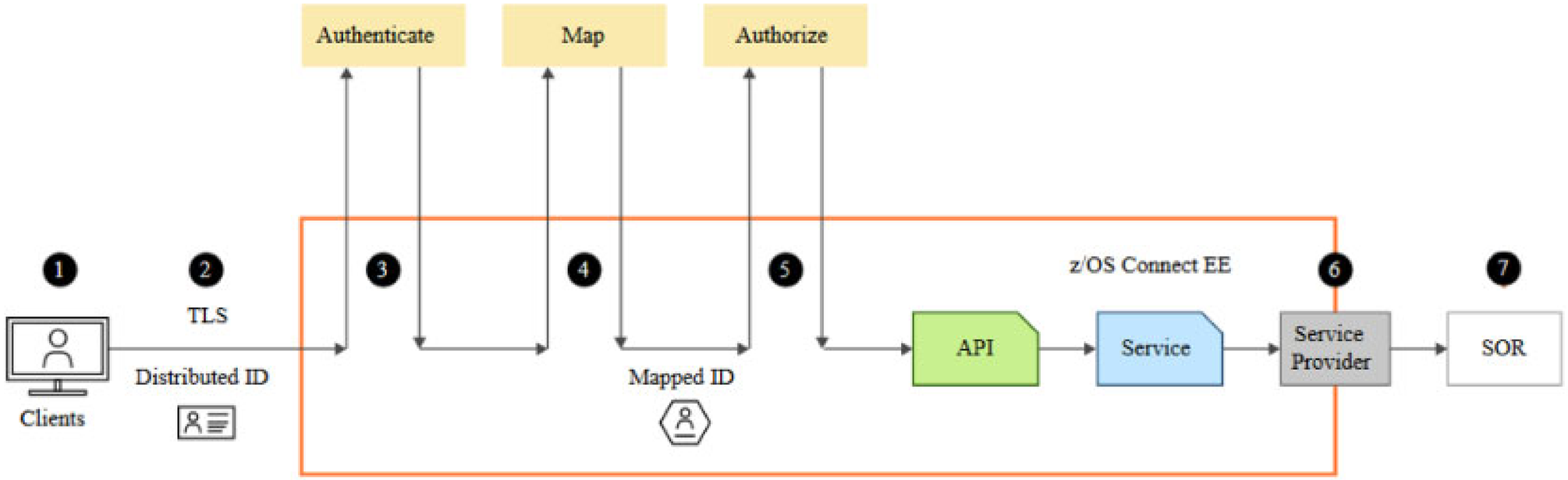
- Ensuring that the message has not been altered in transit (**Data Integrity)** and ensuring the confidentiality of the message in transit (**Encryption)**
  - TLS (encrypting messages and using a digital signature)

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



mitchj@us.ibm.com

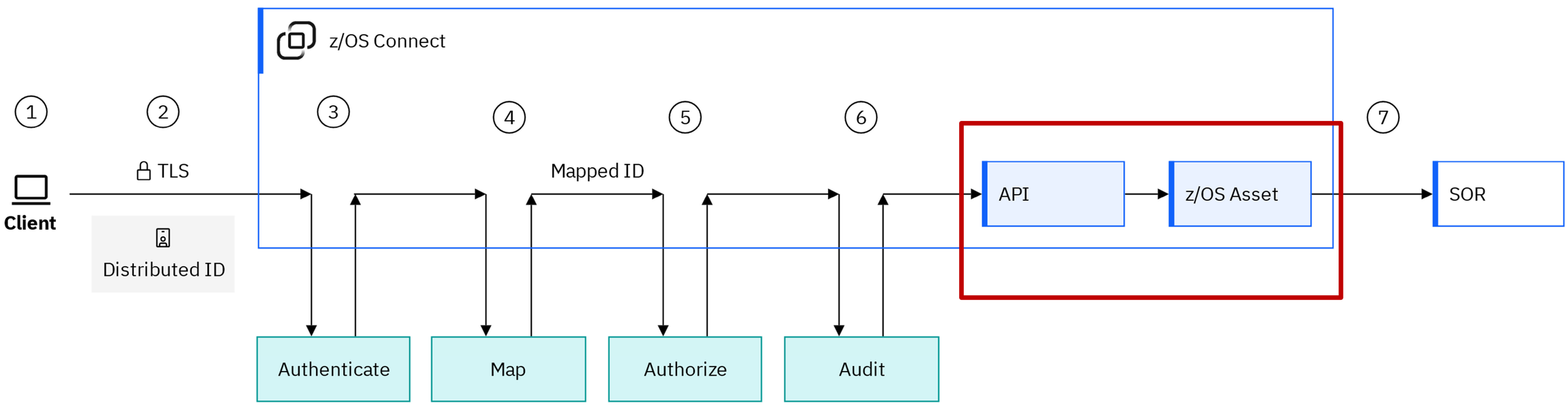
A typical z/OS Connect API Provider inbound security - inbound (OpenAPI 2)



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

mitchj@us.ibm.com

Details of a typical z/OS Connect EE API Provider security flow - inbound (OpenAPI 3)



The flow includes the following security steps that can be performed by IBM z/OS Connect. The credentials are provided by the client. These can be a user ID and password, a JWT, or a TLS certificate.

1. The credentials, including the identity, are passed on the connection between the client and IBM z/OS Connect. The identity is typically a distributed ID, such as an X.509 distinguished name and associated LDAP realm that originates from a remote system. Alternatively, the identity might be a SAF user ID. The data that is sent on the connection can be encrypted using TLS.
2. The client is authenticated. This can be within IBM z/OS Connect or by requesting verification from a third-party server.
3. The authenticated identity can be mapped to a user ID in the IBM z/OS Connect user registry.
4. The user is authorized to invoke the API operation if they have the required role.
5. The API request is audited by using the Liberty Audit feature.
6. The authenticated user identity can be propagated to the System of Record (SoR) when the SoR supports this capability. Alternatively, the SoR connection can be configured to use a functional identity.

mitchj@us.ibm.com

Let’s explore the security options for inbound API Provider connections and accessing z/OS resources

mitchj@us.ibm.com

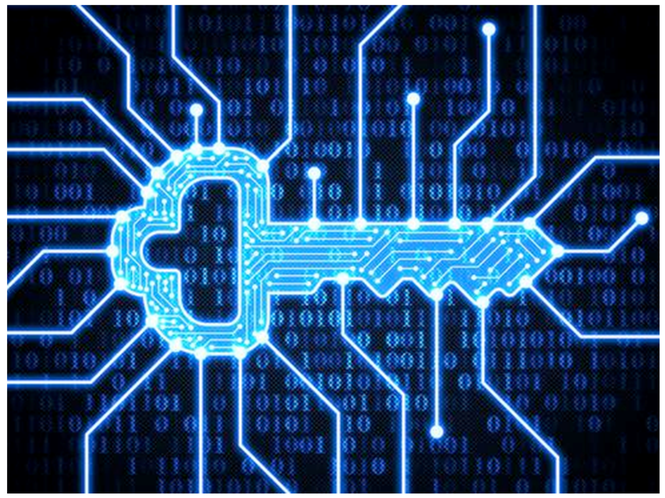
General security terms or considerations

Security involves

- Identifying who or what is requesting access (**Authentication)**
  - Basic Authentication
  - Mutual Authentication using Transport Layer Security (TLS), formerly known as SSL
  - Third Party Tokens

- Ensuring that the message has not been altered in transit (**Data Integrity)** and ensuring the confidentiality of the message in transit (**Encryption)**
  - TLS (encrypting messages and using a digital signature)

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

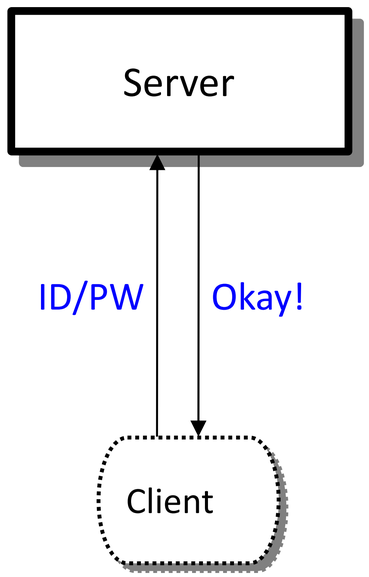


mitchj@us.ibm.com

Liberty Authentication Options

Several different ways this can be accomplished:

Basic Authentication

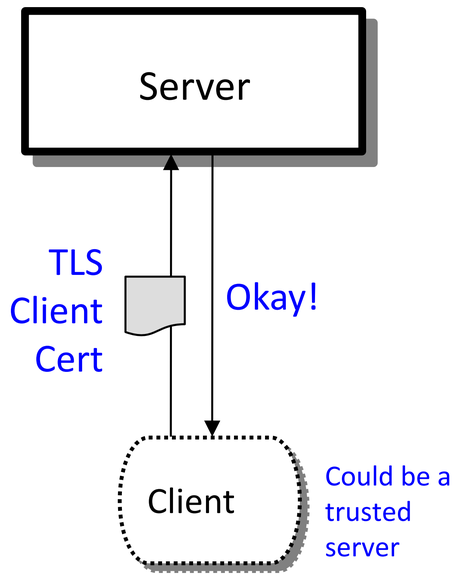


Client supplies ID/PW or ID/PassTicket

Server checks registry:

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

Client Certificate



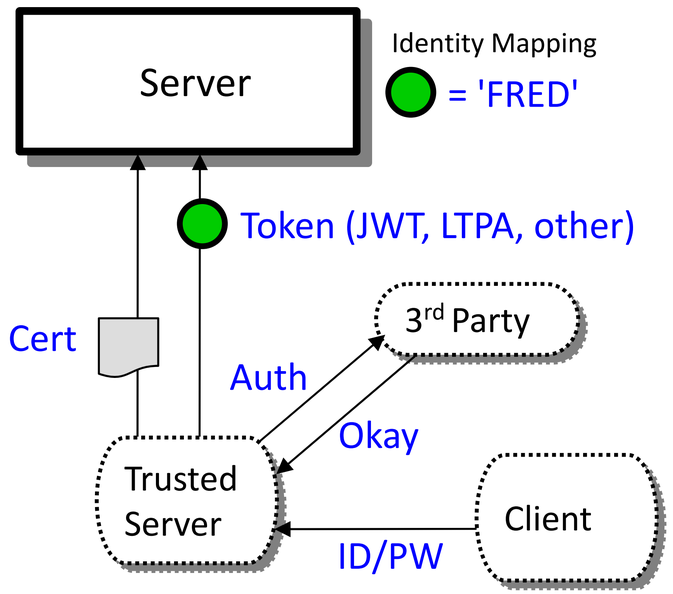
Client supplies client personal certificate

Server validates client personal certificate and maps it to an identity

Registry options:

- **SAF**

Third Party Authentication



Client authenticates to 3rd party sever

Client receives a trusted 3rd party token

Token flows to server and is mapped to an identity

Registry options:

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

mitchj@us.ibm.com

z/OS Connect Security server XML Authentication Configuration (OpenAPI 2)

- requireAuth - requires the client to provide credentials

```
<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>
```

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.

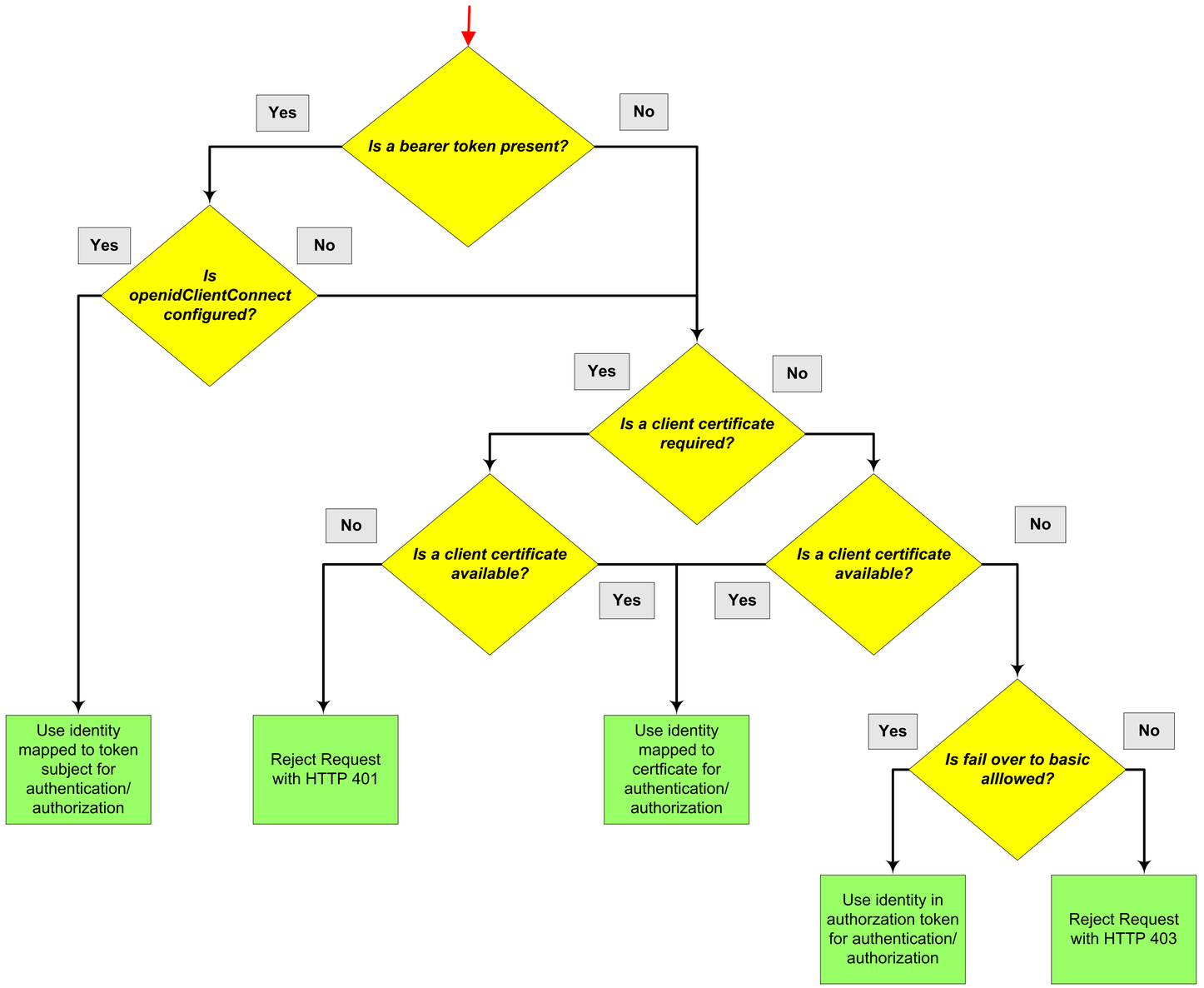
Requires that users specify security credentials to be authenticated in order to access all API requesters. If the requireAuth attribute is not set, the global setting on the zosconnect_zosConnectManager element is used instead, unless the requireAuth attribute is overridden on the specific API requester.

The requireAuth attribute controls whether an inbound request must provide credentials using one of the three authentication methods, e.g., basic, client certificate, or third-party token.

