

IBM z/OS Connect EE V3.0

# Customization - CICS Security



*Lab Version Date: May 8, 2020*

## Table of Contents

<b>Overview .....</b>	<b>4</b>
<b><i>Enabling Identity Propagation to CICS.....</i></b>	<b>5</b>
<i>Test with identity propagation not enabled.....</i>	<i>5</i>
<i>Test with identity propagation enabled .....</i>	<i>7</i>
<i>Summary .....</i>	<i>10</i>
<b><i>Enabling CICS TLS security to a zCEE Server .....</i></b>	<b>11</b>
<i>Review the Loan Application .....</i>	<i>11</i>
<i>Creating CICS SAF resources .....</i>	<i>16</i>
<i>Enabling CICS TLS support .....</i>	<i>19</i>
<i>Test the TLS connection from CICS to the zCEE server.....</i>	<i>24</i>
<i>Summary .....</i>	<i>26</i>

**Important:** On the desktop there is a file named *Basic Configuration CopyPaste.txt*. This file contains commands and other text used in this workshop. Locate that file and open it. Use the copy-and-paste function (**Ctrl-C** and **Ctrl-V**) to enter commands or text. It will save time and help avoid typo errors. As a reminder text that appears in this file will be highlighted in yellow.

## ***General Exercise Information and Guidelines***

- ✓ This exercise requires the completion of the *zCEE Basic Configuration* and *zCEE Basic Security Configurations* exercises before it can be performed.
- ✓ This exercise requires using z/OS user identities *FRED* and *USER1*. The password for these users will be provided by the lab instructions.
- ✓ There are examples of *server.xml* scattered through this exercise. Your *server.xml* may differ depending on which exercises have been previously performed. Be sure the red lines in these examples are either added or already present.
- ✓ The acronyms RACF (resource access control facility) and SAF (system authorization facility) are used in this exercise. RACF is the IBM security manager product whereas SAF is a generic term for any security manager product, e.g. ACF2 or Top Secret or RACF. An attempt has been to use SAF when referring to information appropriate for any SAF product and to use RACF when referring to specific RACF commands or examples.
- ✓ Any time you have any questions about the use of IBM z/OS Explorer, 3270 screens, features or tools, do not hesitate to ask the instructor for assistance.
- ✓ Text in **bold** and highlighted in **yellow** in this document should be available for copying and pasting in a file named *Basic Configuration CopyPaste* file on the desktop.
- ✓ Please note that there may be minor differences between the screen shots in this exercise versus what you see when performing this exercise. These differences should not impact the completion of this exercise.

## Overview

This exercise demonstrates use examples to show the steps required to enable security between a CICS region and a z/OS Connect EE (zCEE) server.

In part one of the exercise, identity propagation between a z/OS Connect EE (zCEE) server and CICS region is enabled and tested. Identity propagation in this context means that the identity authenticated by the zCEE server will be sent to the CICS region for subsequent CICS authorization checks. and in the second part TLS security between a CICS region and a z/OS Connect (zCEE) server will be demonstrated.

In part two of the exercise, TLS support will be added to a CICS region and the steps required to enable the exchange of certificates for mutual authentication between the CICS region and the zCEE server will be demonstrated. Finally, TLS security and mutual authentication from a zCEE server to outbound API provider will be demonstrated.

## Enabling Identity Propagation to CICS

When identity propagation is not required the only CICS connectivity related resource required is a TCPIPService resource, see an example below:

```

WG31
File Edit Settings View Communication Actions Window Help

OVERTYPE TO MODIFY                                CICS RELEASE = 0710
CEDA ALTER TCPIPService( IPIC                      )
TCPIPService   : IPIC
GRoup         : SYSPGRP
DEscription   ==>
Urm           ==> DFHISAIP
Portnumber    ==> 01491      1-65535
Status        ==> Open      Open | Closed
Protocol      ==> IPic      Http | Eci | User | IPic
Transaction   ==> CISS      0-32767
Backlog       ==> 00000
TSqprefix     :
Host          ==> ANY
(Mixed Case)  ==>
Ipaddress     ==> ANY
SPeciftcps    ==>
Socketclose   ==> No        No | 0-240000 (HHMMSS)
MAXPersist    ==> No        No | 0-65535
MAXDataLen    ==> 000032    3-524288

SYSID=CICS APPLID=CICS53Z

PF 1 HELP 2 COM 3 END          6 CRSR 7 SBH 8 SFH 9 MSG 10 SB 11 SF 12 CNCL
06/022
Connected to remote server/host wg31 using lu/pool TCP00112 and port 23

```

When the above resource definition is installed in a CICS region the CICS region will listen for inbound IPIC request on port 1491. When a request arrives, the CICS transaction CISS will be started and process the inbound IPIC request. The CICS authorization under which the CICS transaction will execute depends on the security configuration of the TCPIPService.

Now enable identity propagation between the zCEE server and a CICS region.

### Test with identity propagation not enabled

But first explore the CICS application involved and observe the behavior when identity propagation is not enabled.

The CICS application that will be executed uses the command below to save the CICS transaction identity in a response container and will return it to the client.

```

* Save current CICS userid
EXEC CICS ASSIGN USERID(USERID of Response-Container)
END-EXEC.

```

- \_\_\_1. Edit the *server.xml* configuration file for the *myServer* server, e.g. */var/zosconnect/servers/myServer/server.xml* and add an include for *shared.xml*, see below:

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

This will install some predefined services and APIs in the server.

```
<include location="${server.config.dir}/includes/safSecurity.xml"/>
<include location="${server.config.dir}/includes/ipic.xml"/>
<include location="${server.config.dir}/includes/keyringMutual.xml"/>
<include location="${server.config.dir}/includes/groupAccess.xml"/>
<include location="${server.config.dir}/includes/shared.xml"/>
```

- \_\_\_2. Stop and restart the server with MVS commands *P BAQSTRT* and *S BAQSTRT*.

