

IBM Sterling Connect:Direct Secure Plus



Connect:Direct Secure+ Common Tasks

Version 6.x

Contents

IBM Sterling Connect:Direct Secure+ 4

How Connect:Direct Establishes a Secure Connection	4
Connect:Direct Secure+ Session (Handshake) Flow Grid	6
Connect:Direct Secure+ Flow Grid Explanation . . .	7
SSL Certificate Validation Simplified	9

Using Client Authentication 11

What is Client Authentication?	11
When Should Client Authentication Be Used? . .	12

Creating and Managing Certificates 13

Distinguished Names in an X.509 Certificate. . . .	13
Using Unix System Service (USS) gskkyman (key database for C:D z/OS)	14
Building a Key Database Using gskkyman. . . .	14
Building Certificates Using gskkyman	16
Export Your Certificate from Your Key Database	20
Import a Remote Certificate into Your Key Database.	22
Build a Certificate Signing Request (CSR)	25
Import CA Signed Certificate From Certificate Signing Request (CSR).	28
Export a Remote Certificate from gskkyman . .	30
Copying z/OS Certificates from gskkyman . . .	32
Copy and Paste Certificate to Pre-Allocated File.	32
Copy Certificate as HFS file Using Connect:Direct	34
Copy and Paste Using Open Editor (in USS). .	35
ISHELL (IS) – OPENMVS ISPF SHELL	35
EDIT (OE) – CREATE OR CHANGE SOURCE DATA.	36
Loading Private Key (For Use as Site Certificate) into gskkyman.	39
Exporting Private Key from gskkyman to Use as Site Certificate	43
Using IBM Key Manager (IKEYMAN)	47
Creating a Key Database (Keystore) using IKEYMAN	47
Creating Certificates using IKEYMAN	49
Building a Certificate Signing Request (CSR) Using IKEYMAN.	51

Creating the Certificate Signing Request (CSR)	51
Adding the Intermediate and Root Certificates to the Keystore.	54
Receive the Signed Certificate to Create the KeyCert	56
Extracting CA Certificates using IKEYMAN . . .	58
Add CA Certificates using IKEYMAN	59
Defining Private Key in IKEYMAN to Use as Site Certificate in gskkyman or RACF	60
Importing Private Key Into IKEYMAN	62
Renew an Existing Key Certificate.	65
Using IKEYMAN From the Command Line Interface (CLI)	67
Create a Key Database.	67
Create a Certificate Signing Request (CSR)	67
Create a Self-Signed Certificate	68
Add Certificate to Key Database.	69
Extract Certificate from Key Database. . . .	70
Viewing Certificates in the Keystore	70
Renew Existing Key Certificate.	71
Updating Secure+ to Use the New Key Certificate.	71
Migrating to New Site Certificate (or Using an Alternate Site Certificate).	72
Removing the Old Certificate and Adding the New Certificate.	72
Keeping Both the Old and New Certificates. .	72

Managing the Secure+ Parameter File (PARMFILE) on C:D z/OS 74

Manually Create Connect:Direct Secure+ PARMFILE	74
Creating the Secure+ PARMFILE from the NETMAP (Quick Start).	75
Creating Each Node Manually (Without Using the NETMAP)	78
Saving the New Secure+ PARMFILE	82
Cipher Filtering and TLSv1.3 (CD z/OS 6.1 or Greater)	84
Support for TLSv1.3	84
Cipher Filtering / Cipher Sorting	85
Converting Pre-C:D z/OS 5.2 Secure+ PARMFILE to C:D z/OS 5.2 or Later.	87

Strong Password Encryption (SPE) Used.	88
Strong Password Encryption (SPE) NOT Used. .91	
Making a New Pass Phrase for the Connect:Direct	
Secure+ PARMFILE.	93
Secure+ PARMFILE Cloning	97
Encrypting the Pre-C:D z/OS 5.2 Secure+	
PARMFILE with TDES	102
Strong Password Encryption (SPE)	106
Implementing Strong Password Encryption . .	106
Disabling Strong Password Encryption	110
Using FIPS Mode	112
Security Modes	113
SP800-131 A Transition Mode.	113
SP800-131 A Strict Mode.	113
NSA Suite B 128 bit.	113
NSA Suite B 192 bit.	114
Configuring FIPS Mode in C:D z/OS	114