Note that there are no equivalent configuration elements for an OpenAPI 3 server.

mitchj@us.ibm.com

Authentication credential precedence order for determining authorization identity

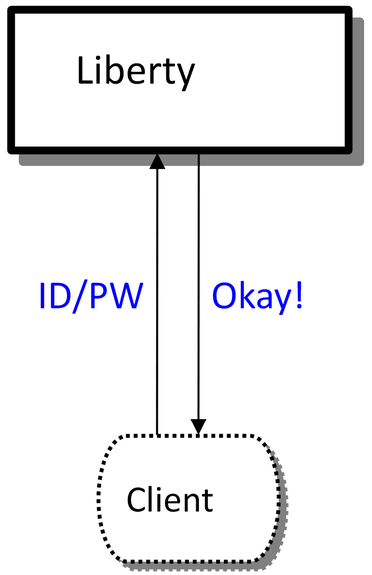


mitchj@us.ibm.com

Authentication - Basic Authentication

Several different ways this can be accomplished:

Basic Authentication

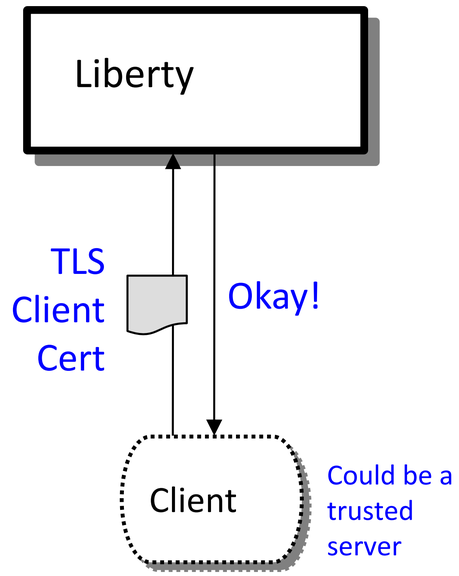


Client supplies an identity/password or an identity/PassTicket

Server checks registry:

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

Client Certificate



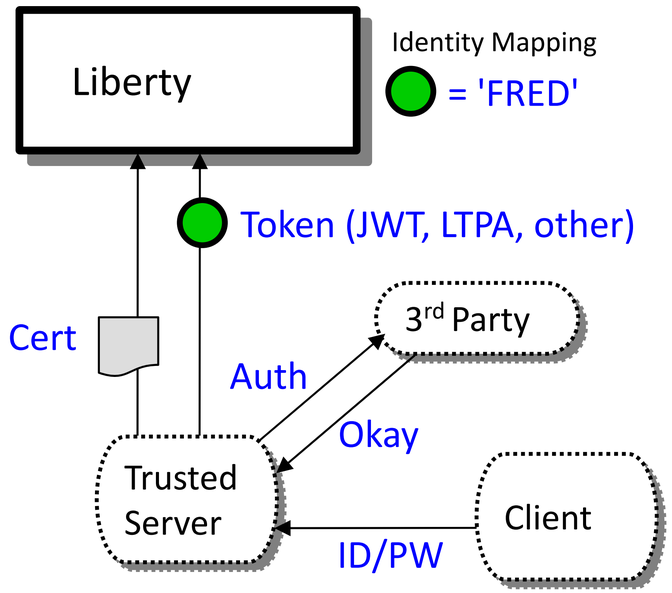
Client supplies client certificate

Server validates client certificate and maps it to an identity

Registry options:

- SAF

Third Party Authentication



Client authenticates to 3rd party sever

Client receives a trusted 3rd party token

Token flows to server and is mapped to an identity

Registry options:

- We do not need to know the details.

mitchj@us.ibm.com

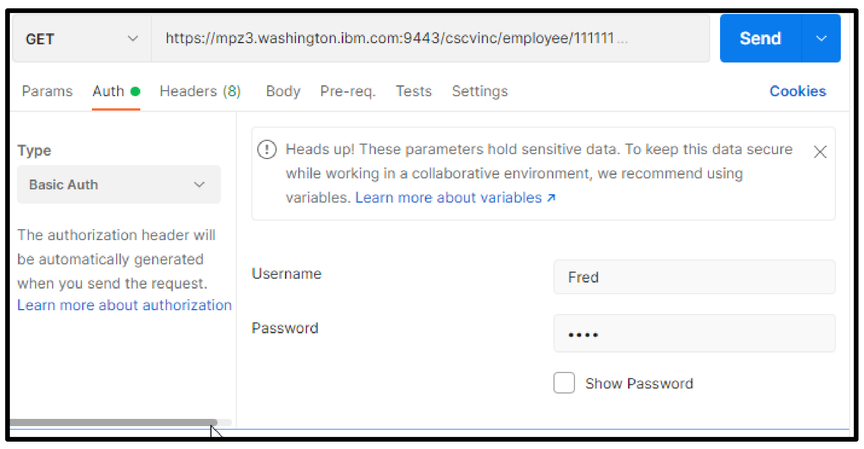
Basic authentication – Where the client provides an identity and password

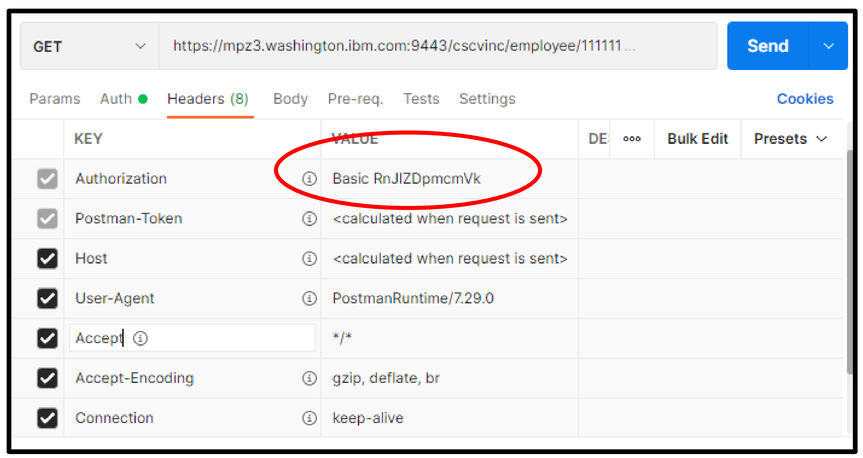
- server XML security configuration:

```
<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, basic authentication information (identity and password) is provided in the HTTP header in a Basic *Authorization t*oken with the identity and password encoded or formatted using Base64.
  - An example with Postman:

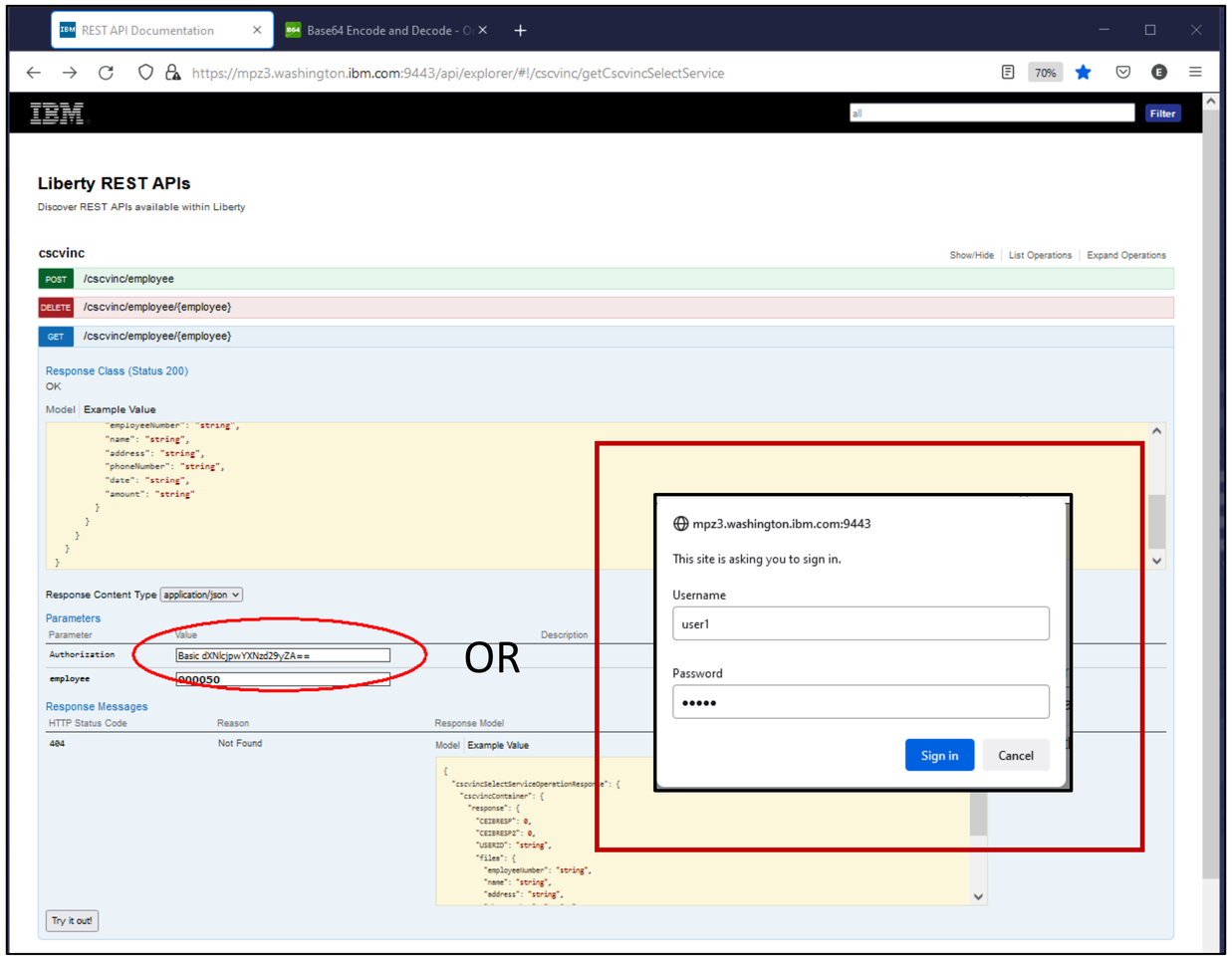


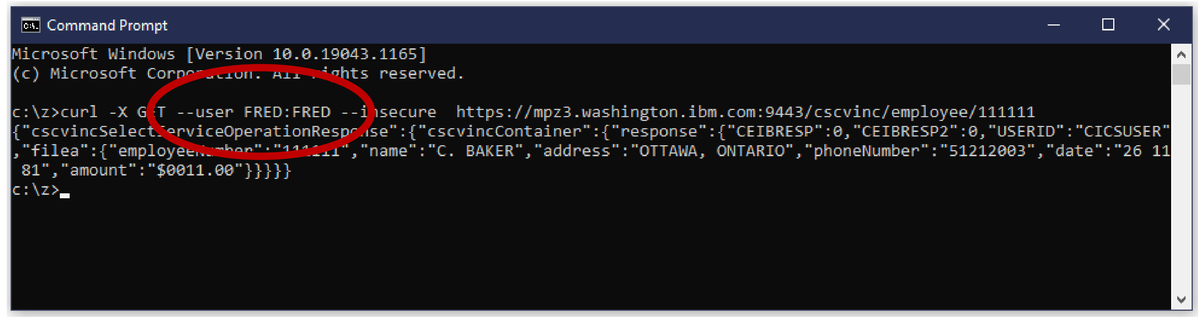


mitchj@us.ibm.com

There are multiple ways to provide an identity and password

- 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 t*oken with the identity and password encoded or formatted using Base64.
  - Examples using the API Explorer feature , cURL, and a Java client.





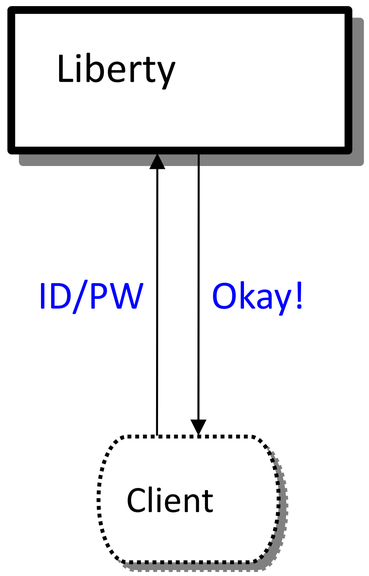
https://www.base64encode.org/

mitchj@us.ibm.com

Authentication - TLS Mutual Authentication

Several different ways this can be accomplished:

Basic Authentication

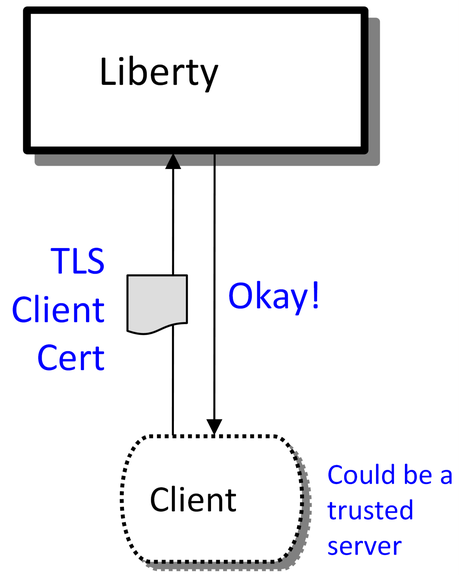


Server prompts for ID/PW

Client supplies ID/PW or ID/PassTicket

Server checks registry:
- Basic (server.xml)
- SAF

Client Certificate



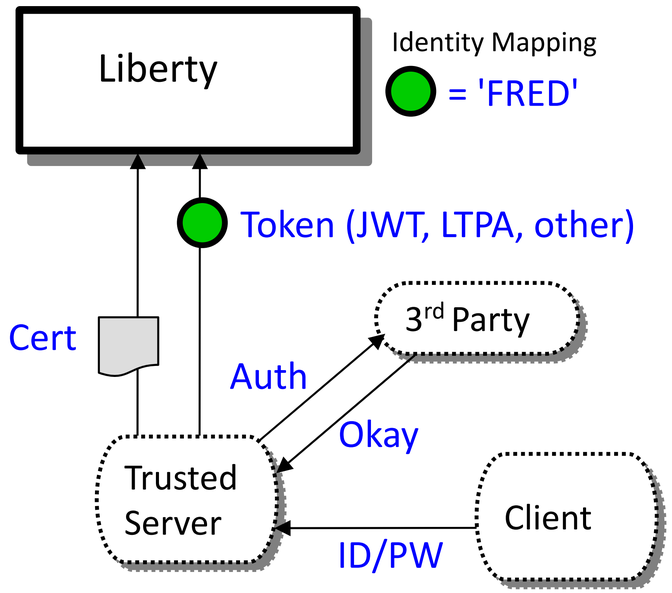
Server prompts for client certificate.

Client supplies personal certificate

Server validates client certificate and maps it to an identity

Registry options:
- **SAF**

Third Party Authentication



Client authenticates to 3rd party sever

Client receives a trusted 3rd party token

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

Registry options:
- We may not need to know these details.

mitchj@us.ibm.com

Liberty JSSE (HTTPS) 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 send to for clients

Key ring for client connections to server endpoints

OpenAPI 2

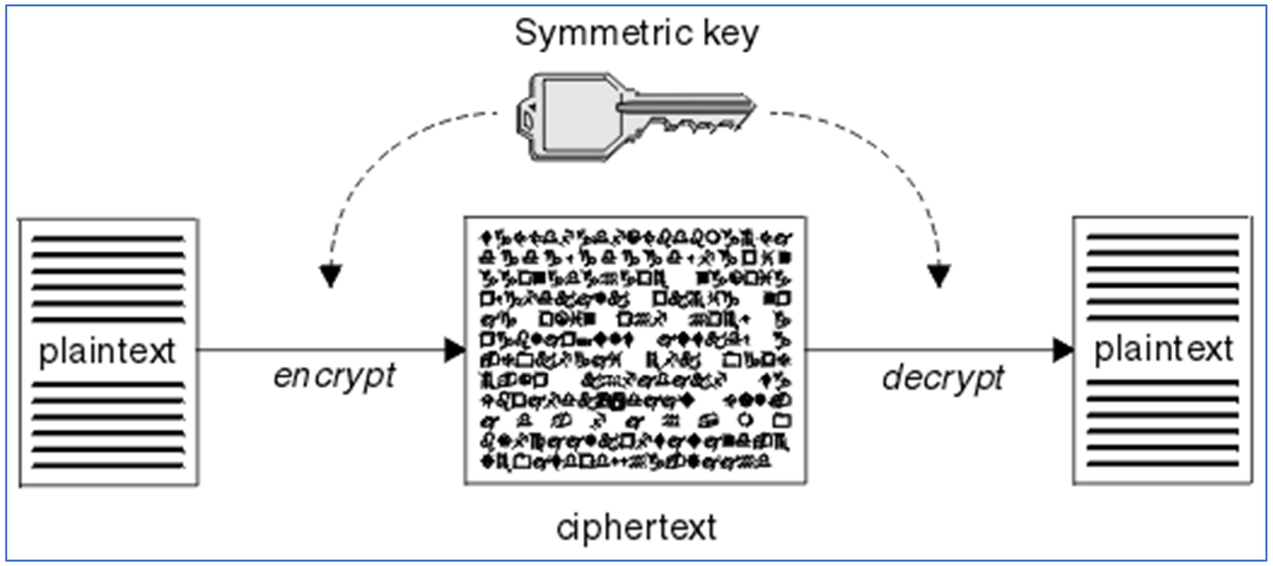
```
<zosconnect_zosConnectManager
            requireAuth="true"
            requireSecure="true|false"/>

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

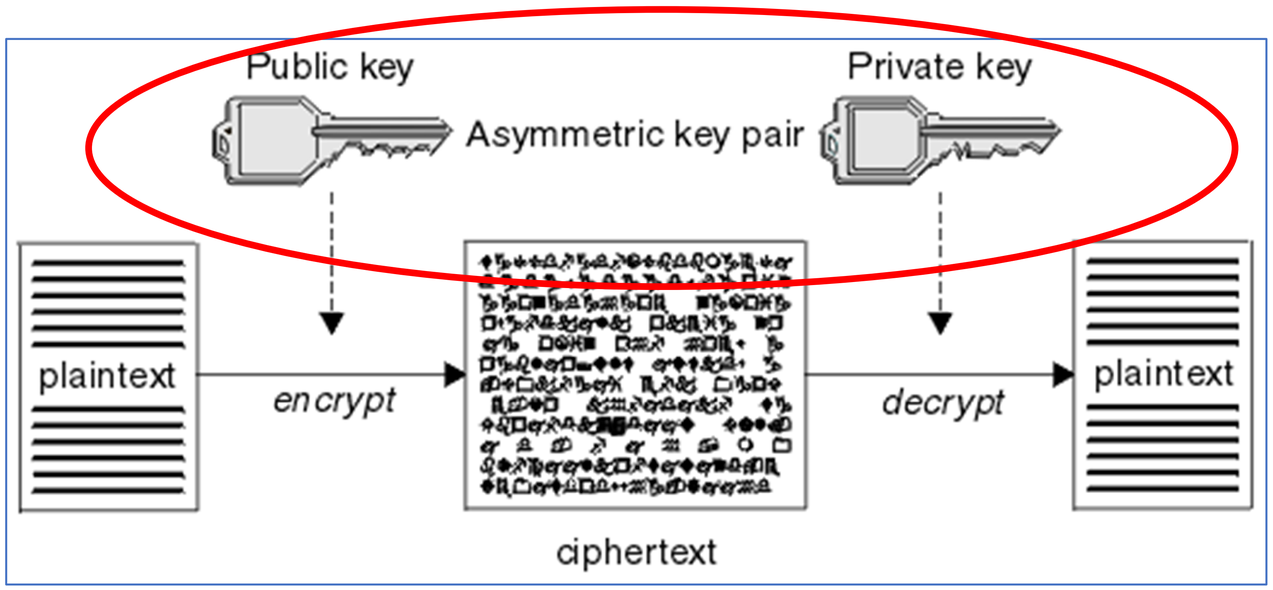
safkeyring:///**KeyRing** v safkeyring:**//owner/KeyRing**

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.

mitchj@us.ibm.com

Tech-Tip: Symmetric key v. Asymmetric key pairs





A symmetric key is a key shared by the endpoints. Both endpoints use the same key to encrypt and decrypt messages.