**Tech-Tip:** MVS and JES2 commands can be entered from SDSF by enter a / (slash) on the command line followed by the command itself (e.g. /D T). The command results can be found in the system log. If a command is especially long, then simply enter a / (slash) to display a *SDSF – System Command Extension* panel where a command can span multiple lines. When an MVS command must be entered, the instructions in these exercises will indicate that the command is a MVS command and you may enter the command at the prompt by using the / (slash) prefix or using the *SDSF – System Command Extension* panel.

- \_\_\_3. Open a DOS command prompt and go to directory *c:/z/admin*.

- \_\_\_4. Enter the cURL command below:

```
curl -X get --cacert certauth.pem --cert fred.p12:secret --cert-type P12
https://wg31.washington.ibm.com:9443/cscvinc/employee/000100
```

Invoking this API will cause the CICS program to be executed and return the identity under which the CICS program executed in the JSON response property *USERID*, see below:

```
c:\z\admin>curl -X get --cacert certauth.pem --cert fred.p12:secret --cert-type P12
https://wg31.washington.ibm.com:9443/cscvinc/employee/000100
{"cscvincServiceOperationResponse":{"Container1":{"RESPONSE_CONTAINER":{"CEIBRESP2":0,
"FILE_AREA":{"STAT":"","ADDRX":"SURREY, ENGLAND","AMOUNT":"$0100.11",
"PHONE":"32156778","DATEX":"26 11 81","NUMB":"000100","COMMENT":"*****",
"NAME":"S. D. BORMAN"},"ACTION":"S","USERID":"CICSUSER","CEIBRESP":0}}}}
```

In this case identity propagation was not enable and no other identity was provided so the transaction executed under the CICS default user identity *CICSUSER*.

- \_\_\_5. Enter the cURL command below:

```
curl -X GET --cacert certauth.pem --cert user1.p12:secret --cert-type P12  
https://wg31.washington.ibm.com:9443/cscvinc/employee/000100
```

The CICS program will be executed and return the CICS transaction identity in the JSON response property USERID, see below:

```
c:\z\admin>curl -X get --cacert certauth.pem --cert user1.p12:secret --cert-type P12  
https://wg31.washington.ibm.com:9443/cscvinc/employee/000100  
{ "cscvincServiceOperationResponse": { "Container1": { "RESPONSE_CONTAINER": { "CEIBRESP2": 0,  
"FILEA_AREA": { "STAT": "", "ADDRX": "SURREY, ENGLAND", "AMOUNT": "$0100.11",  
"PHONE": "32156778", "DATEX": "26 11 81", "NUMB": "000100", "COMMENT": "*****",  
"NAME": "S. D. BORMAN" }, "ACTION": "S", "USERID": "CICSUSER", "CEIBRESP": 0 } } } }
```

Although a different client certificate was provided, identity propagation was not enabled, so again the CICS transaction executed under the CICS default user identity.

### ***Test with identity propagation enabled***

When no action is taken, CICS will auto installed an IPCONN resource when a TCPIPService is started. The template used to auto install an IPCONN resource does not enable identity propagation. Enabling identity propagation from a zCEE server to a CICS region requires overriding the default auto installed CICS IPCONN resource with an explicit IPCONN resource with identity propagation enabled for each TCPIPService.

1. The IPCONN resource below specifies **CSCVINC** for both the *APplid* and *Networkid* attributes. The values specified here must match corresponding values in a *zosconnect\_cicsIpicConnection* configuration element, see below. The value for *Tcpipservice* must match the name of the associated TCPIPService resource.

```
WG31
File Edit Settings View Communication Actions Window Help

OVERTYPE TO MODIFY                                CICS RELEASE = 0710
CEDA ALTER IPCONN( CSCVINC )
  IPconn      : CSCVINC
  Group       : SYSPGRP
  Description ==>
  IPIC CONNECTION IDENTIFIERS
  APplid      ==> CSCVINC
  Networkid   ==> CSCVINC
  Host        ==>
  (Mixed Case) ==>
  Port        ==> No                No | 1-65535
  Tcpipservice ==> CSCVINC
  HA          ==> No                No | Yes
  IPIC CONNECTION PROPERTIES
  Receivecount ==> 001              1-999
  SENDcount    ==> 000              0-999
  Queuelimit   ==> No               No | 0-9999
  MAXqtime     ==> No               No | 0-9999
+ OPERATIONAL PROPERTIES

SYSID=CICS APPLID=CICS53Z

PF 1 HELP 2 COM 3 END                6 CRSR 7 SBH 8 SFH 9 MSG 10 SB 11 SF 12 CNCL
MA A                                08/029
Connected to remote server/host wg31 using lu/pool TCP00103 and port 23
```

2. Identity propagation attributes are specified in the *Security* section of the IPCONN resource. In this case the *Linkauth* attribute was set to **Secuser**, *Userauth* attribute was set to **Identify** and the *IDprop* attribute was set to **Required** in advance.

```

WG31
File Edit Settings View Communication Actions Window Help

OVERTYPE TO MODIFY OR PRESS ENTER TO EXECUTE          CICS RELEASE = 0710
CEDA ALTER IPCONN( CSCVINC )
+ Autoconnect ==> No          No | Yes
  Inservice   ==> Yes        Yes | No
SECURITY
  Ssl         ==> No          No | Yes
  Certificate ==>              (Mixed Case)
  Ciphers     ==>
  (Mixed Case)
  Linkauth    ==> Secuser      Secuser | Certuser
  SECURITYNAME ==>
  Userauth    ==> Identify     Local | Identify | Verify | Defaultuser
  IDprop      ==> Required     Notallowed | Optional | Required
RECOVERY
  Xlnaction   ==> Keep         Keep | Force
MIRROR TASK PROPERTIES
  Mirrorlife  ==> Request      Request | Task | Uow
DEFINITION SIGNATURE
+ Definetime  : 02/16/20 13:04:07
                                           SYSID=CICS APPLID=CICS53Z

PF 1 HELP 2 COM 3 END          6 CRSR 7 SBH 8 SFH 9 MSG 10 SB 11 SF 12 CNCL
MA A                                04/022
Connected to remote server/host wg31 using lu/pool TCP00103 and port 23