Setting Up Secure+ Connections.115

Setting Up Secure+ SSL or TLS Connection on	
C:D z/OS	115
Update the Client (Sender) Node.	116
Add or Change Client (Site) Certificate on	
Local Node.	118
Update the Remote Node on Client	125
Update the Server (Receiver) Node in Server	
PARMFILE	127
Add or Change Server Certificate on Local	
Node	128
Update the Remote Node on Server	134
Using Secure+ on C:D Windows / C:D Unix	136
Setting Up a Secure+ SSL/TLS Connection for	
C:D Windows / C:D Unix	138
Create New CMS KeyStore	138
Import Certificate into CMS KeyStore	140
Update the .Local Node for C:D	
Windows / C:D Unix	145
Update the Remote Node for C:D	
Windows / C:D Unix	149

Using Secure+ with z/OS Security

Applications.153

Secure+ using RACF.	153
Create RACF Keyring	153
Generate Certificate Request (CSR).	153
Generate Self-Signed Certificate	154
Load Certificate into Keyring	154
Secure+ using CA-ACF2	157
CA-ACF2 Identifiers	157
List Connect:Direct Keyring and Certificate. .	157
Export Certificate Format.	158
CA-ACF2 Keyring Commands	159
Reference.	160
Secure+ using CA-Top Secret.	161
CA-Top Secret Identifiers	161
List Connect:Direct Keyring and Certificate. .	161
Export Certificate Format.	162
Import Certificate Format	163

Troubleshooting165

Common Secure+ Errors and Possible	
Solutions.	166
Saving the Secure+ PARMFILE	166
Initializing Connect:Direct with Secure+	167
SSL Authentication	168
FIPS Errors	172
Capturing a z/OS System SSL Trace	173
Method 1: Using the GSKSRVR Utility	173
Method 2: Does not require GSKSRVR	174
Reference	175
Creating a Secure+ LOOPBACK Node	176
C:D Windows/Unix Node	176
C:D z/OS Node.	177
Sending in C:D z/OS Secure+ PARMFILE and Access	
File to Support	179
Manually Create Secure+ VSAM Files (PARMFILE	
and Access File)	180

IBM Sterling Connect:Direct Secure+

IBM Sterling Connect:Direct Secure+ (or Connect:Direct Secure+) provides enhanced security for IBM Sterling Connect:Direct using the SSL/TLS protocol, which gives Connect:Direct (C:D) strong authentication and strong encryption. On z/OS, C:D uses IBM System SSL, a z/OS system service, to implement the SSL/TLS protocol. On UNIX and Windows, C:D uses IBM Global Security Kit (GSKit), which is embedded in the product (prior to C:D Windows 4.7 and C:D Unix 4.2, OpenSSL was used).

NOTE: IBM System SSL, also referred to as the SSL Toolkit, is not the same as the SSL protocol.

How Connect:Direct Establishes a Secure Connection

SSL/TLS uses certificates to authenticate the SSL/TLS server to the SSL/TLS client and to establish an encryption key for the session. When SSL/TLS client authentication is enabled, the client is also authenticated to the server. To relate SSL/TLS and Connect:Direct terminology: when a C:D primary node (PNODE) initiates a connection with a C:D secondary node (SNODE), the PNODE is the SSL/TLS client and the SNODE is the SSL/TLS server.