An asymmetric key pair is the preferred solution. There is no risk of compromise by sending a symmetric or shared key outside of a protected communication flow.

A message encrypted with a public key can only be decrypted by endpoint that has the private key. The privacy of the messages is ensured.

If an endpoint can successfully decrypt a message message encrypted received with a private key, the endpoint sending the message has successfully asserted its validity by proving it has the private used to encrypt the message.

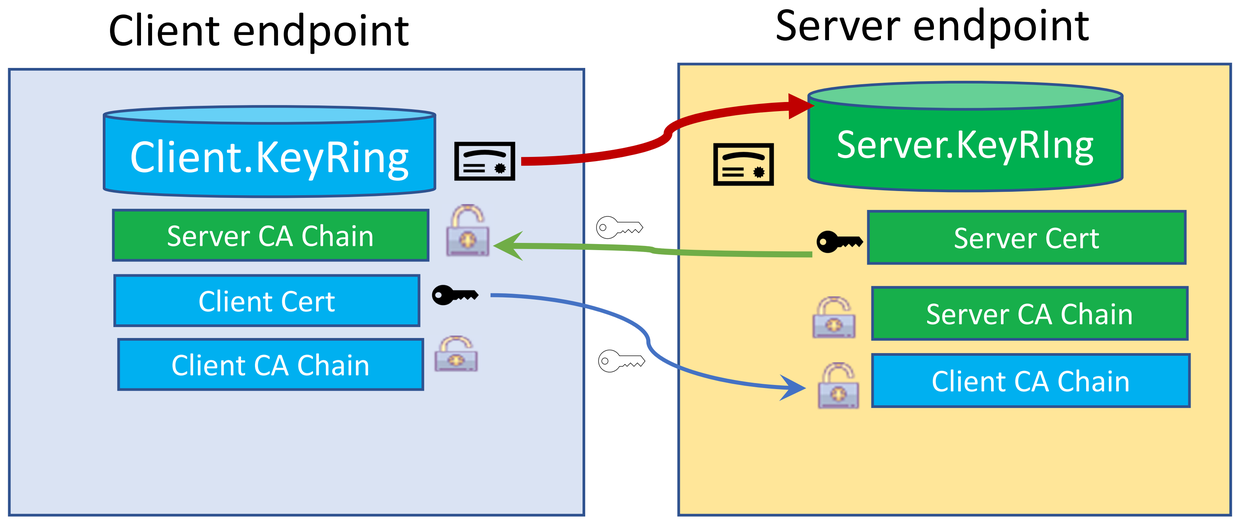
mitchj@us.ibm.com

The basic TLS Handshake Flow (HTTPS)

The HTTPS protocol involves a TLS handshake –

Server Authentication (always occurs when HTTPS is the protocol)

Mutual Authentication (optional, at the request of the server endpoint)



Certificate with a private key*

Certificate and public key

Certificate Authority (CA) certificate chain#

*For server and/or mutual authentication to work, the endpoint sending the 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 up to and including the root CA.

mitchj@us.ibm.com

Tech-Tip: Key rings(non-virtual) and accessing a certificate's private keys

Two types of RACF profiles resources are used to control access to key ring and certificates

- **RDATALIB** for controlling access to a specific key ring
- **FACILITY** for controlling access to key rings globally

User certificates (connected to the key ring with usage PERSONAL)

- **Global profiles uses the FACILITY** resource **IRR.DIGTCERT.LISTRING**:
  - **READ** access is required to access one's own key ring and private key
  - **UPDATE** access is required to access another user's key private key

- **Specific key rings uses the** RDATALIB class **<ring owner>.<ring name>.LST**
  - **READ** access is required to access one's own private key
  - **UPDATE** access is required to access another identity's private keys

CERTAUTH and SITE certificates (connected to the key ring with usage PERSONAL)

- **Global profiles uses the FACILITY** resource **IRR.DIGTCERT.GENCERT**:
  - **CONTROL** access is required to access a CERTAUTH or SITE certificate private key ring

- **Specific key rings uses the** RDATALIB class **<ring owner>.<ring name>.LST**
  - **CONTROL** access is required to access the private keys of CERTAUTH and SITE certificates

Remember: When switching from global FACILITY class profiles to specific ring RDATALIB class profiles, the RDATLIB resources will be checked first

mitchj@us.ibm.com

Tech/Tip: Details of the flow with mutual authentication (TLS 1.2)

1. A Client sends a request to server for a protected session in a ***ClientHello*** message. Included in the request is the TLS capabilities of the client (e.g., TLS 1.2 or 1.3) and a list of supported ciphers in preference order.
2. The server selects the TLS version and selects cipher from the list sent by the client and returns this information in a ***ServerHello*** message.
3. The server's certificate public information (including the **public key**) is sent to the client in a ***Certificate*** message.
4. The server sends cryptographic information for the client to use for encrypting a pre-master key in a **Server key exchange** message.
5. **For mutual authentication, the server sends a *CertificateRequest* message requesting a client's personal certificate.**
6. The server concludes by sending a ***ServerHelloDone*** message.
7. The client verifies the server's certificate with its trust store.
8. **If mutual authentication is requested, the client sends its public personal certificate information in a *Certificate* message**
9. The client then uses the ***server's public* key** to generate and encrypt a 48 byte "premaster secret" message which is sent to the server in a ***ClientKeyExchange*** message.
10. **When mutual authentication is requested, a digitally signature (hashed) of the concatenation of all previous handshake messages is encrypted with the client's private key sent in a *CertificateVerify* message.**
11. The ***Change Cipher*** message is used to change the from cipher used during the handshake so all subsequent messages will be encrypted using a different cipher.
12. The server uses its **private key** to decrypt the "premaster secret" message (**only the private key can be used to decrypt the message**).
13. **If mutual authentication is requested, the server verifies the client's personal certificate with its key ring and uses the client's public key to decrypt and verify the message sent in the *CertificateVerify* message.**
14. Both the Client and Server use the "premaster secret" to compute a 'master secret', also know as "shared secret" or "session key" (symmetric encryption)
15. Client and server will use this "shared secret" or "session key" to encrypts messages sent between the endpoints.

mitchj@us.ibm.com

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 *IBM SDK Policy files* topic in the *Security Reference for IBM SDK, Java Technology Edition* for the version you are using.

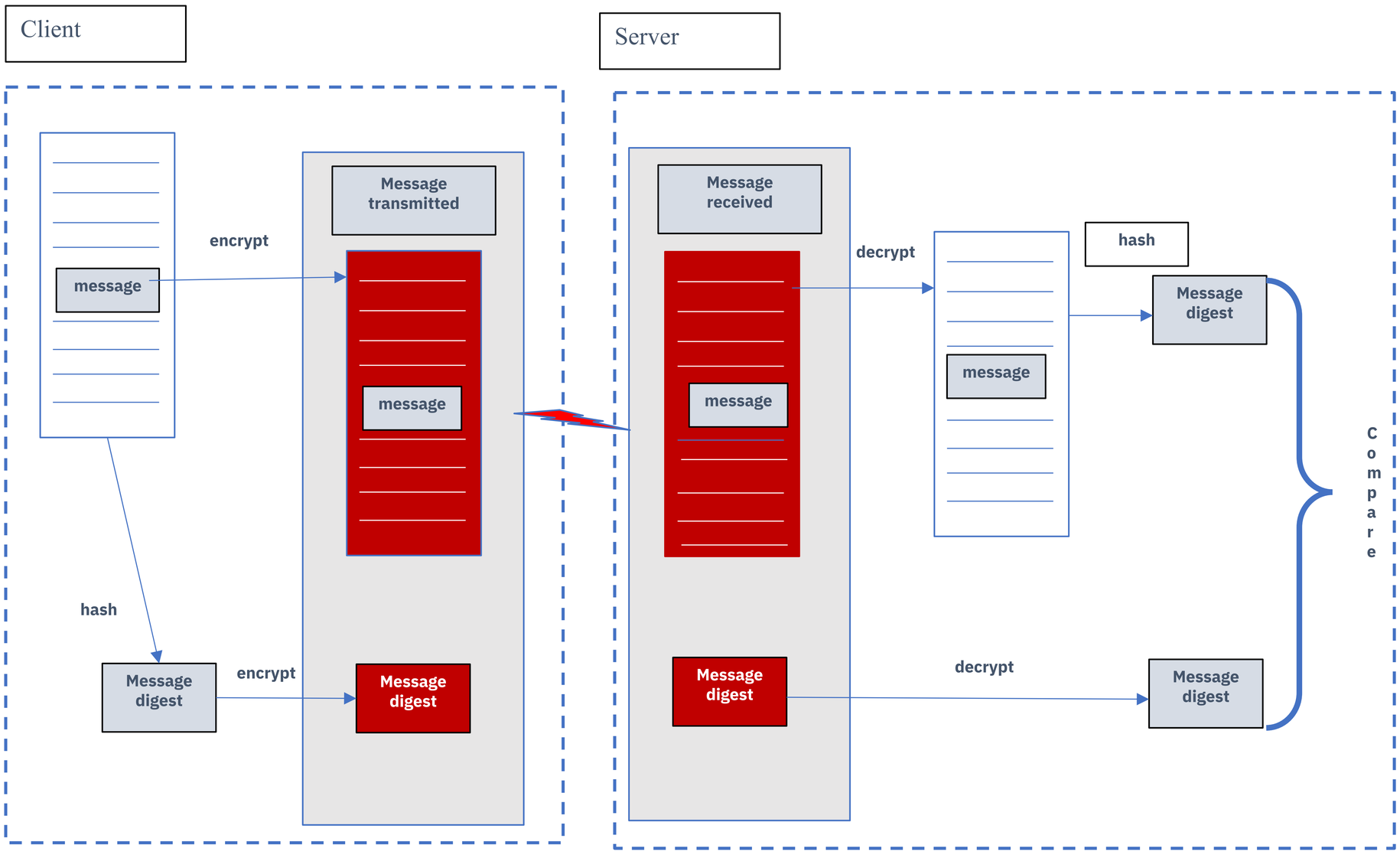
Table 1. CipherSpecs supported by IBM MQ and their equivalent CipherSuites

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

mitchj@us.ibm.com

Message Integrity and Encryption (client to server endpoint)



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

```
RACDCERT ID(LIBSERV) GENCERT SUBJECTSDN(CN('wg31.washington.ibm.com') +
 O('IBM') OU('LIBERTY')) WITHLABEL('Liberty Server Cert') ALTNAME(DOMAIN('wg31z.washington.ibm.com')
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

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
```

mitchj@us.ibm.com

Slide 69

The Liberty JSSE server XML configuration for outbound connections

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

   <ssl id="cicsTLSSettings"
      keyStoreRef="CICSKeyStore"
      trustStoreRef="CICSKeyStore"
      clientKeyAlias="Liberty Client Cert"/>
   <keyStore id="CICSKeyStore"
      location="safkeyring:///Liberty.CICS.KeyRing"
      password="password" type="JCERACFKS"
      fileBased="false" readOnly="true" />
   <ssl id="db2TLSSettings"
      keyStoreRef="Db2KeyStore"
      trustStoreRef="Db2KeyStore"
      clientKeyAlias="Liberty Client Cert"/>
   <keyStore id="Db2KeyStore"
      location="safkeyring:///Liberty.Db2.KeyRing"
      password="password" type="JCERACFKS"
     fileBased="false" readOnly="true" />
<ssl id="otherTLSSettings"
     keyStoreRef="OtherKeyStore"
     trustStoreRef="OtherKeyStore">
     outboundConnection
       host="wg31.washington.ibm.com"
       port="9555"
       clientCertificate="Client Cert"/>
  </ssl>
  <keyStore id="OtherKeyStore"
     location="safkeyring:///Other.KeyRing"
     password="password" type="JCERACFKS"
     fileBased="false" readOnly="true" />
```

```
<zosconnect_authorizationServer sslCertsRef="SSL repertoire"/>
<zosconnect_cicsIpicConnection sslCertsRef="cicsTLSSettings"/>
<zosconnect_db2Connection sslCertsRef="db2TLSettings"> *
<zosconnect_endpointConnect sslCertsRef= "SSL repertoire"/>
<zosconnect_zosConnectRestClient sslCertsRef="SSL repertoire"/>
<zosconnect_zosConnectServiceRestClientConnection sslCertsRef="SSL repertoire"/>
```

```
F BAQSTRT,REFRESH,KEYSTORE
F BAQSTRT,REFRESH,KEYSTORE,ID=CICSKeyStore
F BAQSTRT,REFRESH,KEYSTORE,ID=Db2KeyStore
F BAQSTRT.REFRESH,KEYSTORE,ID=OtherKeyStore
```

mitchj@us.ibm.com

* OpenAPI 3 only

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
```