```

**Tech Tip:** There will be at least one security check performed when CICS starts the mirror transaction. The security check will be for *READ* access to either transaction code *CSMI* (the CICS default mirror transaction) or the value of the transaction code specified in service's configuration for the *Transaction ID* attribute when the *Transaction ID Usage* attribute is set to *EIB\_AND\_MIRROR*

A SAF check will be performed with the identity propagated from z/OS Connect. But before this check, another SAF check may be performed using the *link identity*. The *link identity* is determined as follows. For a TLS connection, e.g. *LINKAUTH(CERTUSER)*, the *link identity* will be the local SAF identity mapped to the client certificate. For a non-TLS connection, e.g. *LINKAUTH(SECUSER)*, the *link identity* will be the value provided in the *SECURITYNAME* IPCONN attribute. If no value is provided in this attribute, the CICS default user identity will be used for the *link identity*.

If the *link identity* matches the SAF identity under which the CICS region is running, only the propagated identity is used for a SAF check for accessing to the mirror transaction. If the *link identity* does not match the SAF identity of the CICS region then a SAF check is also performed for the *link identity's* access to the mirror transaction.

Setting the *Userauth* attribute to *Identify* requires the client to provide the identity authenticated by the client.

Review the CICS documentation regarding the *IDprop* attribute. Behavior of this attributes depends on whether the zCEE server and the CICS region are in the same Sysplex or not.



- \_\_\_3. Edit the *server.xml* file again and replace the include for *ipic.xml* with an include for *ipicIDprop.xml*

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

Below is the updated *zosconnect\_cicsIpicConnection* configuration element in the *ipicIDProp.xml* file for the connection reference used by the API services in this API.

```
<zosconnect_cicsIpicConnection id="cscvinc"
  host="wg31.washington.ibm.com"
  zosConnectNetworkid="CSCVINC"
  zosConnectApplid="CSCVINC"
  port="1453"/>
```

Again, note the values for *zosConnectNetworkid* and *zosConnectApplid* must match the *Networkid* and *APplid* defined in the CICS IPCONN resource definition.

- \_\_\_4. Stop and restart the server with MVS commands ***P BAQSTRT*** and ***S BAQSTRT***.

**Tech Tip:** A restart is required other than refresh because the state of the connection between the zCEE server and CICS is being changed. A refresh would not work in this circumstance.

- \_\_\_5. At a DOS command prompt and change to directory *c:/z/admin*.

- \_\_\_6. Enter the cURL command below:

```
curl -X get --cacert certauth.pem --cert fred.p12:secret --cert-type P12
https://wg31.washington.ibm.com:9443/cscvinc/employee/000100
```

The CICS program will be executed and return the transaction identity under which the CICS program executed in the JSON response property *USERID*, see below:

```
c:\z\admin>curl -X get --cacert certauth.pem --cert fred.p12:secret --cert-type P12
https://wg31.washington.ibm.com:9443/cscvinc/employee/000100
{"cscvincServiceOperationResponse":{"Container1":{"RESPONSE_CONTAINER":{"CEIBRESP2":0,
"FILE_AREA":{"STAT":"","ADDRX":"SURREY, ENGLAND","AMOUNT":"$0100.11",
"PHONE":"32156778","DATEX":"26 11 81","NUMB":"000100","COMMENT":"*****",
"NAME":"S. D. BORMAN"},"ACTION":"S","USERID":"FRED","CEIBRESP":0}}}}
```

In this case the identity (FRED) associated with the client certificate provided by the cURL command in file *fred.p12* is propagated to CICS.

\_\_\_7. Enter the cURL command below:

```
curl -X GET --cacert certauth.pem --cert user1.p12:secret --cert-type P12  
https://wg31.washington.ibm.com:9443/cscvinc/employee/000100
```

The CICS program will be executed and return the CICS transaction identity in the JSON response property USERID, see below:

```
c:\z\admin>curl -X get --cacert certauth.pem --cert fred.p12:secret --cert-type P12  
https://wg31.washington.ibm.com:9443/cscvinc/employee/000100  
{ "cscvincServiceOperationResponse": { "Container1": { "RESPONSE_CONTAINER": { "CEIBRESP2": 0,  
"FILEA_AREA": { "STAT": " ", "ADDRX": "SURREY, ENGLAND", "AMOUNT": "$0100.11",  
"PHONE": "32156778", "DATEX": "26 11 81", "NUMB": "000100", "COMMENT": "*****", "NAME": "S.  
D. BORMAN" }, "ACTION": "S", "USERID": "USER1", "CEIBRESP": 0 } } } }
```

In this the case the identity (USER1) associated with the client certificate provided by the cURL command in file *user1.p12* is propagated to CICS.

## Summary

In this section, a simple REST client has been used to invoke an API which accesses a CICS region to execute an application. The application returns the identity under which the task is executing to the client. Initially, the REST client accesses the CICS program without propagating the identity authenticated by the zCEE server. Changes were made in CICS and the server.xml so that the identity associated with the client certificate (fred.p12 and user1.p12) provided by the cURL command were propagated to CICS.

## Enabling CICS TLS security to a zCEE Server

Adding TLS support to a CICS regions requires the creation of a key ring belonging to the identity under which the CICS region is executing (look for message IEF695I in the CICS regions JES messages). This key ring contains the personal and all the certificate authority certificates that will be used during TLS handshakes. The creation of the key ring and the connection of certificates to the key ring are done using the RACDCERT RACF command. Enabling TLS support in the CICS region itself will be done by adding a few SIT overrides. This will require a restart of the CICS region.

### Review the Loan Application

The *CICS Miniloan Application* is a CICS application that uses BMS to display screens in a 3270-terminal session. The application is started by entering CICS transaction **APIR** in a CICS terminal session. This will display the screen displayed below.

The application use rules for determining whether a loan request is approved or rejected based on the information entered using the above screen.