- Connect:Direct ALWAYS performs Server Authentication in a Secure+ session; Client Authentication is optional.
- When the PNODE (client) Submits a process script the connection request is made to the SNODE (server). The SNODE checks the Connect:Direct NETMAP and validates the request and then determines if a Secure Session is being requested.
- Once it has determined that a Secure Session has been requested the SNODE (typically the Remote Node) opens the keyring or key database on their end and extracts the Public Certificate part of the file and sends that to the PNODE.
- The PNODE takes the Public Certificate from the SNODE and compares it to the certificate stored in the keyring / key database that was sent to them previously by this remote node.
- If the SNODE certificate sent finds a match in the PNODE keyring/key database, the connection proceeds with other checks on a Connect:Direct level like the Submitter ID, password, file names, etc.
- If the SNODE certificate sent does not find a match in the PNODE keyring/key database, this means the certificate sent previously by this Remote Node and stored in the PNODE keyring does not match the site certificate on the SNODE and the SSL Handshake (and therefore the process) fails with an error.
- A mismatch can mean the SNODE has provided a non-matching (or incorrect) certificate to the PNODE, the certificate loaded in the keyring/key dataset is bad (e.g. wrong format), a full certificate chain was not found (ROOT and all Intermediates) or the certificate chain was not loaded correctly (e.g. each certificate and intermediate must be loaded individually, not all at once).
- In the case of a mismatch either the Remote Node needs to provide a new certificate that does match or the PNODE needs to reload the certificate(s) into their keyring/key database.

- **Remember IBM does not provide certificates.** IBM only provides a place for you to store the certificates and validates the chain. If the transfer is failing due to a bad certificate, the party that supplied the certificates being used will need to be contacted.
- Therefore, it is critical to understand what being the PNODE (Submitter) means. Using this analysis allows you to know exactly which end of the transfer has provided a non-matching or bad certificate.
- Once the Secure+ transfer from the PNODE (submitter) is working, and you are confident that all server certificates are validated properly you can then utilize an additional Secure+ option to validate the client's certificates: Client Authentication.
- Client Authentication is available for use. It is not required; it is optional.
- Client Authentication is determined by the SNODE using the **Enable Client Auth** flag (z/OS) or **Enable Client Authentication** (Windows or Unix).
- To use Client Authentication only the SNODE needs to set the **Enable Client Auth** flag.
- Connect:Direct always performs Server Authentication where the PNODE initiates and the SNODE sends their certificate out of the keyring/key database.
- The **Enable Client Auth** flag means the SNODE also requires the PNODE to send their certificate from their keyring/key database to be authenticated. When Client Authentication is enabled, Server Authentication is still done, but after the Server Authentication has successfully validated the Server (SNODE) certificate, the Client (PNODE) certificate is then authenticated.
- Therefore, if the SNODE has Client Authentication enabled, both the PNODE certificate(s) and the SNODE certificate(s) are authenticated before the session can successfully transfer.
- If one side makes changes or has an issue and transfers begin to fail, Client Authentication will need to be disabled to determine which end of the transfer has a problem with their certificates.
- **IMPORTANT:** If a secure connection does not work, the problem may be on one or both ends. Begin problem determination by turning off Client Authentication (meaning only server authentication will be done) and make sure this works (this means verifying the certificates from the server or receiver are good and load correctly on the client or sender). Once this connection works, turn Client Authentication back on and, if the connection now fails, correct whatever problem there is with the certificates from the client side (make sure they are good and loaded correctly on the server or receiver).

Connect:Direct Secure+ Session (Handshake) Flow Grid

PNODE (Client)		SNODE (Server)
Establish TCP connection with the SNODE		
(1) Look up SNODE Secure+ Configuration		
(2) Send first FMH68 with XDR tokens CRYPT and SSLO set to 2, 3, 4, 5 or 6 2 = SSL 3=TLS 4 = TLSv1.1 5 = TLSv1.2 6 = TLSv1.3	----->> FMH68 ----->>	(3) Get PNODE name from XDR token TNOD
		(4) Look up PNODE configuration
	<<----- S+FMH68 <<-----	(5) Reply with S+FMH68
	At this point, both the PNODE and the SNODE have agreed to attempt a Secure+ connection ***** Begin SSL Handshake *****	
(6) Send 'Client Hello' with requested protocol, ciphers	----->> Client Hello ----->>	(7) Validate protocol, select common cipher
	Server Authentication (always done)	
(9) Authenticate server certificate against local trust store (i.e. keyring/key database)	<<----- Server Hello <<----- <<----- Server Certificate <<-----	(8) Send 'Server Hello' with selected cipher and protocol, then server certificate and (possibly) intermediate certificate(s)
(10) Did the SNODE request Client Authentication? If no, go to step (13).		
	Client Authentication (optional, requested by the SNODE)	
(11) Send client certificate and (possibly) intermediate certificate(s)	----->> PNODE Certificate ----->>	(12) Authenticate client certificate against the local trust store (i.e. keyring/key database)
(13) Complete SSL Handshake		
	***** SSL Handshake Completed *****	
(14) Send third FMH68 confirming the handshake completion	----->> FMH68 ----->>	
	A Secure+ session has been established. The standard C:D FMH flow will proceed using SSL encryption.	
	<<----- FMH70 <<-----	(15) Send FMH70 to PNODE
(16) Send FMH70 +rsp to SNODE	----->> FMH70 +rsp ----->>	