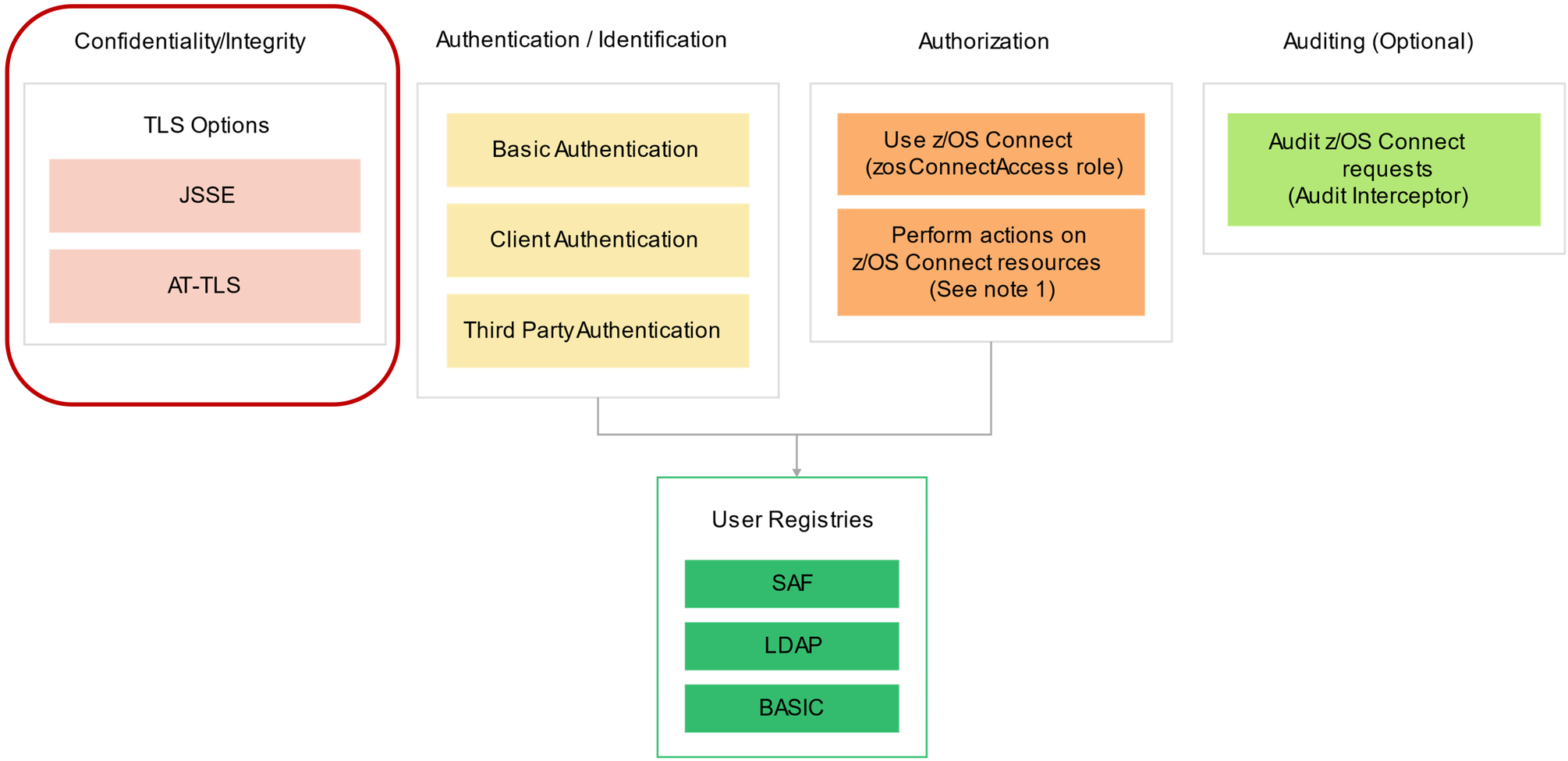
```
<httpEndpoint id="defaultHttpEndpoint"
    host="*"
    httpPort="9080"
    httpsPort="9443" />

<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

mitchj@us.ibm.com

Liberty and z/OS Connect EE security options (OpenAPI 2)

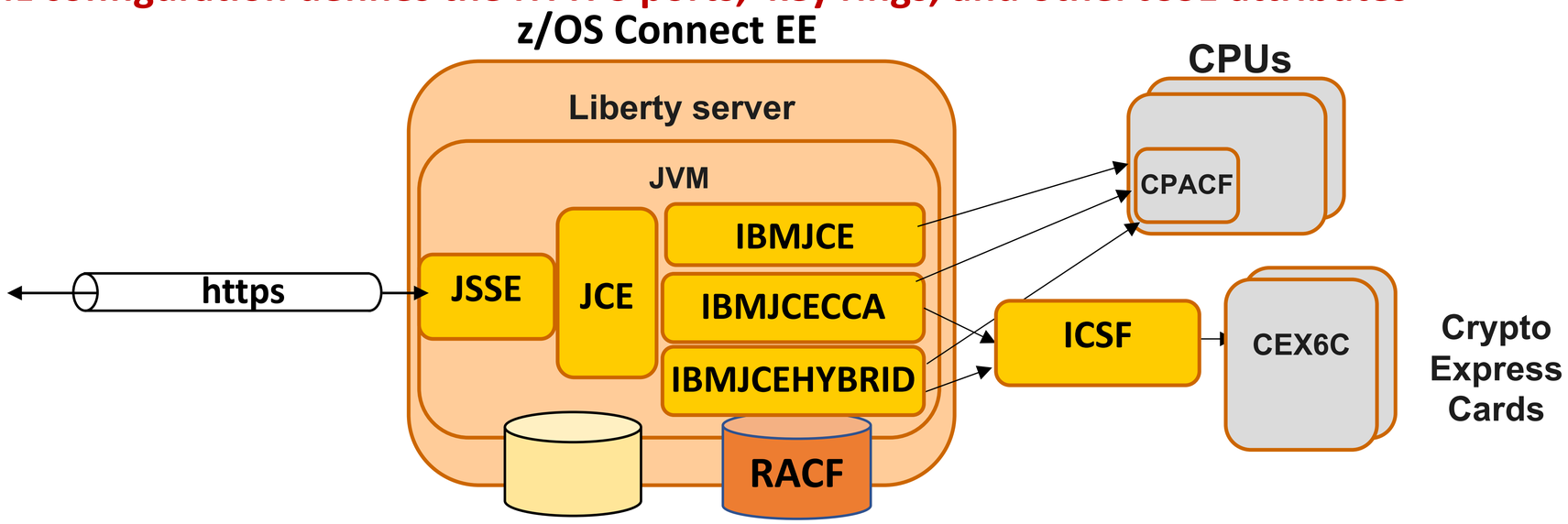


The actions which can be controlled by authorization are deploying, querying, updating, starting, stopping and deleting of APIs, services and API requesters.

mitchj@us.ibm.com

Using JSSE with Liberty

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



server.xml

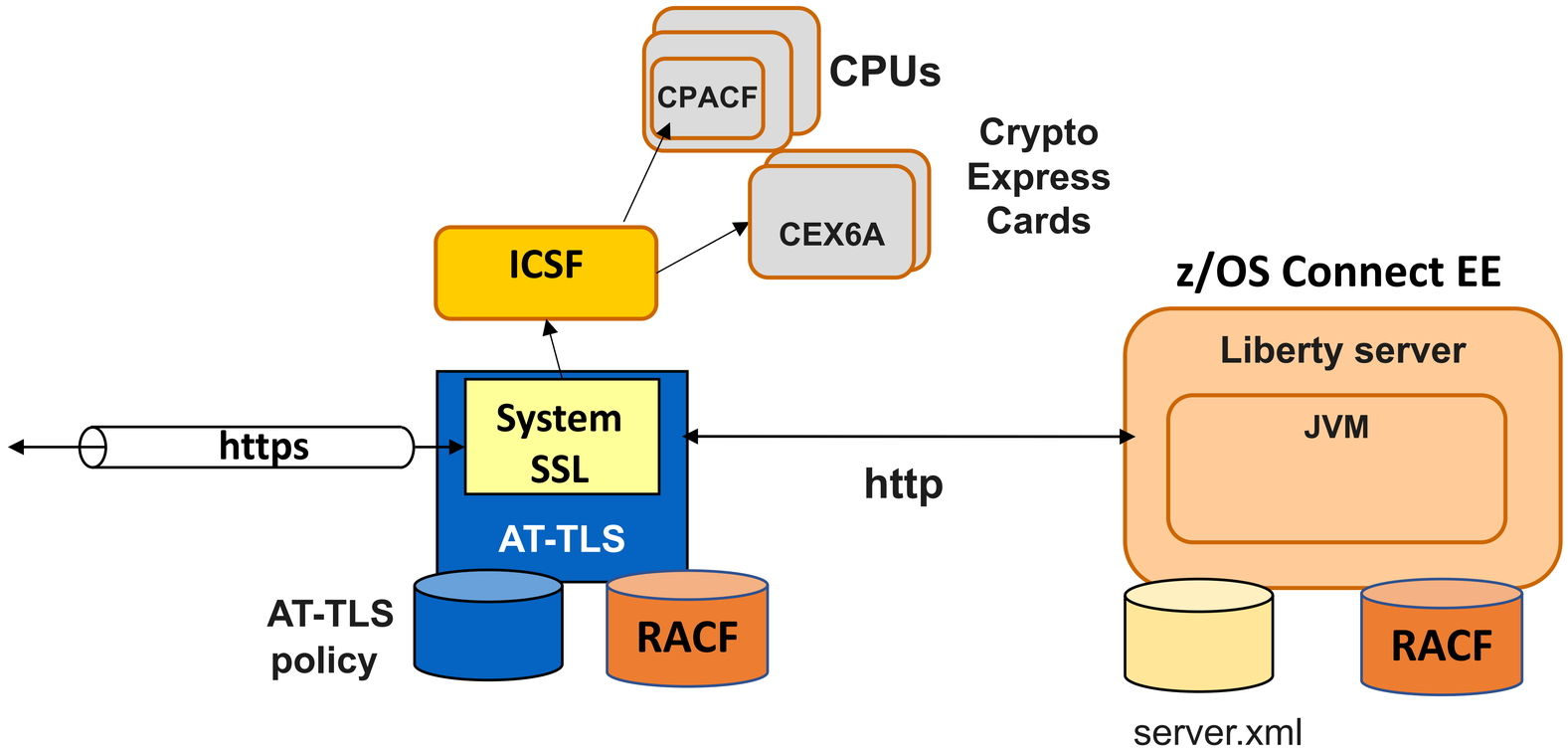
- z/OS Connect EE support for TLS is based on **Liberty** server support
- **Java Secure Socket Extension** (JSSE) API provides framework and Java implementation of TLS protocols used by Liberty HTTPS support
- **Java Cryptography Extension** (JCE) is standard extension to the Java Platform that provides 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.

mitchj@us.ibm.com

Slide 73

Using AT-TLS with Liberty

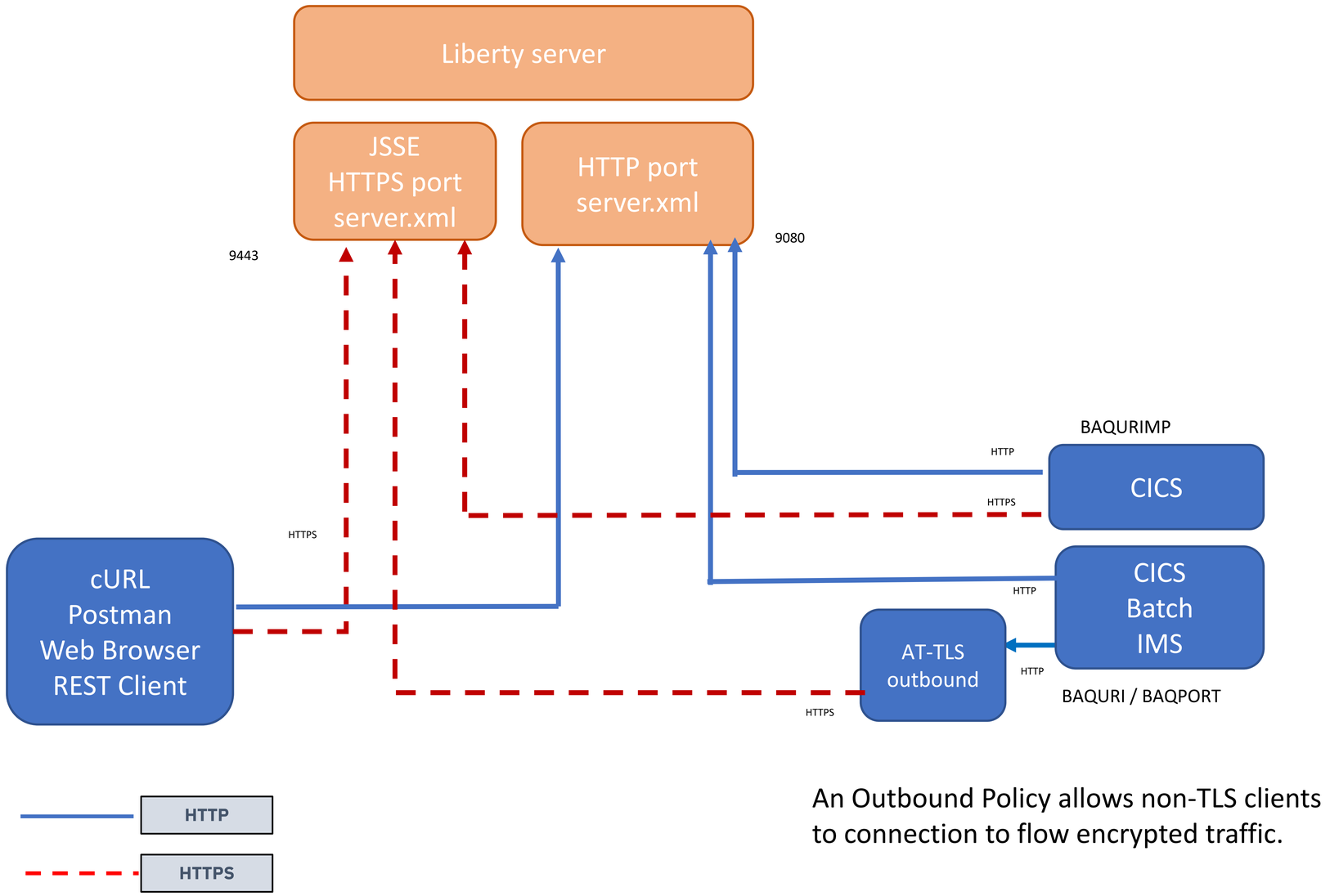
The server XML configuration uses <u>no</u> HTTPS protocol, key rings or other JSSE attributes



- **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

mitchj@us.ibm.com

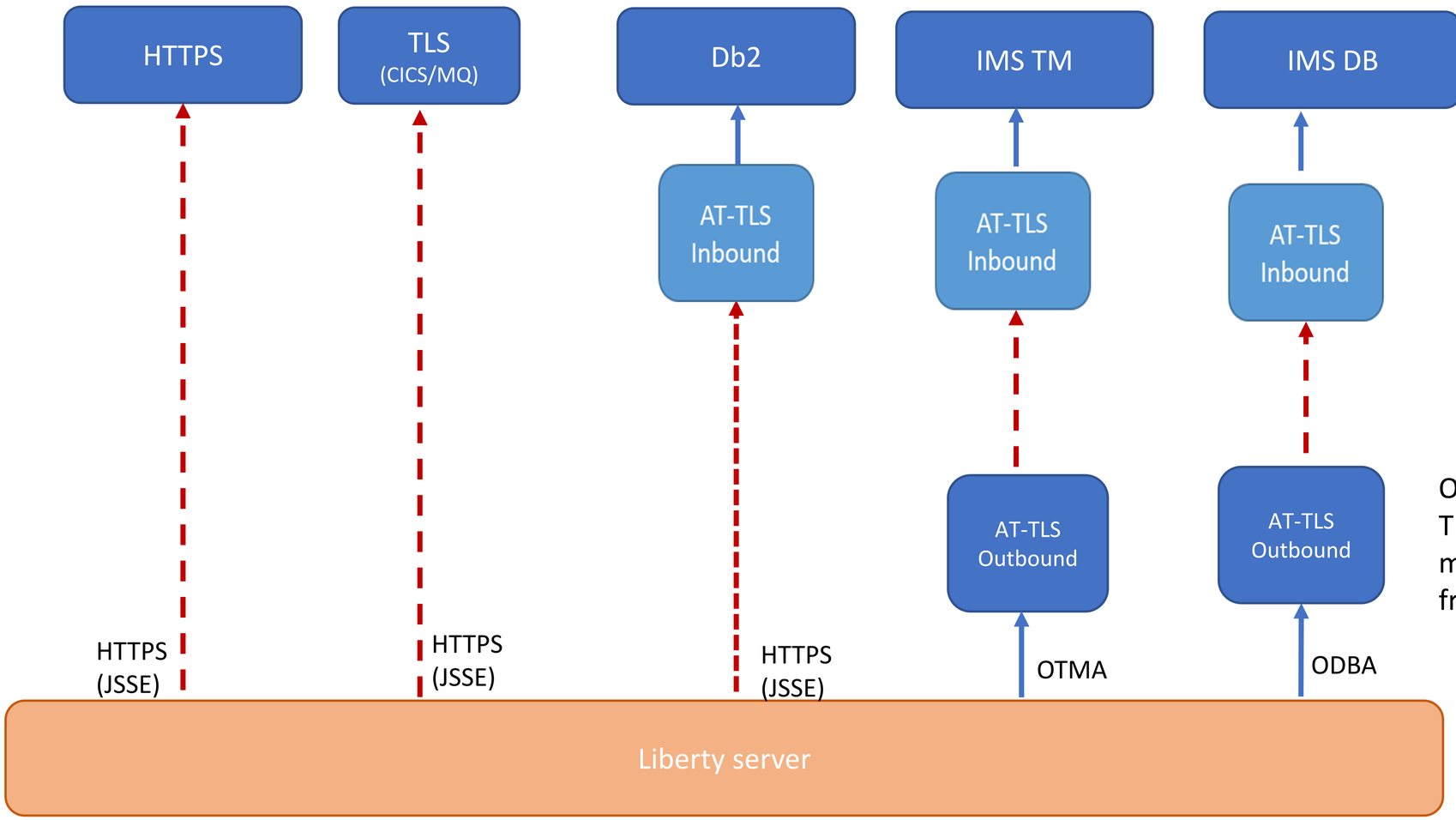
TLS client encryption to a Liberty server scenarios



An Outbound Policy allows non-TLS clients to connection to flow encrypted traffic.

[1] mitchj@us.ibm.com

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



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

Outbound Policies provide TLS support for non-JSSE managed request outbound from Liberty

mitchj@us.ibm.com

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

mitchj@us.ibm.com

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"
      clientKeyAlias="Liberty Client Cert"/>

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

SSL repertoires

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

F BAQSTRT,REFRESH,KEYSTORE

mitchj@us.ibm.com

Let’s explore TLS options using the contents of these key rings

Liberty’s outbound key ring

Liberty’s inbound key ring

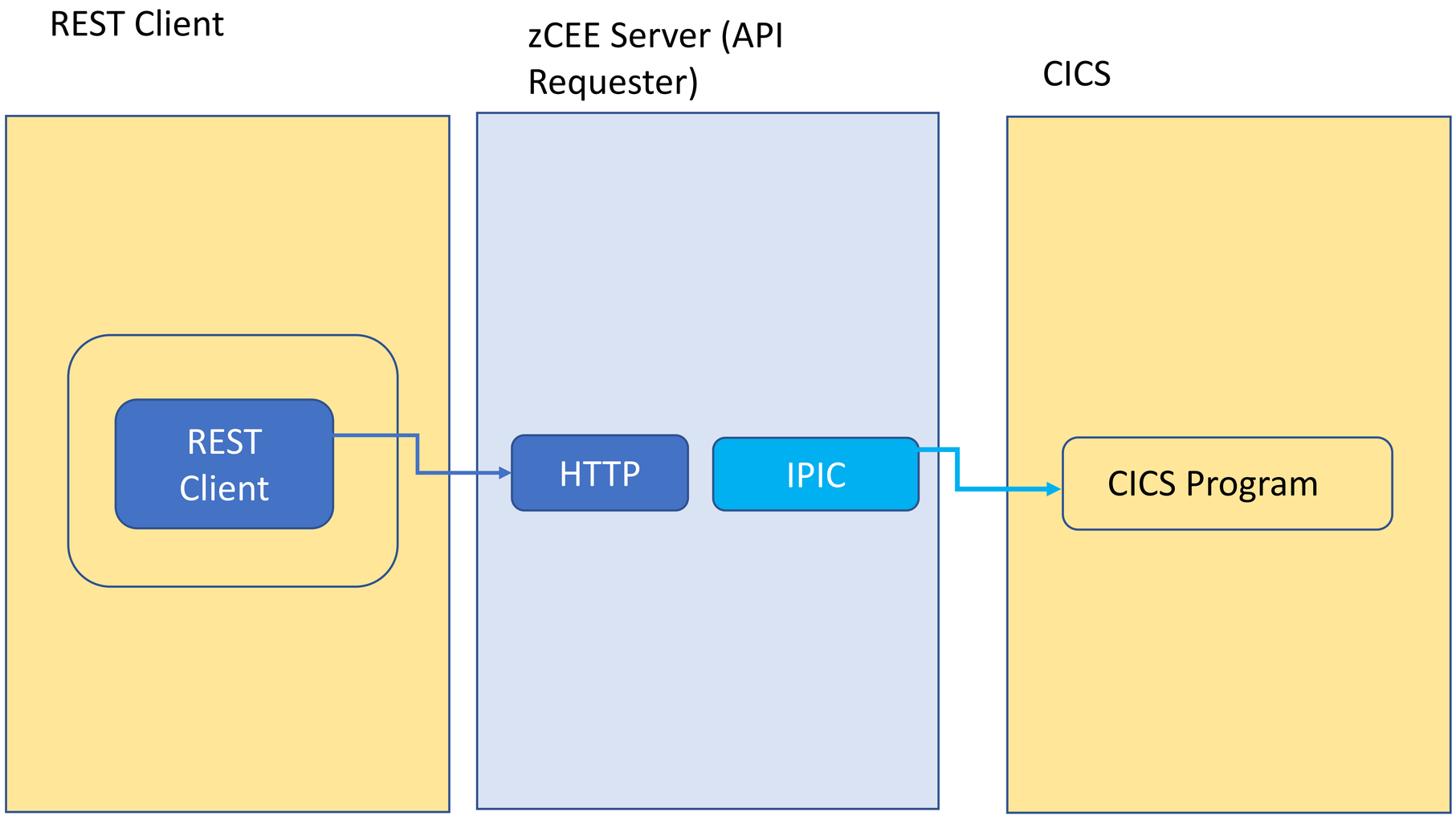
```
Digital ring information for user LIBSERV:
  Ring:
       >zCEE.KeyRing<
  Certificate Label Name              Cert Owner     USAGE      DEFAULT
  --------------------------------    ------------   --------   -------
  zCEE CA                             CERTAUTH       CERTAUTH     NO
  Liberty CA                          CERTAUTH       CERTAUTH     NO
  zCEE Client Cert                    ID(LIBSERV)    PERSONAL     YES
  xyz Client Cert                     ID(LIBSERV)    PERSONAL     No
  DB2 CA                              CERTAUTH       CERTAUTH     NO
  MQ CA                               CERTAUTH       CERTAUTH     NO
  CICS CA                             CERTAUTH       CERTAUTH     NO