The rules for rejecting a loan can be for any one of the following:

- If the credit score of the borrower is less than 300.
- If the yearly repayment amount is more than 30% of the borrower's income.
- If the income of the borrower is less than \$24,000.

- If the age of the borrower is more than 65 years.
- The loan amount is more than \$1,000,000.

N.B., Most of the 3270 screen shots in the remainder of this exercise are shown in reverse video simply for printing purposes.

Below is an example of a display when all the rules are used to reject a loan request.

The screenshot shows a CICS Miniloan Application window titled 'WG31'. The window has a menu bar with 'File', 'Edit', 'Settings', 'View', 'Communication', 'Actions', 'Window', and 'Help'. The main display area shows the following information:

Borrower Information		Loan Information	
Name	JOHN	Loan Amount	10000000
Age	150	Yearly Repayment	50000
Credit Score	100	Interest Rate	00005
Yearly Income	24000	Effective Date	20200101 YYYYMMDD
CICS Identity	CICSUSER		
Approval Status:	Loan not approved		

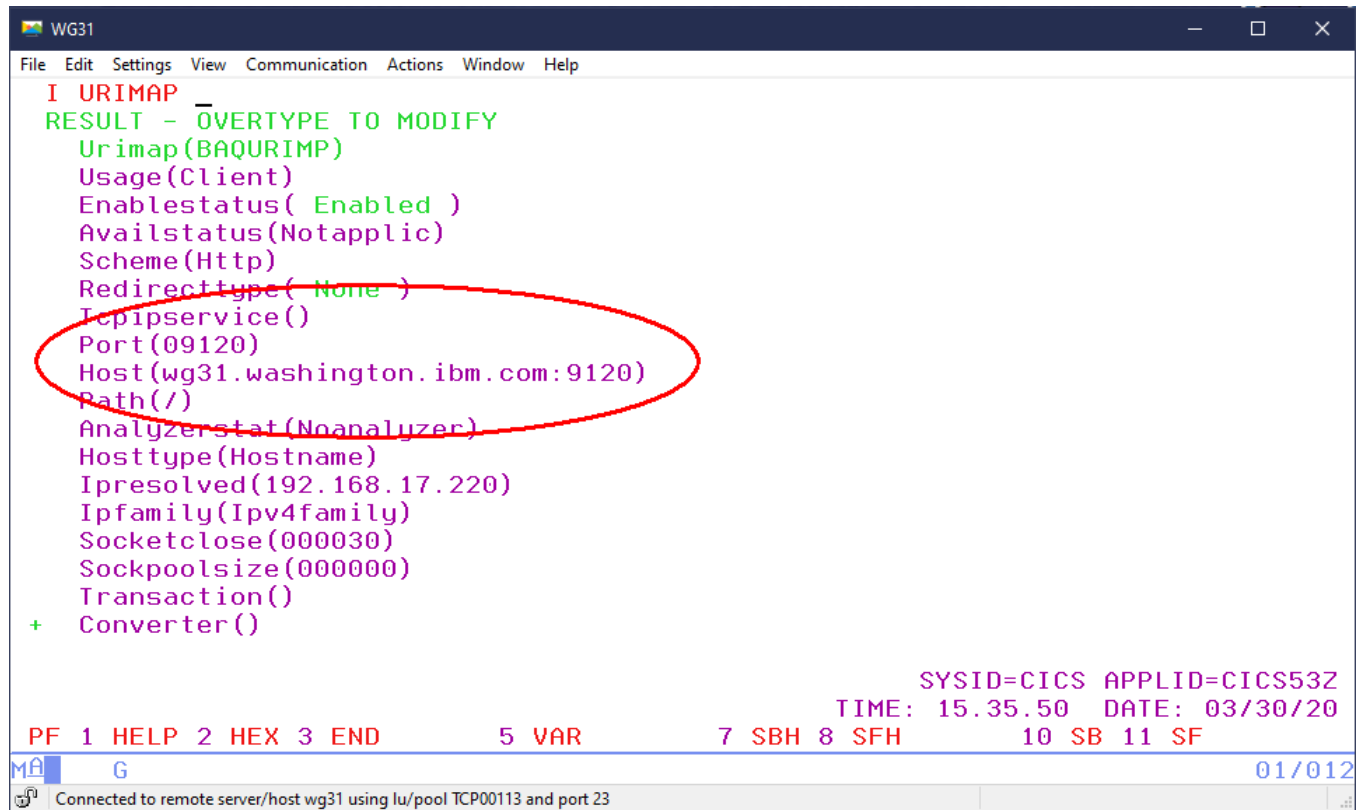
Below the information, the following reasons for rejection are listed:

- The age exceeds the maximum.
- The loan cannot exceed 10000000
- Too big Debt-To-Income ratio
- Credit score below 200
- The yearly income is lower than the basic request

At the bottom of the window, there is a status bar showing 'F3=Exit', a cursor position 'D', a date '05/014', and a connection status 'Connected to remote server/host wg31 using lu/pool TCP00111 and port 23'.

The source for this application can be found in member *LOANAPIR* in data set *USER1.ZCEE.SOURCE*.

The host, port and protocol (Scheme) CICS will use to communicate with the z/OS Connect server is configured in a *BAQURIMAP* CICS URIMAP resource.



```

I URIMAP
RESULT - OVERTYPE TO MODIFY
Uri(BAQURIMP)
Usage(Client)
Enablestatus( Enabled )
Availstatus(Notapplic)
Scheme(Http)
Redirecttype( None )
Tcpipservice()
Port(09120)
Host(wg31.washington.ibm.com:9120)
Path(/)
Analyzerstat(Noanalyzer)
Hosttype(Hostname)
Ipresolved(192.168.17.220)
Ipfamily(Ipv4family)
Socketclose(000030)
Sockpoolsize(000000)
Transaction()
+ Converter()

SYSID=CICS APPLID=CICS53Z
TIME: 15.35.50 DATE: 03/30/20
PF 1 HELP 2 HEX 3 END 5 VAR 7 SBH 8 SFH 10 SB 11 SF
G 01/012
Connected to remote server/host wg31 using lu/pool TCP00113 and port 23