Connect:Direct Secure+ Flow Grid Explanation

1. After completing the TCP connection, the PNODE checks the configuration of the SNODE (remote node) in its Secure+ Admin file (PARMFILE).
 2. If the configuration specifies a Secure+ session, the PNODE sends an FMH68 that includes the XDR tokens CRYP (Secure+ active) and SSLO (SSL Override) with a value of 2 (SSL protocol), 3 (TLSv1.0 protocol), 4 (TLSv1.1 protocol), 5 (TLSv1.2 protocol) or 6 (TLSv1.3 protocol).
 3. On receipt of the initial FMH68, the SNODE retrieves the PNODE node name from the FMH68.
 4. The SNODE looks up the remote node record in its PARMFILE and identifies the protocol (SSL, TLSv1.0 (or just TLS), TLSv1.1, TLSv1.2, or TLSv1.3) with which the remote node is configured. Most platforms (such as z/OS, Windows, and Unix), as SNODE, require an exact match on the protocol requested in the FMH68 with the remote node record on the SNODE; a few others do not check the protocol and instead allow the SSL handshake to negotiate the protocol.
- NOTE:** Beginning with C:D z/OS 5.2, C:D Windows 4.7 and C:D Unix 4.2, multiple protocols can be coded, and the highest “available” protocol will be used. For example, if TLSv1.0, TLSv1.1 and TLSv1.2 are defined for a remote, and that remote comes in with TLSv1.0, then TLSv1.0 will be used. If the same remote later comes in with TLSv1.2, then TLSv1.2 will be used.
5. Assuming the remote node entry for the PNODE is configured for Secure+, the SNODE replies with an S+FMH68 confirming that a Secure+ session will be attempted.
 6. The PNODE System SSL sends a 'Client Hello' message with the requested protocol and list of available ciphers.

NOTE: The term **System SSL** is used for the SSL application doing the actual authentication; this is either **IBM System SSL** on z/OS or **IBM Global Security Kit (GSKit)** on Windows or Unix.

7. The SNODE System SSL validates the requested protocol and selects a common cipher from its own list of available ciphers.

NOTE: To enforce the use of a specific cipher, configure **ONLY** that cipher in the respective remote node records. As more ciphers are made available, the lower the security level for that connection.

The recommended Protocol and Cipher is **TLS** (preferably **TLSv1.2**) and **TLS_RSA_WITH_AES_256_CBC_SHA**.

NOTE: Going from a pre-C:D z/OS 5.2 version to C:D z/OS 5.2 or later, the names of the cipher suites (or ciphers) within the Secure+ PARMFILE changed from prefix **SSL_** to prefix **TLS_**. These cipher names are only used to identify them within Connect:Direct Secure+; **System SSL** looks at the hexadecimal cipher code.

HEX	Pre-Connect:Direct 5.2 Cipher Name	same as	HEX	Connect:Direct 5.2 Cipher Name
2F	SSL_RSA_AES_128_SHA		002F	TLS_RSA_WITH_AES_128_CBC_SHA
35	SSL_RSA_AES_256_SHA		0035	TLS_RSA_WITH_AES_256_CBC_SHA
0A	SSL_RSA_WITH_3DES_EDE_CBC_SHA		000A	TLS_RSA_WITH_3DES_EDE_CBC_SHA
09	SSL_RSA_WITH_DES_CBC_SHA		0009	TLS_RSA_WITH_DES_CBC_SHA

Connect:Direct also has a FIPS mode; while in FIPS mode, only certain ciphers are supported. During the TLS handshake, any non-FIPS mode ciphers are ignored.