  Ring:
       >Liberty.KeyRing<
  Certificate Label Name              Cert Owner     USAGE      DEFAULT
  --------------------------------    ------------   --------   -------
  Liberty Server Cert                 ID(LIBSERV)    PERSONAL     YES
  Liberty CA                          CERTAUTH       CERTAUTH     NO
  zCEE CA                             CERTAUTH       CERTAUTH     NO
  CICS CA                             CERTAUTH       CERTAUTH     NO

Digital ring information for user CICSSTC:
  Ring:
       >CICS.KeyRing<
  Certificate Label Name              Cert Owner     USAGE      DEFAULT
  --------------------------------    ------------   --------   -------
  CICS CA                             CERTAUTH       CERTAUTH     NO
  CICS Client Cert                    ID(CICSSTC)    PERSONAL     YES
  Liberty CA                          CERTAUTH       CERTAUTH     NO
  zCEE CA                             CERTAUTH       CERTAUTH     NO

Digital ring information for user DB2USER:
  Ring:
       >Db2.KeyRing<
  Certificate Label Name              Cert Owner     USAGE      DEFAULT
  --------------------------------    ------------   --------   -------
  DB2 CA                              CERTAUTH       CERTAUTH     NO
  zCEE CA                             CERTAUTH       CERTAUTH     NO
  DB2USER                             ID(DB2USER)    PERSONAL     YES
```

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

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

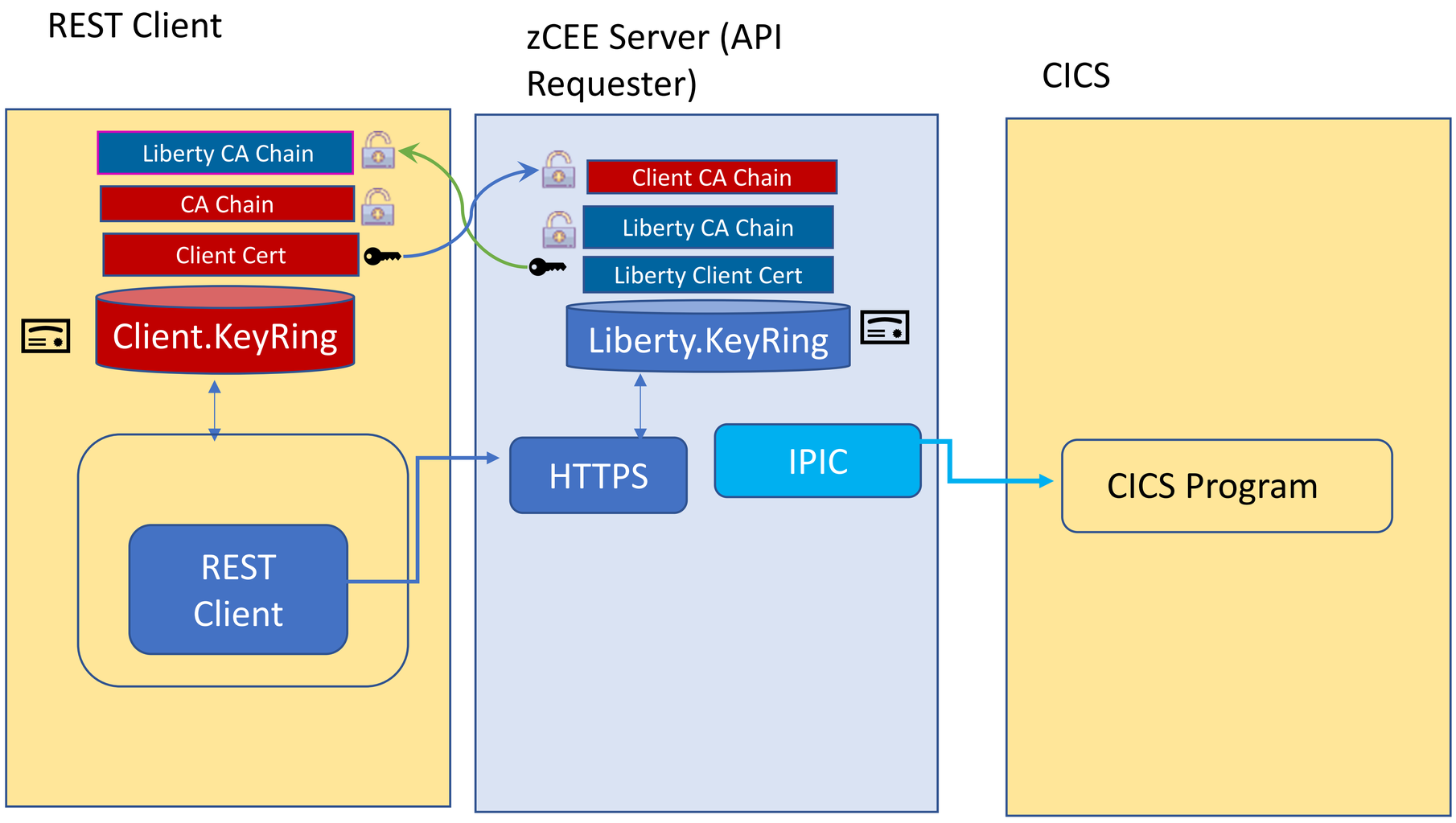
mitchj@us.ibm.com

No TLS between any endpoints



mitchj@us.ibm.com

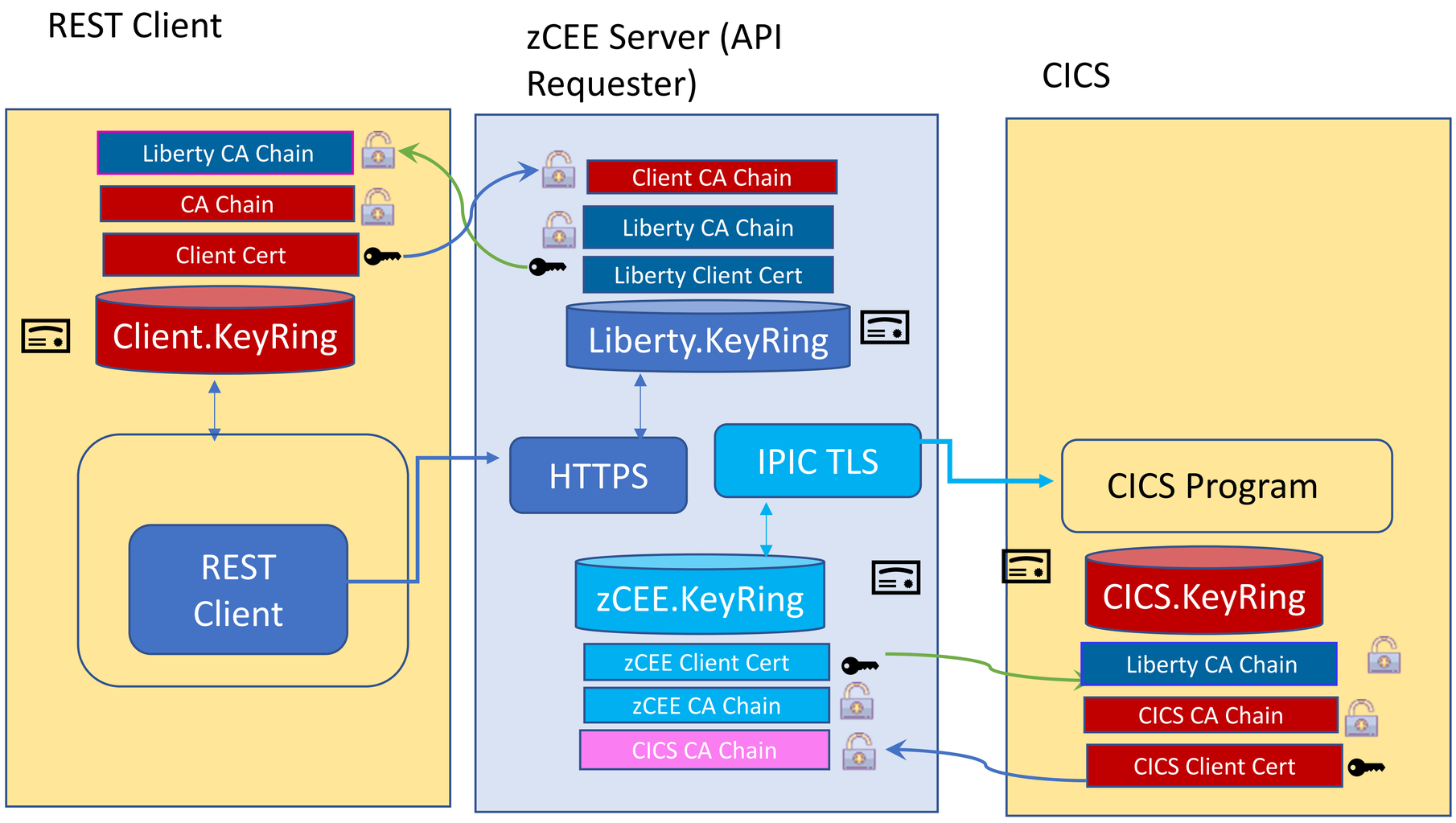
TLS handshake between the client and z/OS Connect server



mitchj@us.ibm.com

Slide 81

Independent TLS handshakes between all endpoints

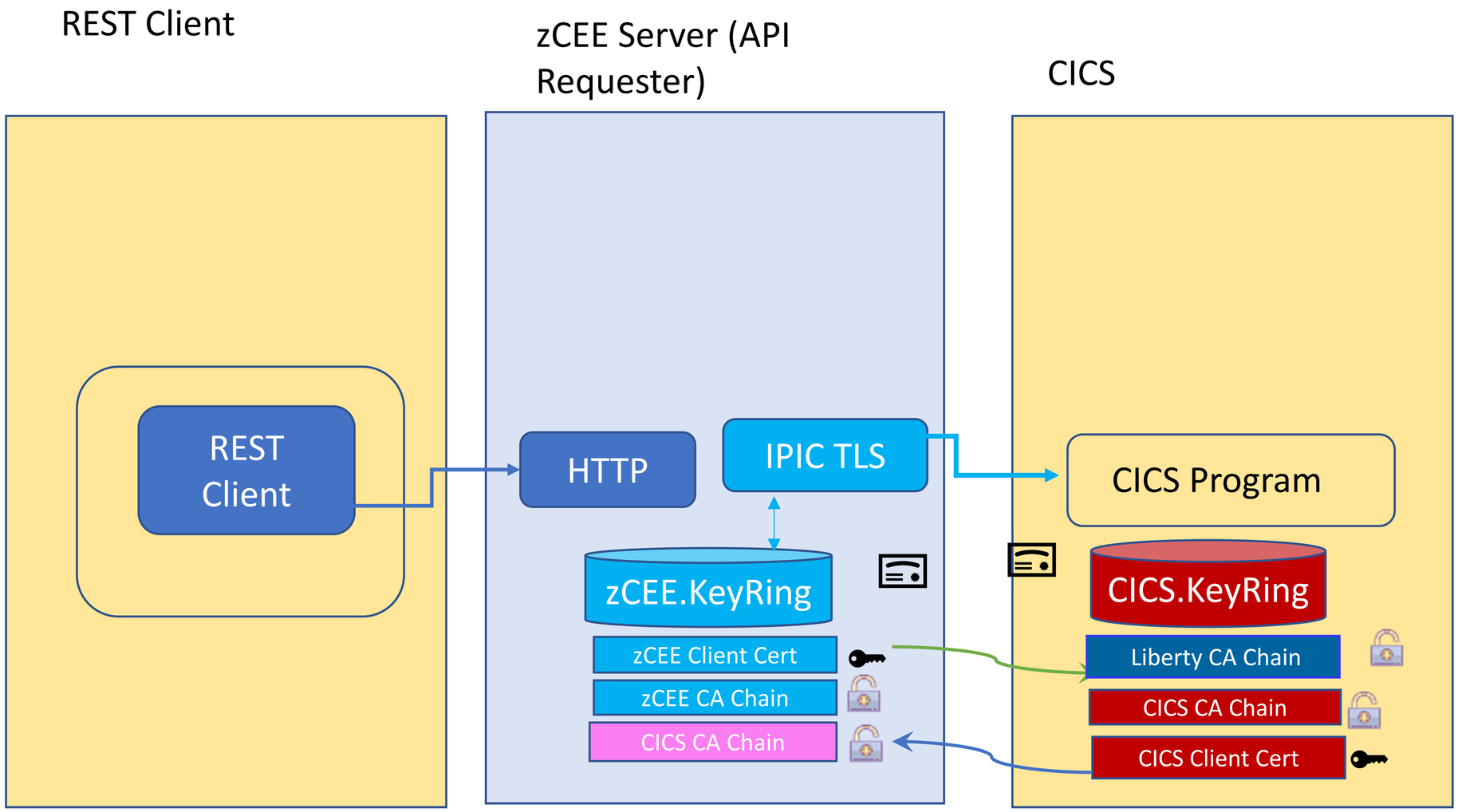



```
KEYRING=CICS.KeyRing,
ENCRYPTION=STRONG,
MAXSSLTCBS=8,
SSLDELAY=600
```

mitchj@us.ibm.com

Slide 82

TLS handshake between the z/OS Connect server and the target endpoint

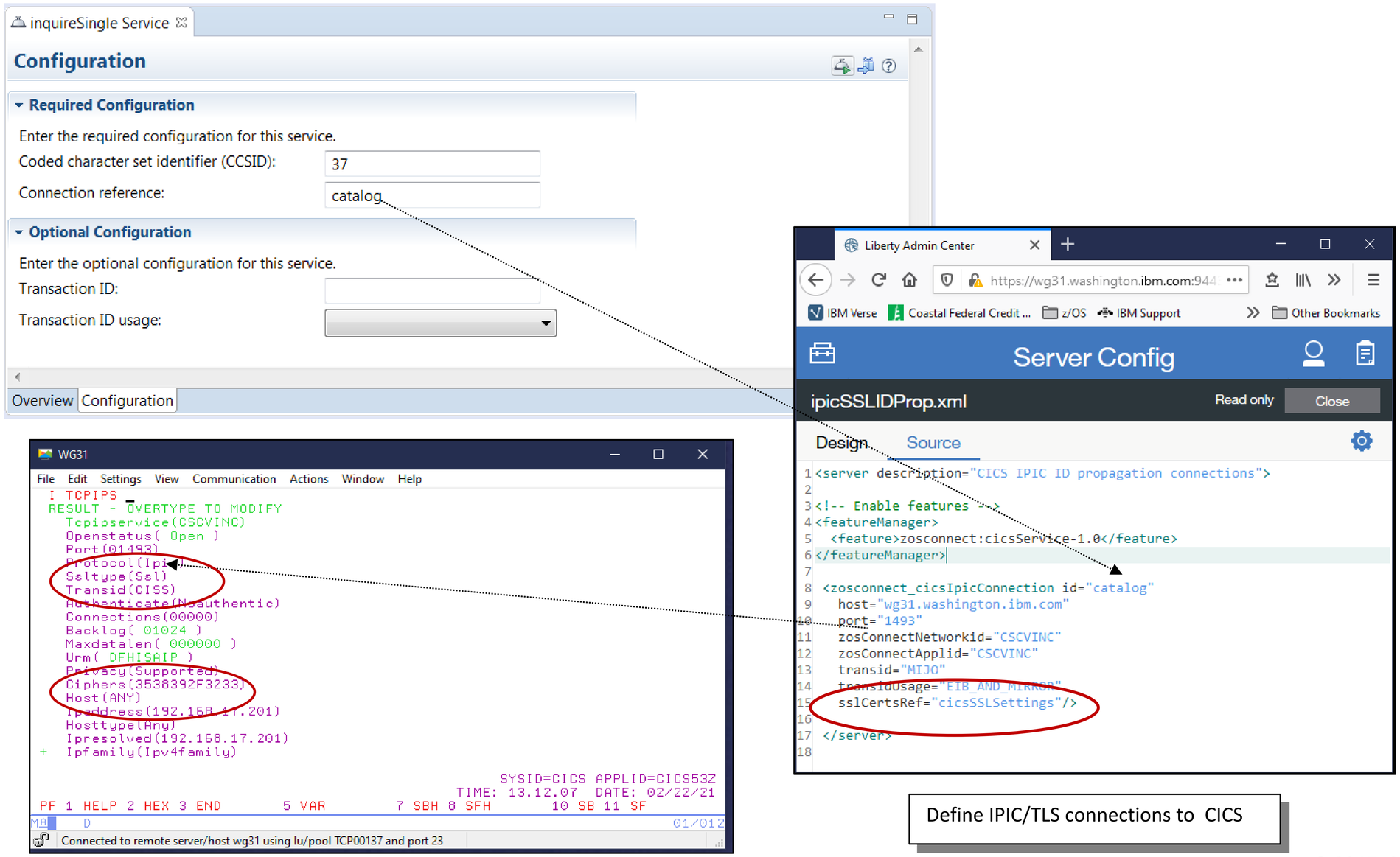