```

If interested, use CEDF to trace through the flow of this program and eventually you will see a screen like the below.

```

WG31
File Edit Settings View Communication Actions Window Help
TRANSACTION: APPIR PROGRAM: LOANAPIR TASK: 0000284 APPLID: CICS53Z DISPLAY: 00
STATUS: ABOUT TO EXECUTE COMMAND
EXEC CICS WEB WRITE
  HTTPHEADER ('zCEE-APIRequester-Metadata')
  NAMELENGTH (26)
  VALUE ('zCEE-APIRequester-Path=%2Fminiloancics%2Floan&zCEE-APIRequester'...)
  VALUELENGTH (113)
  SESSTOKEN ('.....')
  NOHANDLE

LINE: 00090500 EIBFN=X'380E'

ENTER: CONTINUE
PF1 : UNDEFINED PF2 : SWITCH HEX/CHAR PF3 : UNDEFINED
PF4 : SUPPRESS DISPLAYS PF5 : WORKING STORAGE PF6 : USER DISPLAY
PF7 : SCROLL BACK PF8 : SCROLL FORWARD PF9 : STOP CONDITIONS
PF10: PREVIOUS DISPLAY PF11: EIB DISPLAY PF12: ABEND USER TASK

01/001
Connected to remote server/host wg31 using lu/pool TCP00115 and port 23

```

You can use EDF features to review the details of the request (see an example below).

```

WG31
File Edit Settings View Communication Actions Window Help
TRANSACTION: APPIR PROGRAM: LOANAPIR TASK: 0000284 APPLID: CICS53Z DISPLAY: 00
ADDRESS: 1570CB10
1570CB10 000000 A9C3C5C5 60C19789 D98598A4 85A2A385 zCEE-APIRequeste
1570CB20 000010 9960D781 A3887E6C F2C69489 95899396 r-Path=%2Fminilo
1570CB30 000020 81958389 83A26CF2 C6939681 9550A9C3 ancics%2Floan&zC
1570CB40 000030 C5C560C1 9789D985 98A485A2 A3859960 EE-APIRequester-
1570CB50 000040 D485A388 96847ED7 D6E2E350 A9C3C5C5 Method=POST&zCEE
1570CB60 000050 60C19789 D98598A4 85A2A385 9960C1A2 -APIRequester-As
1570CB70 000060 A28599A3 8584C984 7EC3C9C3 E2E4E2C5 sertedId=CICSUSE
1570CB80 000070 D9000000 00000000 00000000 00000000 R.....
1570CB90 000080 00000000 00000000 00000000 00000000 .....
1570CBA0 000090 00000000 1570D1B0 1570D1ED 1570D22A .....J...J...K.
1570CBB0 0000A0 1570D267 1570D2A4 1570D2E1 1570D31E ..K...Ku..K...L.
1570CBC0 0000B0 1570D35B 00000000 00000000 00000000 ..L$.....
1570CBD0 0000C0 00000000 00000000 00000000 00000000 .....
1570CBE0 0000D0 00000000 1570CBF8 000001E8 0000003D .....8...Y....
1570CBF0 0000E0 1570C5E0 00000828 00000000 00000000 ..E\.....
1570CC00 0000F0 00000000 00000000 00000000 00000000 .....

ENTER: CURRENT DISPLAY
PF1 : UNDEFINED PF2 : BROWSE TEMP STORAGE PF3 : UNDEFINED
PF4 : EIB DISPLAY PF5 : INVOKE CECI PF6 : USER DISPLAY
PF7 : SCROLL BACK HALF PF8 : SCROLL FORWARD HALF PF9 : UNDEFINED
PF10: SCROLL BACK FULL PF11: SCROLL FORWARD FULL PF12: REMEMBER DISPLAY

02/012
Connected to remote server/host wg31 using lu/pool TCP00115 and port 23

```

```

WG31
File Edit Settings View Communication Actions Window Help
TRANSACTION: APIR PROGRAM: LOANAPIR TASK: 0000323 APPLID: CICS53Z DISPLAY: 00
STATUS: ABOUT TO EXECUTE COMMAND
EXEC CICS WEB CONVERSE
  SESSTOKEN ('.....')
  PATH ('/zosConnect/apiRequesters/miniloancics_1.0.0')
  PATHLENGTH (44)
  METHOD (749)
  FROM ('.....JOHN ..... 100..... '...')
  FROMLENGTH (142)
  MEDIATYPE ('application/octet-stream.....')
  SET (X'168D90B0') AT X'1570A73C'
  TOLENGTH (6338)
  STATUSCODE (0)
  STATUSTEXT ('.....v{.y.n.....y...j.....yH..t.....t.nzT&n...n.....'...')
  STATUSLEN (256)
  NOHANDLE
LINE:00037100 EIBFN=X'381C'

ENTER: CONTINUE
PF1 : UNDEFINED PF2 : SWITCH HEX/CHAR PF3 : UNDEFINED
PF4 : SUPPRESS DISPLAYS PF5 : WORKING STORAGE PF6 : USER DISPLAY
PF7 : SCROLL BACK PF8 : SCROLL FORWARD PF9 : STOP CONDITIONS
PF10: PREVIOUS DISPLAY PF11: EIB DISPLAY PF12: ABEND USER TASK

MA D 01/001
Connected to remote server/host wg31 using lu/pool TCP00111 and port 23

```

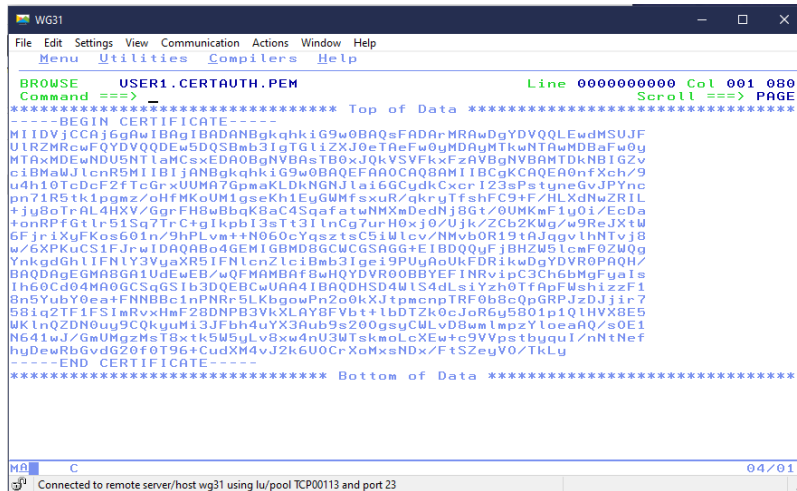
**Tech-Tip:** The EXEC CICS WEB CONVERSE command is the point where the request flows from CICS to the zCEE server.