For information regarding which ciphers can be used for FIPS mode, please see **Using FIPS Mode** on page 112.

8. The SNODE returns a "Server Hello" with the negotiated protocol and selected cipher, then its key certificate (the server certificate) and any matching intermediate certificates included in the key certificate file or local key store (key ring/database if SNODE is a mainframe).

9. The PNODE receives the server certificate and attempts to authenticate it using the contents of its local keyring/key database. Authentication requires that System SSL be able to construct a complete certificate chain ending with a root certificate - one in which the Issuer (**Issued By**) and Subject (**Issued To**) fields match.

NOTE: It is not uncommon for CA-issued certificates to require additional intermediate certificates to complete the authentication chain to a root certificate. Intermediate certificates are typically stored with the key certificate but may also be placed as TRUSTED in the keyring/key database on the authenticating node.

10. The PNODE determines whether the SNODE has requested client authentication (specified in the 'Server Hello'). Client Authentication is optional and is configured on the SNODE.

If Client Authentication was not requested, go to step **13**.

11. The PNODE sends its certificate and any intermediate certificates in the certificate file to the SNODE.

12. The SNODE authenticates the client certificate against the local keyring/key database in the same way as done earlier by the PNODE when the SNODE certificate was authenticated.

NOTE: When Client Authentication is enabled, the SNODE also has the additional option to validate the **Certificate Common Name** field contained in the client's certificate. If the Common Name in the client certificate does not match the **Certificate Common Name** field in the remote node record on the SNODE, the Client Authentication step will fail.

13. If the certificate authentication step(s) completed successfully, a final exchange of messages is done by both PNODE and SNODE System SSL to establish encryption keys, and the SSL handshake is completed.

14. The PNODE sends a final (the third) FMH68 to confirm completion of the SSL handshake and successful setup of a Secure+ session.

15. Standard Connect:Direct message flow resumes with the SNODE sending an encrypted FMH70. From this point forward, all messages exchanged between the two nodes will be encrypted and authenticated with the cipher and message authentication codes established during the SSL handshake.

NOTE: The standard encryption used is to encrypt the FMH headers only; the actual data being transmitted is not encrypted. To have the data encrypted, the **Enable Data Encrypt** flag must be set in the PARMFILE on the remote node definition (or set on the local node and defaulted by the remote). To ensure the data is encrypted, both PNODE and SNODE need to have this set to **Yes**.

NOTE: Beginning in C:D z/OS 6.2, the **Enable Data Encrypt** flag will now be forced to Y, meaning all data is encrypted in C:D z/OS Secure+.

16. The PNODE sends back a response FMH70 to the SNODE and the handshake is now complete.

SSL Certificate Validation Simplified

The following is an example of how the certificate chain (i.e., a certificate with intermediates) gets authenticated for an SSL/TLS transfer sending a file from NodeC (client) to NodeS (server):

NOTE: The “**Issued By**” was formerly “**Issuer**” and the “**Issued To**” was formerly “**Subject**”; some older certificates and older documentation may have **Issuer** and **Subject**.

1. NodeC sends an FHM68 to NodeS requesting a handshake.
2. NodeS sends an S+FMH68 back to NodeC containing their certificate (sending certificate).
3. System SSL (previously called the SSL Toolkit) takes that sending certificate and attempts to authenticate it in the keyring or key database of NodeC by doing the following:
 - a. The **Issued By** on the sending certificate is compared to the certificates in the key ring/database until SSL finds a certificate with a matching **Issued To**.
 - b. When the new certificate is found, SSL then checks to see if the **Issued By** matches the **Issued To**.
 - c. If the **Issued By** and the **Issued To** match, then SSL knows this is the root and the authentication is complete. However, if they are not the same, this is not the root certificate and SSL continues.
 - d. If the **Issued By** and **Issued To** were different, SSL switches the **Issued To** to the **Issued By** and now uses this "new" **Issued By** to search for a certificate with a matching **Issued To**.
 - e. If a match is found, SSL compares the **Issued By** to the **Issued To**; once again, if they are not the same, this is not the root certificate and SSL continues.
 - f. This continues until either no match is found (i.e., SSL Authentication fails) or a certificate is found where the **Issued By** and **Issued To** match, meaning it is the root certificate.