```
KEYRING=CICS.KeyRing,
ENCRYPTION=STRONG,
MAXSSLTCBS=8,
SSLDELAY=600
```

mitchj@us.ibm.com

CICS IPIC using TLS

The server.xml file is the key configuration file:



Define IPIC/TLS connections to CICS

https://www.ibm.com/support/knowledgecenter/SSLTBW_2.4.0/com.ibm.zos.v2r4.gska100/csdcwh.htm

mitchj@us.ibm.com

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

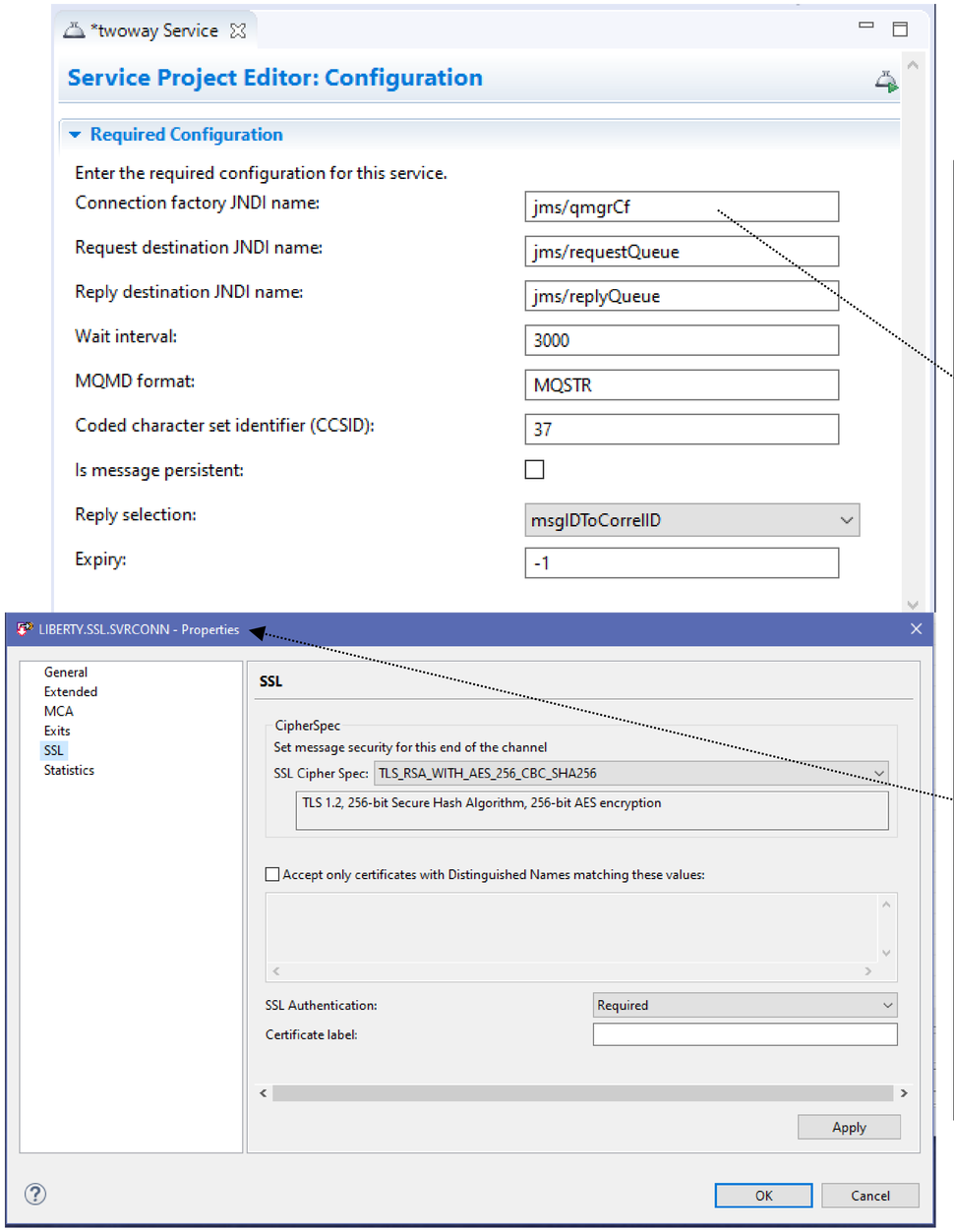
2-character cipher number	4-character cipher number	Short name	Description [1]	FIPS 140-2	Base security level FMID HCPT440	Security level 3 FMID JCPT441
00	0000	TLS_NULL_WITH_NULL_NULL	No encryption or message authentication and RSA key exchange		X	X
01	0001	TLS_RSA_WITH_NULL_MD5	No encryption with MD5 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

mitchj@us.ibm.com

MQ JMS using TLS

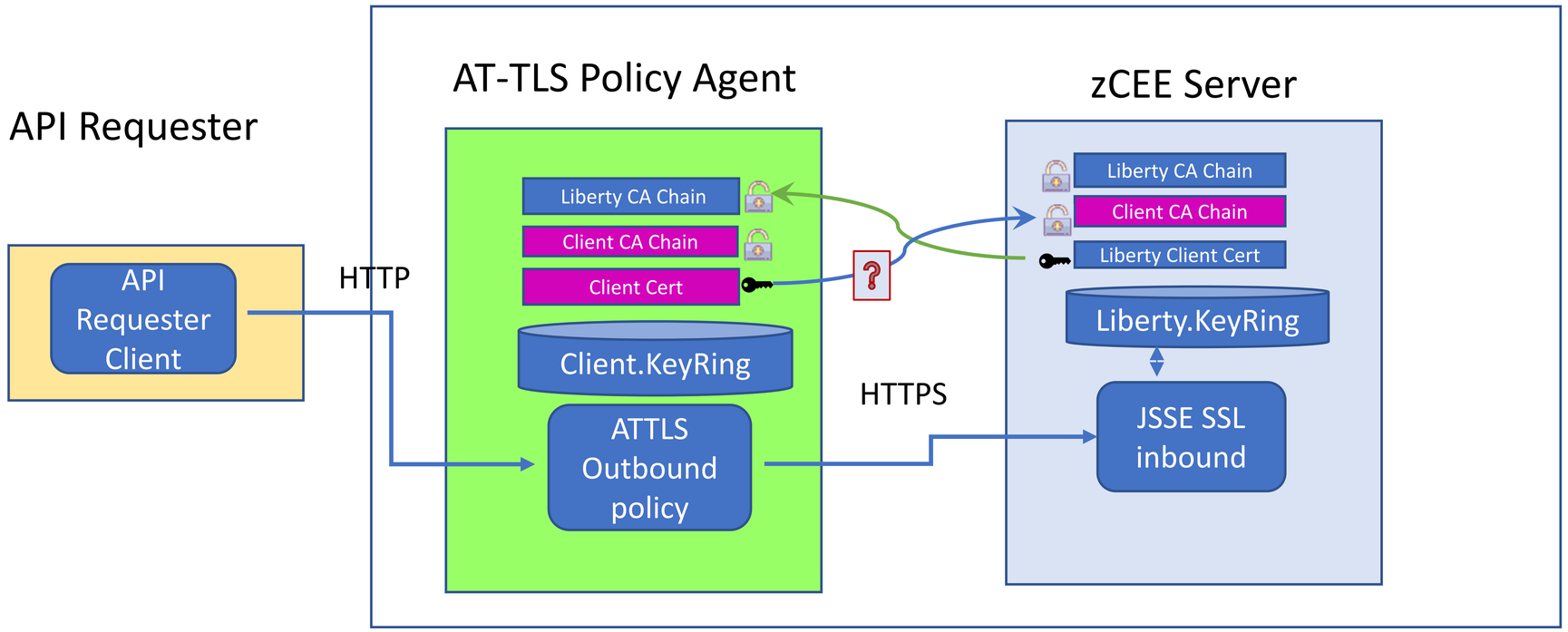
The server.xml file is the key configuration file:



mitchj@us.ibm.com

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>
```

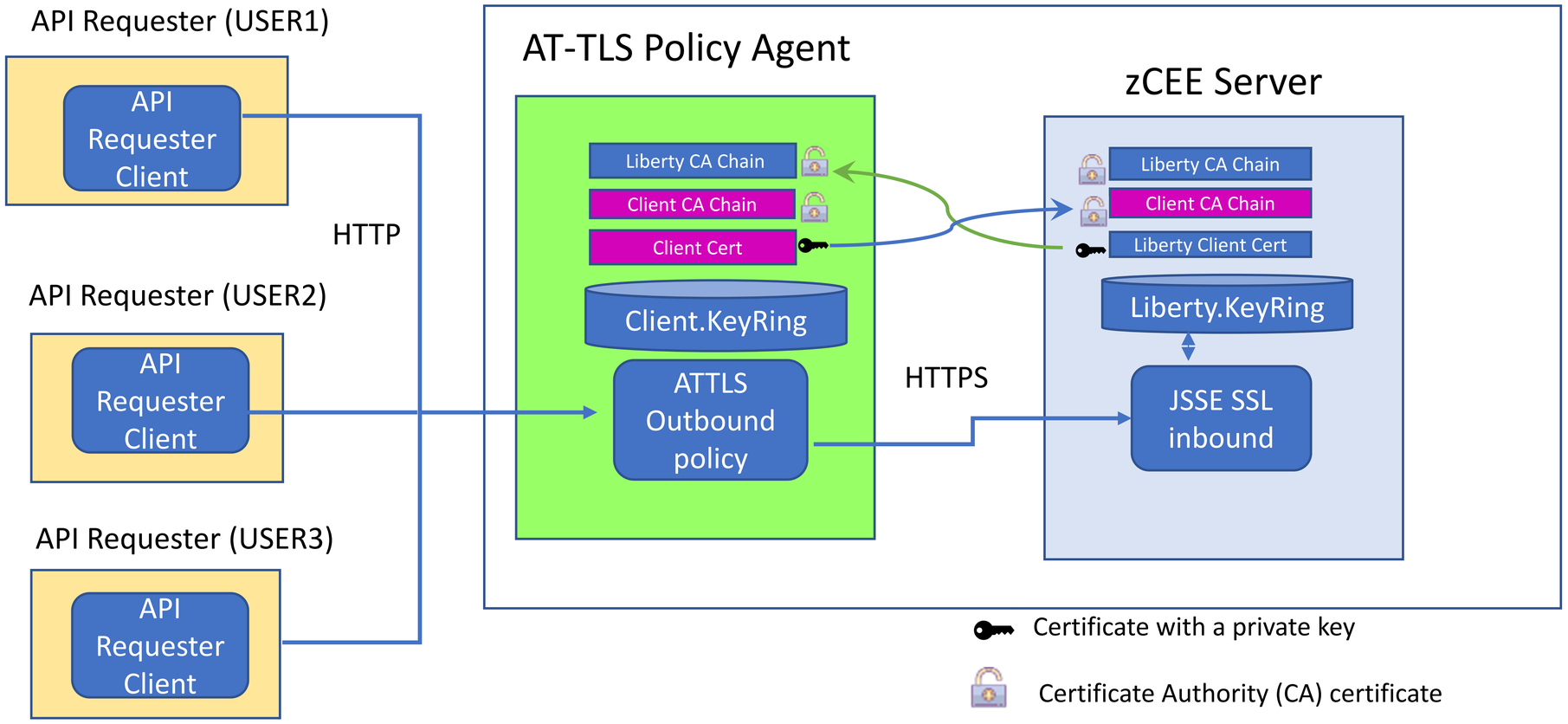
Question if this really needed, remember TLS encryption is independent of TLS authentication.

mitchj@us.ibm.com

Slide 87

AT-TLS - outbound policy handshake scenario

Use of a common key ring name for 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.

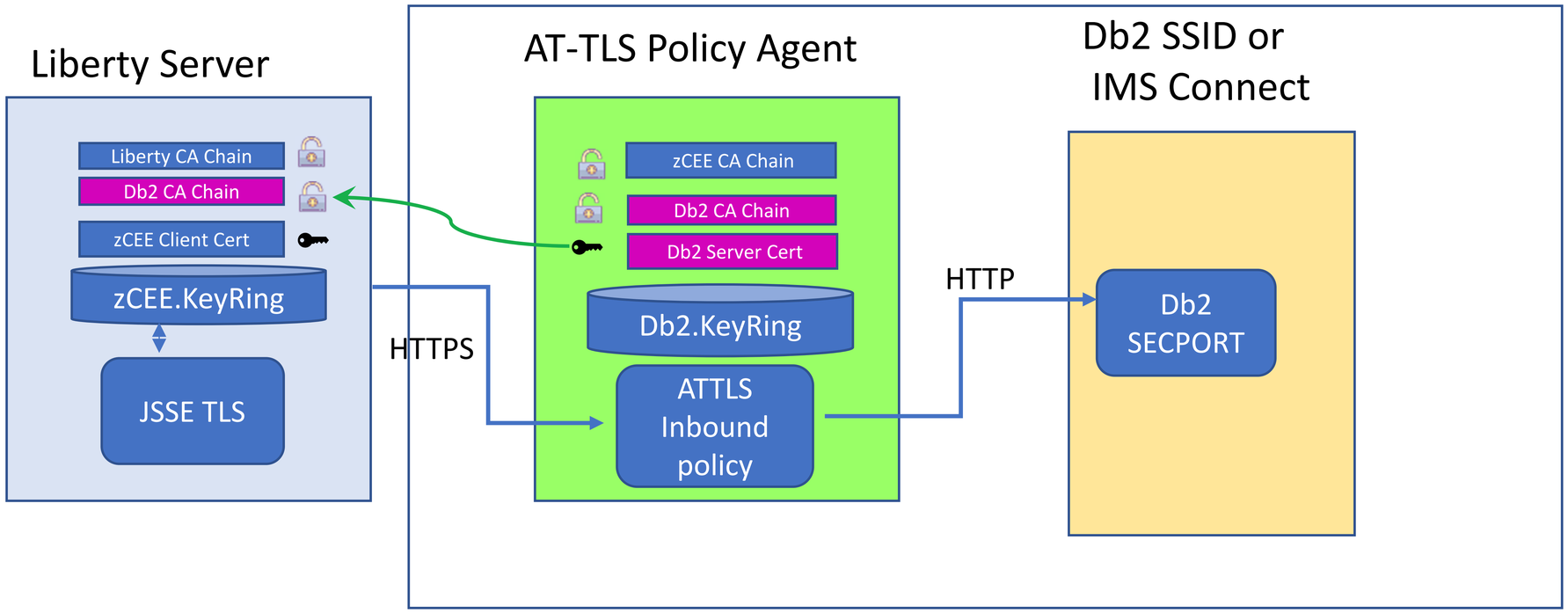
This is a situation when AT-TLS mutual authentication has a benefit.

mitchj@us.ibm.com

Slide 88

AT-TLS - inbound policy handshake scenario (Db2 and IMS)