This exercise will begin going through the steps to enable TLS communications between CICS and the z/OS Connect server.

## Creating CICS SAF resources

In this scenario, the certificate authority used by organization has provided two certificates in advance. Both certificates currently reside in MVS data sets. One certificate is a personal certificate that will be used by CICS. This certificate was exported with a private key and will be used as a server certificate when CICS is acting a server endpoint or as a client or personal certificate when CICS is the client endpoint. The other certificate is the public certificate authority certificate that was used the sign the certificate assigned to CICS (this certificate only contains the public key).

- \_\_\_1. Browse data set *USER1.CERTAUTH.PEM* and you should see the provided certificate authority exported certificate.



- \_\_\_2. Browse data set *USER1.CICS.P12* and you should see the CA provided certificate for use as CICS personal certificate. Since it contains a private key, it is password protected.





- \_\_\_3. Browse data set *USER1.ZCEE30.CNTL*. You should see the members in that data set.
- \_\_\_4. Next, browse member **CICSRCF1**. You should see the RACF commands below. Submit the job for execution.

```
racdcert CERTAUTH withlabel('CICS CA') -  
  add('USER1.CICSCA.PEM')  
  
racdcert id(START1) withlabel('CICS Client Cert') -  
  add('USER1.CICS.P12') password('secret')  
  
setropts raclist(digtcert) refresh  
  
racdcert certauth list(label('CICS CA'))  
  
racdcert id(START1) list(label('CICS Client Cert'))
```

**Tech-Tip:** In summary, these commands add the certificates that were provided by a certificate authority (currently stored in MVS in data sets) and installs them in RACF.

The certificate authority certificate contains the public key of the CA and is used to verify personal certificates. This certificate only contains the public key.

The personal certificate is associated with RACF identity START1. This certificate is mapped to identity START and contains a private key and therefore a password must be provided in order to access the private key when the certificate is added to RACF.

The in-storage profile for digital certificates resources are refreshed and the just added certificates are displayed.

\_\_\_5. Next, browse member **CICSRCF2**. You should see the RACF commands below. Submit the job for execution.

```
/* Create CICS key ring and connect CA and personal certificates */
racdcert id(start1) addring(CICS.KeyRing)

racdcert id(start1) connect(ring(CICS.KeyRing) +
                           label('CICS CA') certauth usage(certauth))

/* Connect default personal certificate */
racdcert id(start1) connect(ring(CICS.KeyRing) +
                           label('CICS Client Cert') default

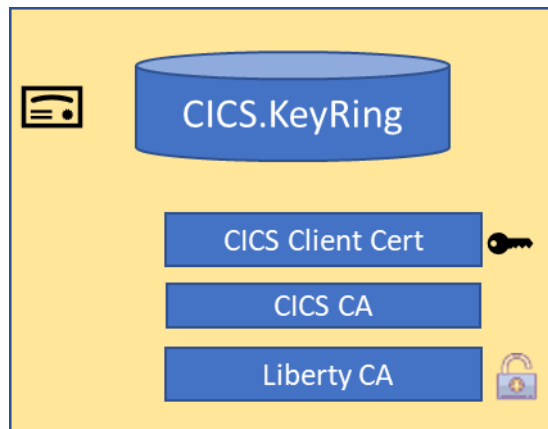
racdcert id(start1) connect(ring(CICS.KeyRing) +
                           label('Liberty CA') certauth usage(certauth))

setropts raclist(digtring,digtnmap) refresh

connect start1 group(zceeusrs)

connect start1 group(gminvoke)
```

Below is visual representation of the key ring just created.



**Tech-Tip:** These commands are creating a key ring belonging to the identity under which the CICS region is executing. Then the personal certificate and CA certificate used to sign the CICS personal certificate are connected to this key ring. Finally, the certificate authority certificate used to sign the server certificate that will be sent from the zCEE server during a handshake is connected to the key ring. The latter is done in order to verify the server certificate sent from the zCEE server

The in-storage profile for digital certificates resources is refreshed and the just added certificates are displayed.

The identity mapped to this client certificate needs access to the zCEE server (via group ZCEEUSRS) and access for invoking APIs and API requesters (via group GMINVOKE).

## Enabling CICS TLS support

The next step is to update the CICS's regions initialization parameters to add support for TLS. This involves adding system initialization table (SIT) parameters.

- \_\_\_1. Edit member *CICSZ* in data set *CICSTS.CICSZ.SYSIN*.
- \_\_\_2. Add the following system initialization overrides anywhere before the *.END* line.

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

- \_\_\_3. Use the CEDA transaction to change the *SCHEME* and *PORT* of the *BAQURIMP* Urimap resource from *HTTP* to *HTTPS* and from *9120* to *9443*, see below.