NOTE: If anywhere in this chain a match cannot be found, the validation error is returned.

4. SSL compares the certificate sent in the S+FMH68 to the certificate found in the key ring/database.

NOTE: SSL can also send a partial or complete certificate chain to be authenticated. If part or all the chain is sent by NodeS, it must be authenticated completely by what is contained in the key ring/database on NodeC.

5. If the certificates match, then SSL completes the server authentication and returns a successful return code to Connect:Direct.
6. If Client Authentication is requested by NodeS (i.e., it is always the server who requests Client Authentication), then this process is repeated with NodeC sending its certificate to NodeS. If Client Authentication is not requested, then the SSL Authentication is complete.
7. NodeS sends a FMH70 to NodeC to indicate the SSL Authentication is complete.

To reiterate, here is what happens:

Sending Certificate has **Issued By** = A2.

Certificate is found with **Issued By** = A1, **Issued To** = A2.

Issued By A2 matches **Issued To** A2, but since this certificate has **Issued By** A1 not matching **Issued To** A2...

Switches **Issued By** to A1, now looks for certificate with **Issued To** = A1

Finds Certificate with **Issued By** = A1, **Issued To** = A1

Since **Issued By** and **Issued To** now match on the same certificate, SSL now knows it has found the root.

The root certificate should be the only certificate marked “**trusted**” in the certificate chain. Also, all certificates in the certification chain must not be expired.

If there is a problem with the certificate chain or any issues with determining why the SSL Authentication is not working successfully, a System SSL Trace (z/OS) or a full trace including Secure+ (Windows or Unix) containing the error will need to be collected by at least the sending side, probably both sides simultaneously, and will probably need to be analyzed by IBM SSL Support.

Please refer to section **Capturing a z/OS System SSL Trace** for assistance in taking a System SSL Trace.

Using Client Authentication

What is Client Authentication?

With any Secure+ transmission, the sending side (the initiator) is the "client" and the receiving side (the recipient) is the "server." When using either the SSL or TLS protocols, certificates are exchanged between the client and the server and must be authenticated before the transmission can be successful.

All Secure+ transactions perform "Server Authentication," meaning that on the handshake to establish the session, the server (receiving side) sends its certificate to the client (sending side) and that certificate is then authenticated against the "server" certificate that is stored in the client's keyring (or key database). If this certificate sent from the server does not match the server certificate in the client's keyring, then the authentication fails, the transmission is halted and the session ends. In this way, the recipient, or destination, of the file is authenticated; that is, the sender is assured the file has gone to the correct and trusted recipient.

To enable Client Authentication, update the remote node definition for this trading partner and set your cursor on **SSL/TLS Parameters** and hit **Enter** to go to the **SSL/TLS Parameters Panel**. Set the **Enable Client Auth** flag is to **Y**, then click **OK**. Then save the PARMFILE.

The screenshot shows a terminal-style window titled "Secure+ Create/Update Panel". At the top, it says "Option ==>". Below that, it displays "Node Name: REMOTE_NODE" and "Type: R (Local or Remote)". A table-like structure shows three columns: "Security Options", "EA Parameters", and "SSL/TLS Parameters". The "SSL/TLS Parameters" column is highlighted with a red box. Below this, there are two rows of settings: "Enable Client Auth" with a value of "Y" (highlighted with a red box) and "Enable Data Encrypt" with a value of "Y". To the right of these is the text "(Yes , No , Default to Local)". Below these settings is a section for certificates with labels "Certificate Label", "Cipher Suites", "Certificate Pathname", and "Certificate Common Name", each followed by a value in a box: "*", "FFFF", "*", and an empty box respectively. At the bottom right, there are "OK" and "Cancel" buttons, both highlighted with red boxes.