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



Note that DB2 is AT-TLS aware
IMS is AT-TLS unware

Certificate with a private key

Certificate Authority (CA) certificate

mitchj@us.ibm.com

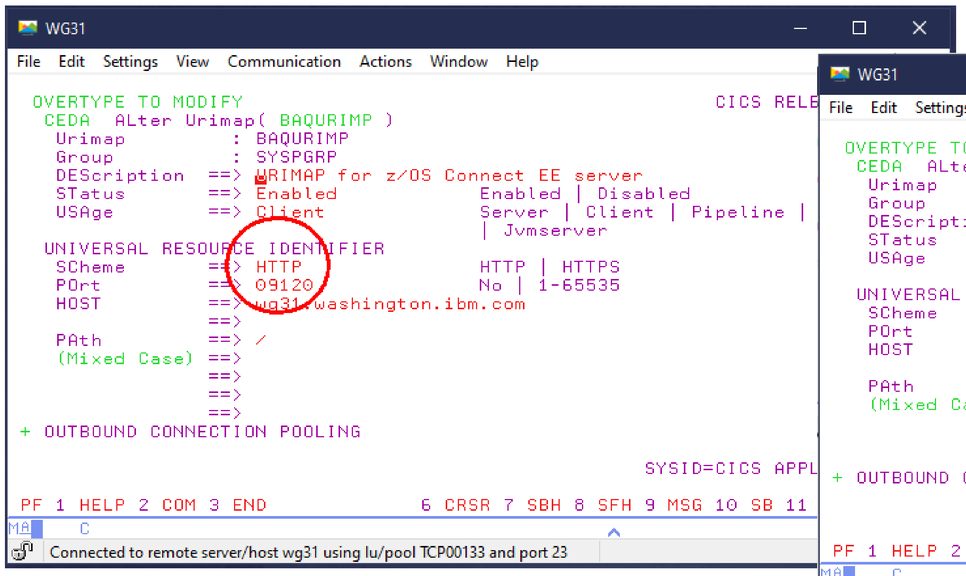
API Requester - HTTP v HTTPS

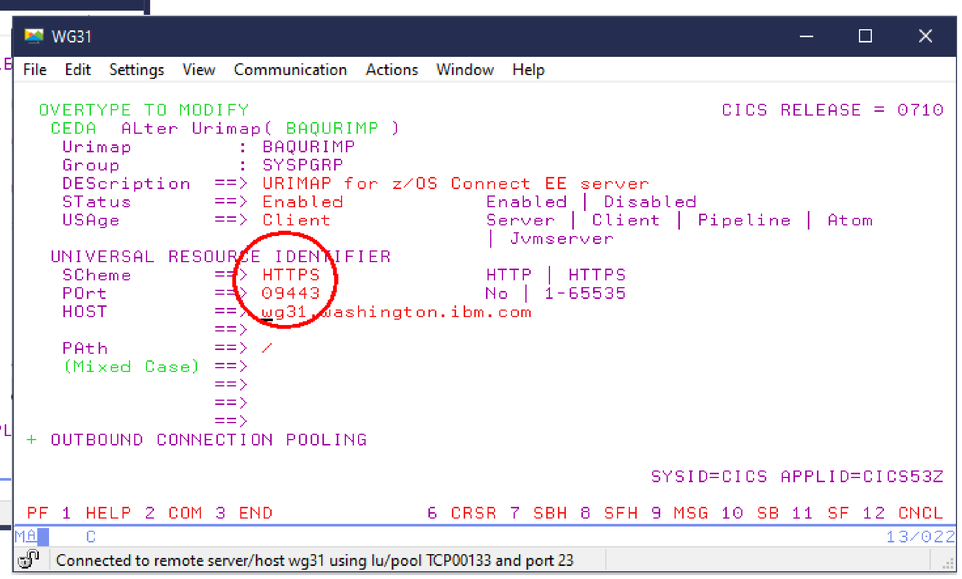
MVS Batch and IMS with and without an outbound AT-TLS policy

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

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

CICS URIMAPs





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

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

mitchj@us.ibm.com

Connection Management

- Amount of time before a connection can be discarded by pool maintenance.
- Amount of time after which a connection request times out.
- Amount of time a connection can be unused or idle until it can be discarded during pool maintenance.
- Maximum number of physical connections for a pool.
- Minimum number of physical connections to maintain in the pool.
- Specifies which connections to destroy when a stale connection is detected in a pool.
- Amount of time between runs of the pool maintenance thread.

```
<connectionManger id="ConMgr1"
agedTimout=-1
connectionTimeout=30
maxIdleTIme=1800
maxPoolSize=50
minPoolSize=0
purgePolicy="EntirePool"
reapTIme=180/>
```

mitchj@us.ibm.com

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

Slide 92

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

mitchj@us.ibm.com

Slide 93

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

mitchj@us.ibm.com

Slide 94

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" />
```

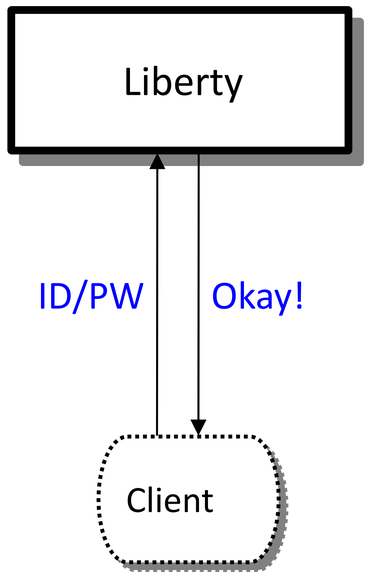
See URL https://www.ibm.com/support/pages/node/6209109 for details on implementing IBMJCECCA and IBMJCEHYBRID hardware encryption providers

mitchj@us.ibm.com

Authentication - Third Party Authentication

Several different ways this can be accomplished:

Basic Authentication

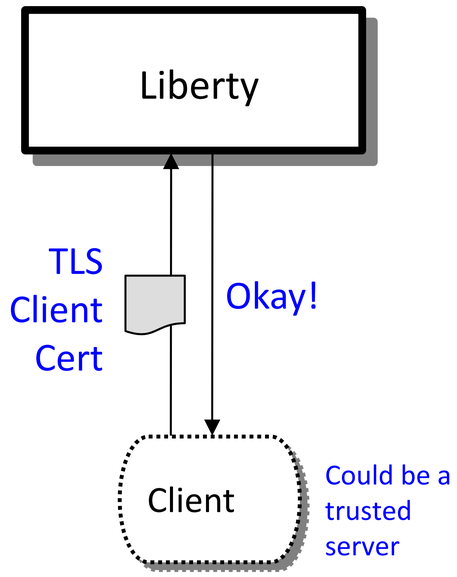


Server prompts for ID/PW

Client supplies ID/PW or ID/PassTicket

Server checks registry:
- Basic (server.xml)
- SAF

Client Certificate



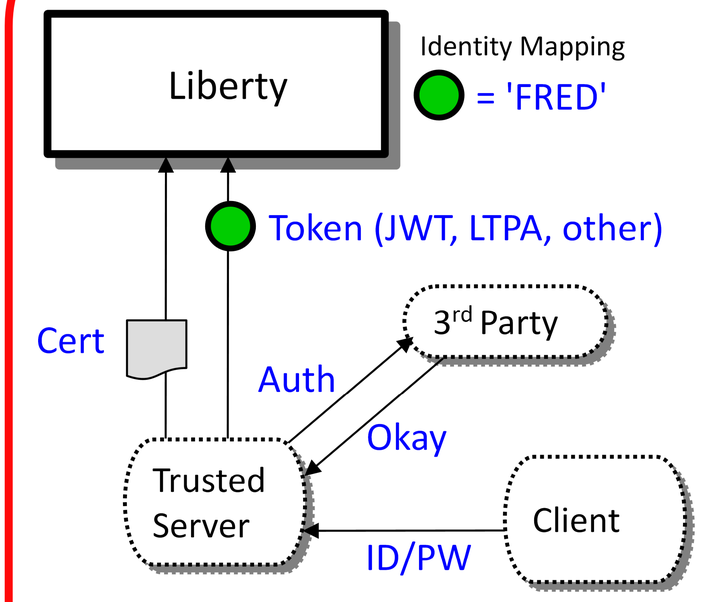
Server prompts for client certificate.

Client supplies certificate

Server validates client certificate and maps to an identity

Registry options:
- SAF

Third Party Authentication



Client authenticates to 3rd party sever

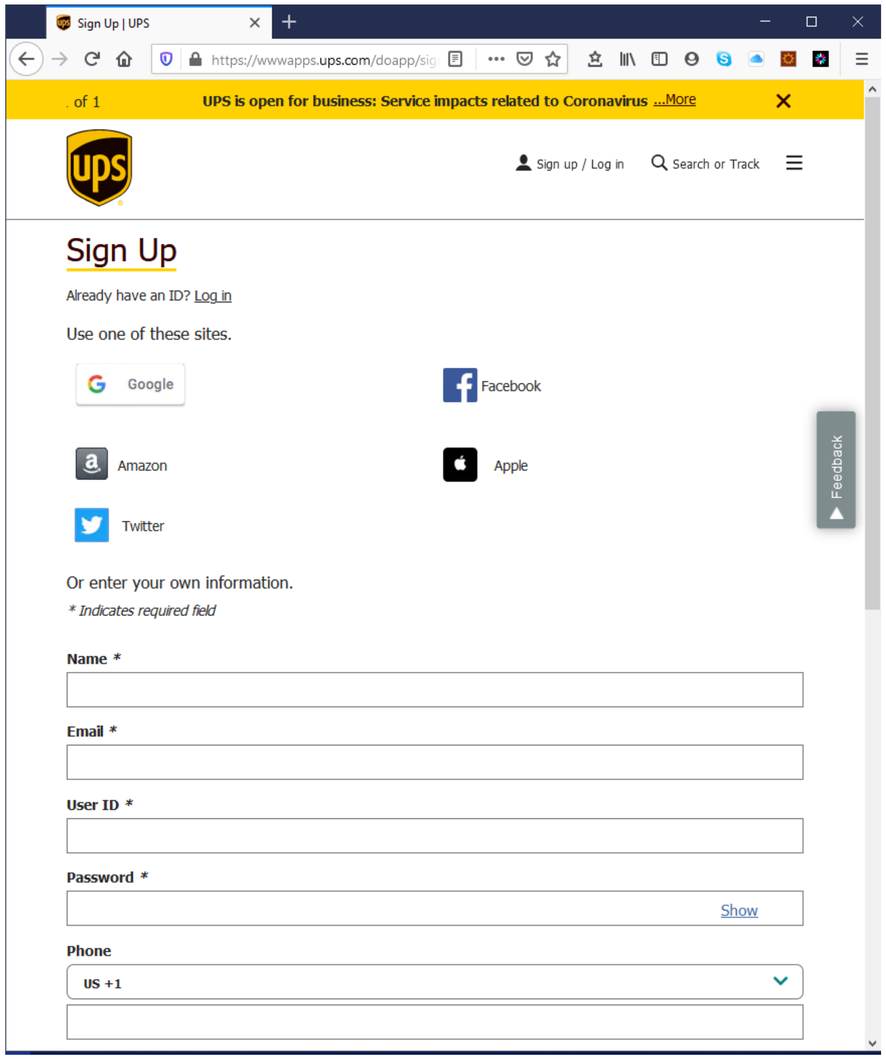
Client receives a trusted 3rd party token

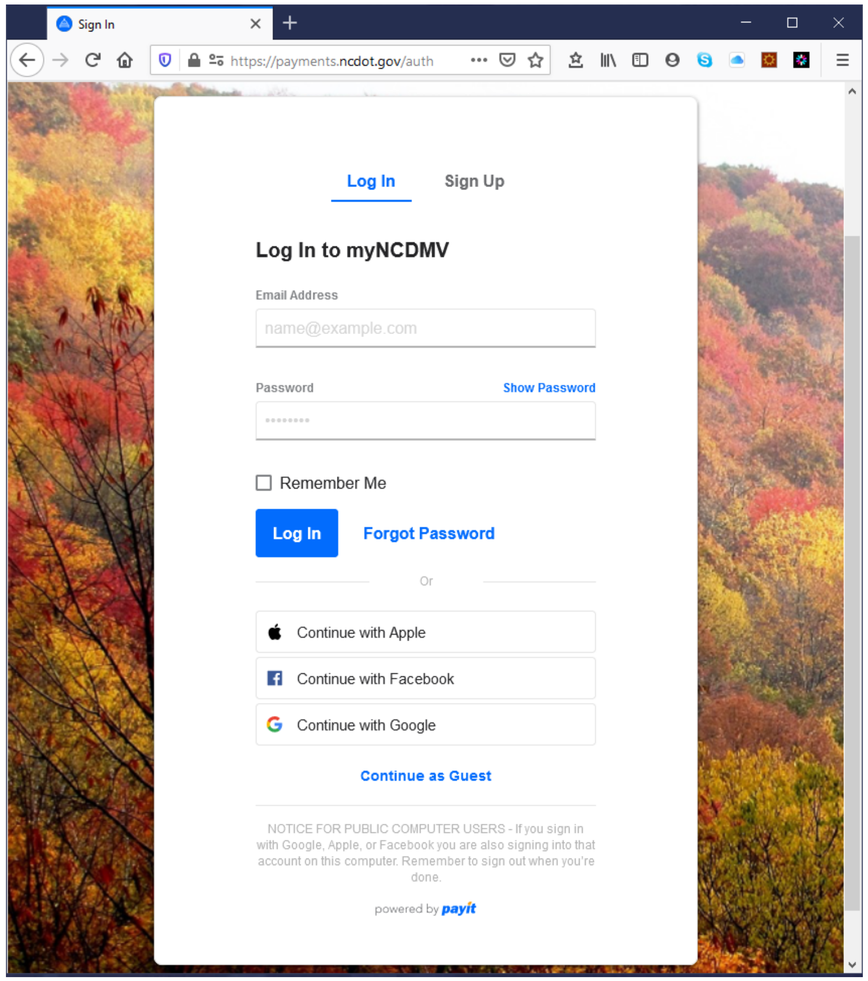
Token flows to Liberty z/OS and is mapped to an identity

Registry options:
- **We may know these detail.**

mitchj@us.ibm.com

Third Party Authentication Examples





mitchj@us.ibm.com

Open security standards

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

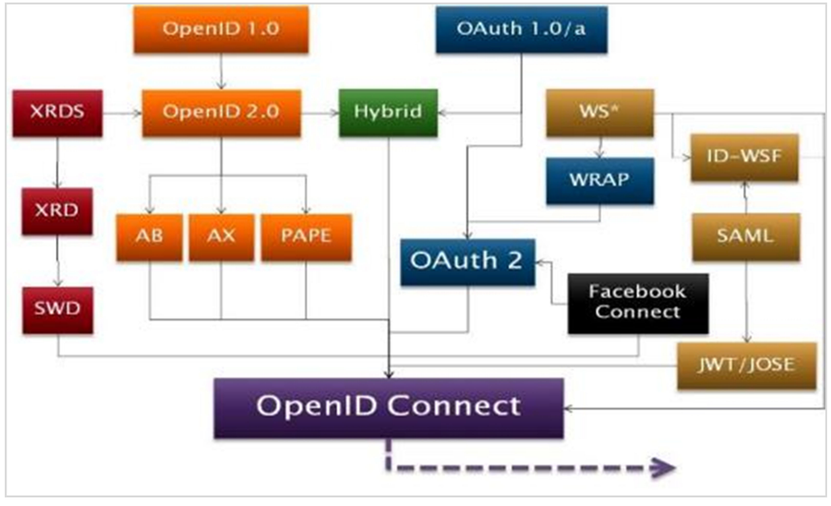
See the YouTube videos:

OAuth 2.0 and OpenID Connect (in plain English)

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

OpenID Connect on Liberty

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



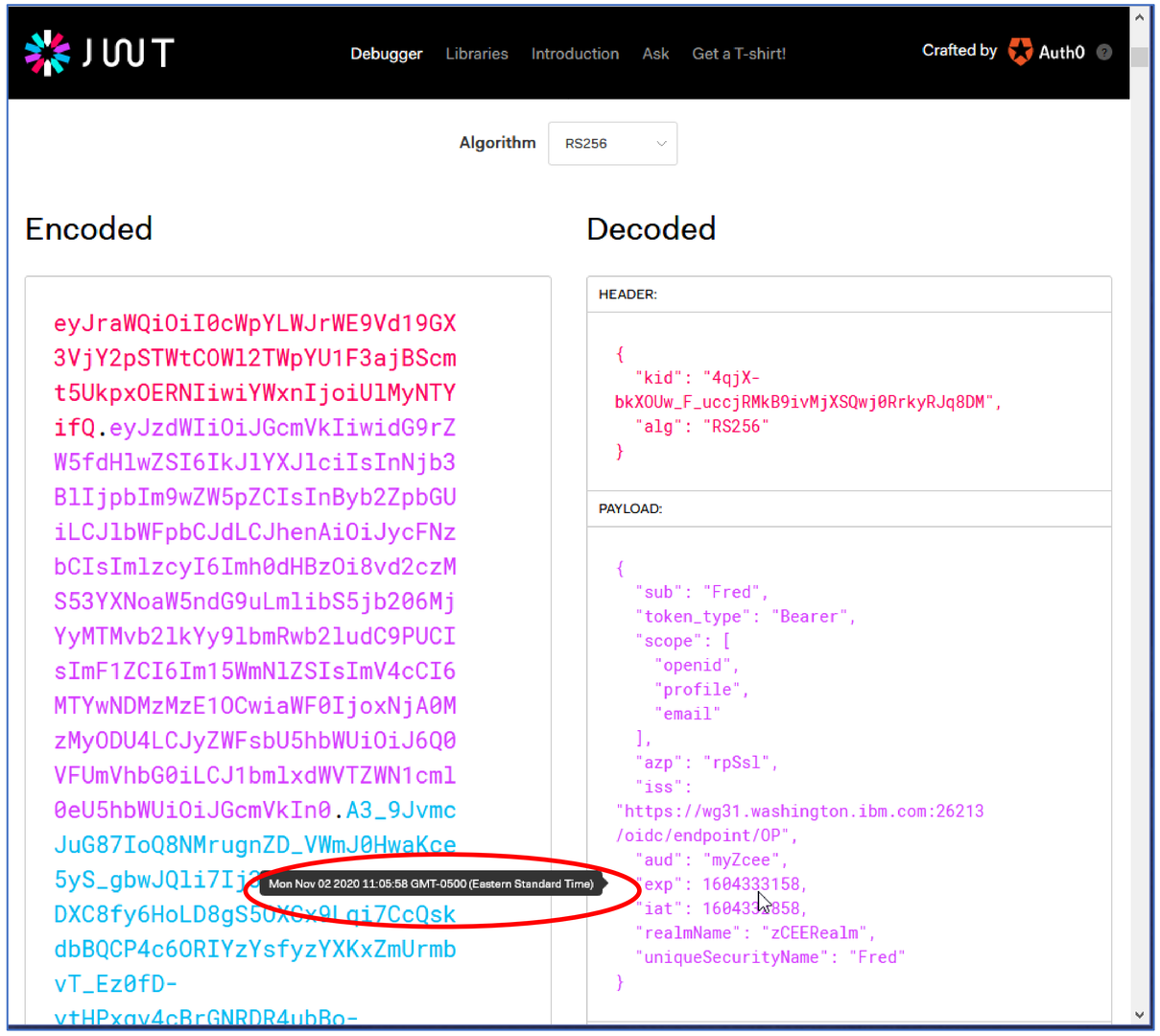
mitchj@us.ibm.com

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



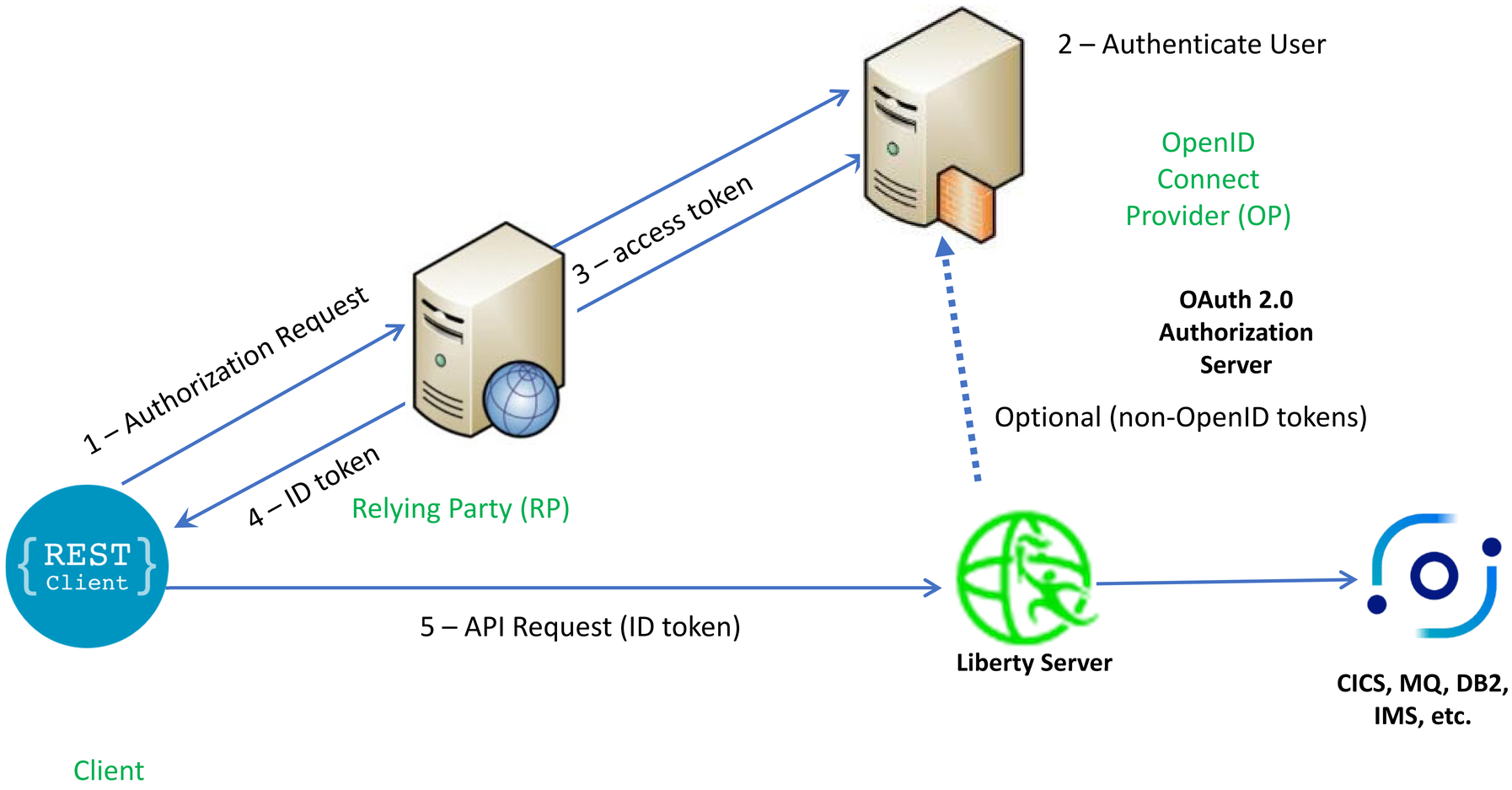
Values derived from the OAUTH configuration:

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

https://jwt.io

mitchj@us.ibm.com

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



mitchj@us.ibm.com

OpenID Connect/OAuth and z/OS Connect

- ***From the z/OS Connect Knowledge Center:*** *z/OS Connect EE security can operate with traditional z/OS security, for example, System Authorization Facility (SAF) and also with open standards such as Transport Layer Security (TLS), JSON Web Token (JWT), and* ***OpenID Connect.***

- ***From the OpenID Core specification:*** *OpenID Connect 1.0 is a simple identity layer on top of the OAuth 2.0 protocol. It enables Clients to verify the identity of the End-User based on the authentication performed by an Authorization Server, as well as to obtain basic profile information about the End-User in an interoperable and REST-like manner.*

- ***OAuth 2.0 Core (RFC 6749) Specifications:*** https://tools.ietf.org/html/rfc6749

- ***OpenID Connect Core Specifications***: https://openid.net/specs/openid-connect-core-1_0.html

- ***Again, for a very good explanation of this topic see YouTube video*** *OAuth 2.0 and OpenID Connect (in plain English)* https://www.youtube.com/watch?v=996OiexHze0

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 (JTW). 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 for more information about the extensions.

Tech/Tip: Let's explore a flow using a Liberty OpendID Provider as an example

This Liberty server configuration provides a good example of the workings of an authorization server.