### **CEDA EX G(SYSPGRP) URIMAP(BAQURIMP)**

```

WG31
File Edit Settings View Communication Actions Window Help

OVERTYPE TO MODIFY                                CICS RELEASE = 0710
CEDA Alter Urimap( BAQURIMP )
  Urimap      : BAQURIMP
  Group       : SYSPGRP
  Description ==> URIMAP for z/OS Connect EE server
  Status      ==> Enabled           Enabled | Disabled
  USAge       ==> Client           Server | Client | Pipeline | Atom
                                           | Jvmserver

  UNIVERSAL RESOURCE IDENTIFIER
  Scheme      ==> HTTPS             HTTP | HTTPS
  Port        ==> 09443             No | 1-65535
  HOST        ==> wg31.washington.ibm.com
  Path        ==> /
  (Mixed Case) ==>
  ==>
  ==>
+ OUTBOUND CONNECTION POOLING

                                           SYSID=CICS APPLID=CICS53Z

PF 1 HELP 2 COM 3 END                      6 CRSR 7 SBH 8 SFH 9 MSG 10 SB 11 SF 12 CNCL
MA D                                         13/022
Connected to remote server/host wg31 using lu/pool TCP00111 and port 23

```

- \_\_\_4. Shutdown CICS with a **CEMT PE SHUTDOWN** command, using either an MVS modify command or in a CICS terminal session.
- \_\_\_5. When CICS is no longer active restart the region with MVS command **S CICSZ**.

- \_\_\_6. The next step is to connect the CICS certificate authority to the key ring used for inbound TLS connection to the zCEE server. Browse member **CICSRCF3**. You should see the RACF commands below. Submit the job for execution.

```
/* Connect the CICS CA certificate to the Liberty inbound key ring */
racdcert id(atsserv) connect(ring(Liberty.KeyRing) +
                             label('CICS CA') certauth usage(certauth))

setropts raclist(digtring) refresh
```

**Tech-Tip:** These commands are connecting the certificate authority certificate to the zCEE server key ring used for inbound TLS handshakes. If this certificate was not connected to this keyring, the personal certificate sent by CICS during mutual authentication would be rejected.

- \_\_\_7. Edit the *server.xml* configuration file for the *myServer* server, e.g. */var/zosconnect/servers/myServer/server.xml* and add an include for *apiRequesterHTTPS.xml*, see below:

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

```
<include location="${server.config.dir}/includes/safSecurity.xml"/>
<include location="${server.config.dir}/includes/ipicIDProp.xml"/>
<include location="${server.config.dir}/includes/keyringMutual.xml"/>
<include location="${server.config.dir}/includes/groupAccess.xml"/>
<include location="${server.config.dir}/includes/apiRequesterHTTPS.xml"/>
<include location="${server.config.dir}/includes/shared.xml"/>
```

This will change the *zosconnect\_endpointConnections* from HTTP to HTTPS.

8. Browse member **ZCEERCF7**. You should see the RACF commands below. Submit the job for execution if this job has not been previously submitted in another exercise

```
/* Create personal certificate for zCEE outbound client request */
racdcert id(libserv) gencert subjectsdn(cn('zCEE Client Cert') +
ou('ATS') o('IBM')) withlabel('zCEE Client Cert') signwith(certauth
label('zCEE CA')) notafter(date(2021/12/31))

/* Create zCEE outbound key ring and connect certificates */
racdcert id(libserv) addring(zCEE.KeyRing)

racdcert id(libserv) connect(ring(zCEE.KeyRing) +
label('zCEE CA') certauth usage(certauth))

racdcert id(libserv) connect(ring(zCEE.KeyRing) +
label('Liberty CA') certauth usage(certauth))

/* Connect CA certificate to Liberty inbound key ring */
racdcert id(libserv) connect(ring(Liberty.KeyRing) +
label('zCEE CA') certauth usage(certauth))

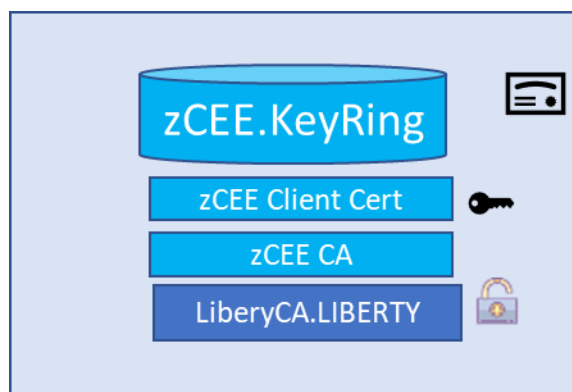
/* Connect default personal certificate */
racdcert id(libserv) connect(ring(zCEE.KeyRing) +
label('zCEE Client Cert') default)

setr raclist(digtcert digtring) refresh

connect libserv group(zceeusrs)
connect libserv group(gminvoke)
```

**Tech-Tip:** These commands create the key ring used for outbound TLS request. The personal certificate used for TLS handshakes is connected to this key ring as well as the CA certificate for validating the server certificate sent by the API provider and the CA certificate used to sign the zCEE server personal certificate. Since in this case the API provider is a zCEE server, the identity mapped to this server's personal certificate needs access to the zCEE server (via group ZCEEUSRS) and access for invoking APIs and API requesters (via group GMINVOKE).

Below is visual representation of the key ring just created.



- \_\_\_8. Submit the job for execution by entering ISPF command ***SUBMIT*** on the command line. The job should complete with a return code of zero. Review the output is desired.
- \_\_\_9. Edit the *server.xml* configuration file for the *myServer* server, e.g. */var/zosconnect/servers/myServer/server.xml* and change the include for file *keyringMutual.xml* to an include of file *keyringOutBoundMutual.xml*, see below:

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

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

```
<!-- 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:///Keyring.LIBERTY"
  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" />
```

- \_\_\_10. Enter MVS commands *P BAQSRT and S BASSTRT* to refresh the zCEE server's runtime configuration.

**Tech-Tip:** Updates to keyrings could have been refreshed in the server by using this command :

*F BAQSTRT,REFRESH,KEYSTORE*

This would have dynamically made the information about CICS CA certificate available in the zCEE runtime.

## Test the TLS connection from CICS to the zCEE server

1. In a CICS terminal session, start the **APIR** transaction and enter the information for a loan request. When you press enter, the request will flow to CICS asserting the CICS default identity. This identity does not have the authority to invoke the API requester and the message shown in the screen below should appear.