Option ==>
Node Name: REMOTE_NODE
Type: R (Local or Remote)
Security Options
EA Parameters
SSL/TLS Parameters
Enable Client Auth Y
Enable Data Encrypt Y
Certificate Label *
Cipher Suites FFFF
Certificate Pathname *
Certificate Common Name
OK
Cancel

NOTE: This can also be set on the local node with the remote node defaulting to the local.

If this flag is set on, then after the server's certificate is authenticated, the server will then request the client to send its certificate, which will then be authenticated against the client certificate already stored in the server's keyring/key database. **Client Authentication is always requested from the server;** if the client certificate sent back to the server does not match the client certificate in the server's keyring/key database, then just like with server authentication, the authentication fails, the transmission is halted and the session ends. In this way, the sender, or originator, of the file is authenticated; that is, the recipient is assured that the file received is coming from the correct and trusted sender.

When Should Client Authentication Be Used?

In most cases, performing the server authentication is enough; that is, it is typically sufficient to only authenticate the recipient, or destination, of the file being sent without authenticating the sender. However, there are instances where both the recipient and the sender of the file require authentication to insure precisely who is sending the file and precisely where it is being sent.

For example, assume one is ordering an item from a retail website. When signing on to the website, server authentication is done to make sure that the correct website is being accessed. However, the website itself does not care who signed on, only that the person signing on has money and wants to purchase an item. In this case, only server authentication is required to insure a secure transaction.

Now, let's say that this same person signs onto his online banking website. Again, server authentication is done to make sure the correct website is being accessed. However, this time the website (i.e. bank) now needs to know who is attempting to sign in, so the website will then require the client to authenticate who they are to make sure that they have the correct authority to sign onto this website (other than just having the correct password). In this case, both server and client authentication are required to insure a secure transaction.

When should Client Authentication be used? When both the sending and the receiving sides need to be authenticated to insure the secure transaction.

One point should be made regarding transmission failures: if a transmission fails, Client Authentication should be turned off so that it can be determined from which side the failure is occurring.

Creating and Managing Certificates

There are multiple applications that can be used to create certificates, but the certificate must be an RSA X.509 V3 certificate in PEM format to be useable by Connect:Direct. When a certificate is exported, for example when providing a certificate for a trading partner's trusted store, it must be exported in PEM format. It is unusual to export a private key from a keystore, but if one is exported, it must be exported in PKCS12 format, if it is to be imported by Connect:Direct on the z/OS, UNIX and Windows platforms. If a private key is in PEM format it must be converted to PKCS12 format before it can be imported by Connect:Direct on the z/OS, UNIX and Windows platforms. IBM Global Security Kit (GSKit) can be used to convert formats.

NOTE: Connect:Direct can support the use of a single keyring or a single key database, but not both.

Distinguished Names in an X.509 Certificate

The **Distinguished Name** (DN) uniquely identifies an entity in an X.509 certificate. The following attribute types are commonly found in the DN:

Attribute	Description
SERIALNUMBER	Certificate serial number
MAIL	Email address
E	Email address (Deprecated in preference to MAIL)
UID or USERID	User identifier
CN	Common Name
T	Title
OU	Organizational Unit name
DC	Domain Component
O	Organization name
STREET	Street / First line of address
L	Locality name
ST (or SP or S)	State or Province name
PC	Postal code / zip code
C	Country
UNSTRUCTUREDNAME	Host name
UNSTRUCTUREDADDRESS	IP address
DNQ	Distinguished name qualifier

The X.509 standard defines other attributes that do not typically form part of the DN but can provide optional extensions to the digital certificate.

The X.509 standard provides for a Distinguished Name (DN) to be specified in a string format. For example:

CN=John Smith, OU=Test, O=IBM, C=GB

The Common Name (CN) can describe an individual user or any other entity, for example a web server.

The DN can contain multiple Organizational Unit (OU) names and Domain Component (DC) attributes. Only one instance of each of the other attributes is permitted. The order of the OU entries is significant: the order specifies a hierarchy of OUs, with the highest-level unit first. The order of the DC entries is also significant.