```
<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>
</oauthProvider>
```

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.

[1] mitchj@us.ibm.com

Tech/Tip: Generating a JWT using Liberty's as an example OPID provider

The Liberty server authorization server's XML configuration

```
<!--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="distributed_User1" 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('distributed_User1'))
     REGISTRY(NAME('*')) WITHLABEL('zCEE JWT distributedUser1')
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')
```

mitchj@us.ibm.com

Tech/Tip: RACMAP Command Summary

```
RACMAP ID(USER1) MAP USERDIDFILTER(NAME('distuser1'))
     REGISTRY(NAME('*')) WITHLABEL('zCEE token user1')
RACMAP ID(USER1) MAP USERDIDFILTER(NAME('distribute_User1'))
     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('USER X500 DN’)

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

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

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

Label: zCEE user1
Distributed Identity User Name Filter:
  >distributed_User1<
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:
  >*<
```

mitchj@us.ibm.com

Slide 106

Liberty OpenID Client identity mapping configuration attributes

Decoded EDIT THE PAYLOAD AND SECRET

HEADER: ALGORITHM & TOKEN TYPE

```
{
  "kid": "kvjtqdLMjOTWiJrjOr73fu2MMt-FjiQrxUOYBzJLR4o",
  "alg": "RS256"
}
```

PAYLOAD: DATA

```
{
  "sub": "auser",
  "token_type": "Bearer",
  "scope": [
    "openid",
    "profile",
    "email"
  ],
  "azp": "rpSsl",
  "iss": "https://wg31.washington.ibm.com:26213
/oidc/endpoint/OP",
  "aud": "myZcee",
  "exp": 1646761228,
  "iat": 1646760928,
  "realmName": "zCEERealm",
  "uniqueSecurityName": "auser"
}
```

```
<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, using IDIDMAP resources and the RACMAP command.

```
<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://wg31.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 (SAF) is not used to create the user subject.

mitchj@us.ibm.com

Liberty 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"
    httpsRequired="true"
    authFilterRef="jwkAuthFilter"
    inboundPropagation="required"
    signatureAlgorithm="RS256"
    userIdentifier="sub"
    mapIdentityToRegistryUser="true"
    issuerIdentifier=https://wg31.washington.ibm.com:26213/oidc/endpoint/OP
    disableLtpaCookie="true"
    audiences="myZcee"
    tokenReuse="true"
    jwkEndpointUrl=https://wg31.washington.ibm.com:26213/oidc/endpoint/OP/jwk
    jwkClientId="jwtClient"
    jwkSecret="jwtSecret"/>
  </openidConnectClient>
```

mitchj@us.ibm.com

JWT used in scenario – putting it all together

```
{
  "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

mitchj@us.ibm.com

Configuring authentication with JWT

Liberty 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="false"
    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

mitchj@us.ibm.com

Using authorization filters

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"/>
<openidConnectClient id="RPsslG" . . . authFilterRef= "API Gateway"/>
<openidConnectClient id="RPsslURL" . . . authFilterRef= "URLFilter"/>
<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.

mitchj@us.ibm.com

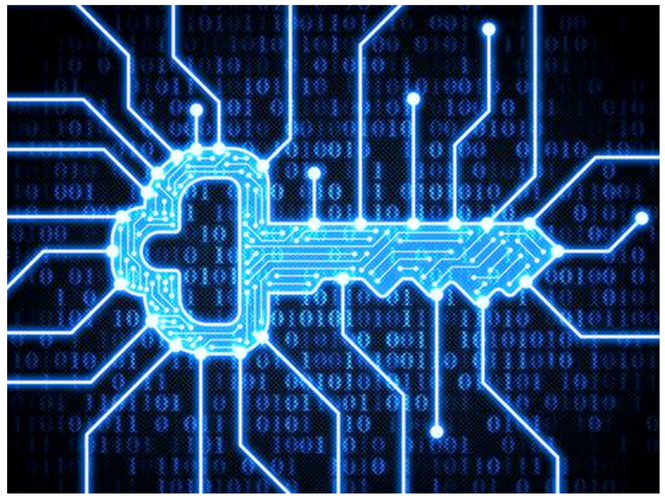
General security terms or considerations

Security involves

- Identifying who or what is requesting access (**Authentication)**
  - Basic Authentication
  - Mutual Authentication using Transport Layer Security (TLS), formerly known as SSL
  - Third Party Tokens

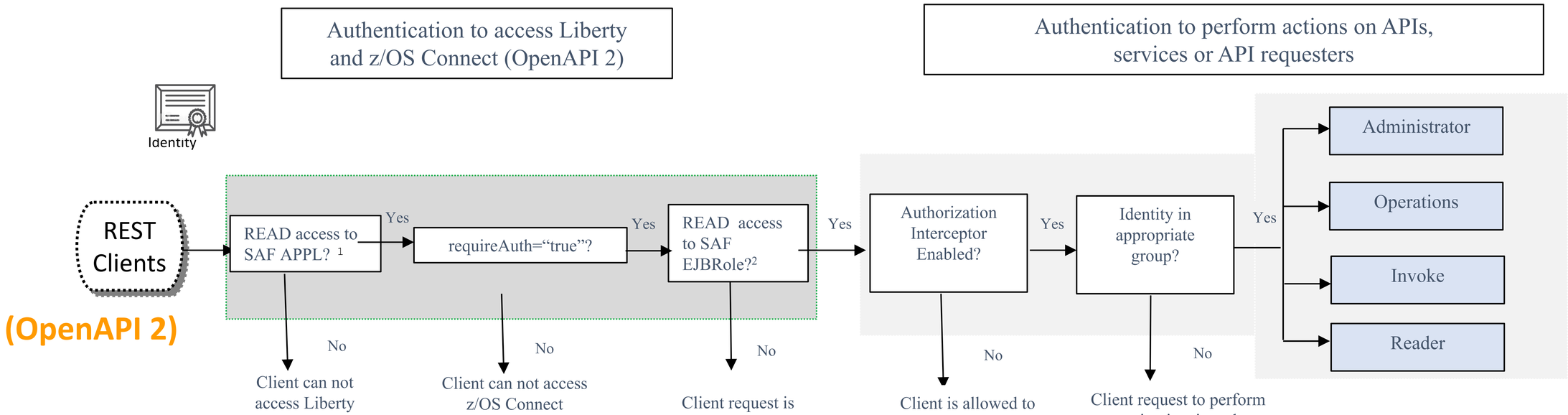
- Ensuring that the message has not been altered in transit (**Data Integrity)** and ensuring the confidentiality of the message in transit (**Encryption)**
  - TLS (encrypting messages and using a digital signature)

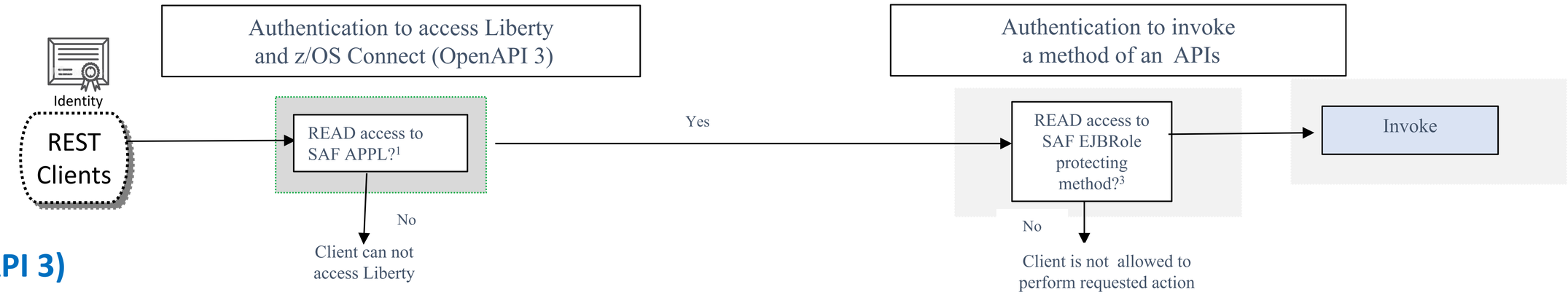
- Controlling access (**Authorization)**
  - Is the authenticated identity authorized to access to z/OS Connect
  - Is the authenticated identity authorized to access a specific API, Services, etc.



mitchj@us.ibm.com

Authorization security flow within z/OS Connect





(OpenAPI 3)

[1]RDEFINE APPL *profilePrefix*
[2]RDEFINE EJBROLE *profilePrefix*.zos.connect.access.roles.zosConnectAccess
[3]REDFINE EJBROLE *profilePrefix.resourceName.role*

mitchj@us.ibm.com

z/OS Connect Authorization Functions (OpenAPI 2)

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 ignored. 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).

mitchj@us.ibm.com

z/OS Connect Authorization Levels (OpenAPI 2)

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 URIs 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.

mitchj@us.ibm.com

z/OS Connect RESTful Administrative APIs Security (OpenAPI 2)

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="SYSPGRP " globalOperationsGroup="GBLOPERS"
          globalInvokeGroup="GBLINVKE" globalReaderGroup="GBLRDR"/>
```

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.

```
<zosConnectAPI 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"/
```

mitchj@us.ibm.com

Interceptor - server XML example (OpenAPI 2)

```
<zosconnect_zosConnectManager
         globalInterceptorsRef="interceptorList_g"
         globalAdminGroup="SYSPGRP"
         globalOperationsGroup="GBLOPERS"
         globalInvokeGroup="GBLINVKE"
         globalReaderGroup="GBLRDR"/>

<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 SYSPGRP OMVS(AUTOGID)*
ADDGROUP GBLINVKE OMVS(AUTOGID) *
CONNECT FRED GROUP(SYSPGRP)
CONNECT USER1 GROUP(GBLINVKE)
```

Global interceptor list – authorization interceptor only

Alternative interceptor list – authorization and audit interceptors

This avoids duplication of interceptors

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

*RDEFINE FACILITY BPX.NEXT.USER APPLDATA('2001/201')

mitchj@us.ibm.com

Tech/Tip: Server XML example – combining TLS/AUTH interceptor (OpenAPI 2)

```
 <zosconnect_zosConnectManager
           requireAuth="true"
           requireSecure="true"
           globalInterceptorsRef="interceptorList_g"
           globalAdminGroup="SYSPGRP"
           globalOperationsGroup="GBLOPERS"
           globalInvokeGroup="GBLINVKE"
           globalReaderGroup="GBLRDR"/>

<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 (when an AT-TLS outbound policy is not available). 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

mitchj@us.ibm.com

Example of z/OS Connect Authorization Levels (Open API 2) (this config has issues)

```
<zosconnect_zosConnectManager
         globalInterceptorsRef="interceptorList_g"
         globalAdminGroup="SYSPGRP" globalOperationsGroup="GBLOPERS"
         globalInvokeGroup="GBLINVKE" globalReaderGroup="GBLRDR"/>

<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>
```

This works as you expect once the artifacts are deployed but:

- Only members of groups SYSPGRP, GBLOPERS or GBLRDR can connect to a z/OS server from the API toolkit (the tooling attempts a GET request for a list of all deployed services and APIs).

- Only members of groups SYSPGRP or GBLOPERS can deploy new z/OS Connect API, service or API requester artifacts (POST access for operations is not available until after the artifact is deployed)

Tech-Tip: When groups are specified for zosConnectAPI, service, or apiRequester configuration elements, the global groups are ignored for certain functions. Other functions, e.g., deploy new artifact, get a list or service statistics, only use the global group membership.

mitchj@us.ibm.com

mitchj@us.ibm.com