WG31

File Edit Settings View Communication Actions Window Help

CICS Miniloan Application

Borrower Information		Loan Information	
Name	JOHN	Loan Amount	100000
Age	100	Yearly Repayment	100000
Credit Score	100	Interest Rate	00005
Yearly Income	20202	Effective Date	20200101 YYYYMMDD
CICS Identity			
Approval Status:			
000000403			
{ "errorMessage": "BAQR7115E: The asserted user ID CICSUSER is not authorized to invoke the API requester on resource /zos Connect/apiRequesters/miniloancics_1.0.0." }			

F3=Exit

MA B 05/014

Connected to remote server/host wg31 using lu/pool TCP00117 and port 23



- \_\_\_2. Use the CICS *CESN* transaction to authenticate to CICS as *FRED* and retry the *LOAN* transaction. This time, identity FRED is flowed to the zCEE server. This identity has authority to invoke the API requester.

The screenshot shows a CICS Miniloan Application window with the following data:

Borrower Information		Loan Information	
Name	JOHN	Loan Amount	100000
Age	100	Yearly Repayment	100000
Credit Score	100	Interest Rate	00005
Yearly Income	10101	Effective Date	20200101 YYYYMMDD
CICS Identity	LIBSERV		
Approval Status: Loan not approved			

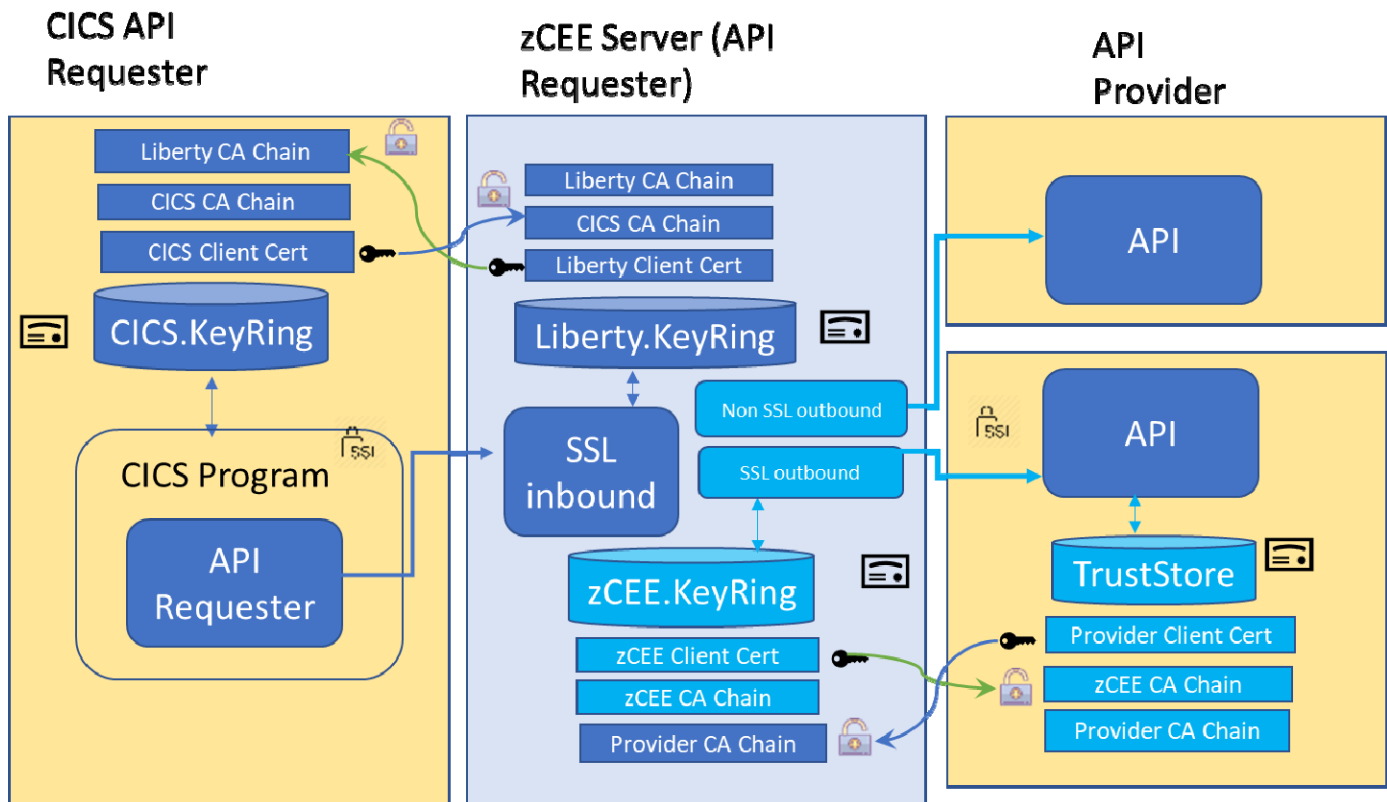
Below the table, the following error messages are displayed:

- The age exceeds the maximum.
- Too big Debt-To-Income ratio
- Credit score below 200
- The yearly income is lower than the basic request

At the bottom of the window, the text "F3=Exit" is visible. The status bar at the bottom indicates "Connected to remote server/host wg31 using lu/pool TCP00117 and port 23".

The results raise a question. Why is the CICS identity *LIBSERV* and not *FRED*? For an explanation see below.

The diagram below shows the identity that flows the CICS API requester. Since the connection between CICS and the zCEE server is protected by TLS, the identity under which the CICS task is running can be asserted from CICS to the zCEE server for an authorization check for executing the API requester. This identity does not flow to the API provider. Since it is a TLS connection and mutual authentication is enabled between the zCEE server, the authenticated identity is the RACF identity mapped to the zCEE server client certificate (LIBSERV). This identity is used for authorization checks and, in this case, propagated to the CICS application.



## Summary

The loan application was used to test accessing a z/OS Connect server using TLS and asserting the CICS identity to the zCEE server for further authorization.