Using Unix System Service (USS) gskkyman (Key Database for C:D z/OS)

gskkyman is a z/OS shell-based program that creates, completes, and manages a z/OS file or z/OS PKCS #11 token that contains PKI private keys, certificate requests, and certificates. The z/OS file is called a key database and, by convention, has a file extension of **.kdb**.

Building a Key Database Using gskkyman

1. You must have a directory created under USS (e.g. /u/userid). This path is a local path in your USS.

NOTE: This path, along with the database name, will be coded as the **Certificate Pathname** in the Secure+ PARMFILE if a key database is used.

2. Access USS via **TSO ISPF Option 6**, then entering **OMVS** or via the command line '**TSO OMVS**'
3. USS is case sensitive.

Once in **OMVS**, enter **gskkyman**.

```
IBM
Licensed Material - Property of IBM
5650-ZOS Copyright IBM Corp. 1993, 2015
(C) Copyright Mortice Kern Systems, Inc., 1985, 1996.
(C) Copyright Software Development Group, University of Waterloo, 1989.

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

IBM is a registered trademark of the IBM Corp.

$

==> gskkyman

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

Hit Enter.

You will see the following:

```
Database Menu

1 - Create new database
2 - Open database
3 - Change database password
4 - Change database record length
5 - Delete database
6 - Create key parameter file
7 - Display certificate file (Binary or Base64 ASN.1 DER)

11 - Create new token
12 - Delete token
13 - Manage token
14 - Manage token from list of tokens

0 - Exit program
Enter option number:
===>
```

Enter **1** to create a new database.

1. Enter **key database name** ending with **.kdb** extension. This will create an **.rsd** file as well.
2. Enter **password** and **confirm**.
3. Enter **password expiration** or just **enter for no expiration**.
4. Enter **record length** or **let default (5000)**. Length must be between 2500 and 65536.
5. If you need this to be a FIPS key database, select **1** for FIPS.

Enter **0** to exit out of gskkyman. You will see the \$ prompt.

If you will need to use **chmod** to grant **read** and **write** permissions to the new database with the following:

chmod o=rw your_database.kdb

```
Label: TestCert

1 - Show certificate information
2 - Show key information
3 - Set key as default
4 - Set certificate trust status
5 - Copy certificate and key to another database/token
6 - Export certificate to a file
7 - Export certificate and key to a file
8 - Delete certificate and key
9 - Change label
10 - Create a signed certificate and key
11 - Create a certificate renewal request

0 - Exit program
Enter option number (press ENTER to return to previous menu): 0
$ chmod o=rw test.kdb
$
===>
```

You now should see the certificate file you just created:

```
$ ls
Z51.crt      Z52.crt      test.crt      TestCert.crt
Z51.kdb      Z52.kdb      test.kdb
Z51.rdb      Z52.rdb      test.rdb
$
===>
```

Type **exit** and hit **Enter** to exit **OMVS**.

NOTE: If you want to copy the certificate to send to the remote, do not get out of **OMVS** yet.

Building Certificates Using gskkyman

NOTE: This database name, along with the path to the database, will be coded as the **Certificate Pathname** in the Secure+ PARMFILE if a key database is used.

From **TSO USPF Option 6** enter **OMVS**, (or enter **TSO OMVS** via the command line and hit **Enter**) then enter **gskkyman**. (remember that USS is case sensitive).

```
IBM
Licensed Material - Property of IBM
5650-ZOS Copyright IBM Corp. 1993, 2015
(C) Copyright Mortice Kern Systems, Inc., 1985, 1996.
(C) Copyright Software Development Group, University of Waterloo, 1989.

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

IBM is a registered trademark of the IBM Corp.

$

====> gskkyman

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

Hit **Enter**.

This brings up the **Database Menu** panel:

```
Database Menu

1 - Create new database
2 - Open database
3 - Change database password
4 - Change database record length
5 - Delete database
6 - Create key parameter file
7 - Display certificate file (Binary or Base64 ASN.1 DER)

11 - Create new token
12 - Delete token
13 - Manage token
14 - Manage token from list of tokens

0 - Exit program

Enter option number:
====>
```

Note: If you need to create a new database before building the certificate, refer to section **Building a Key Database Using gskkyman**.

Enter **2** for **Open database**.

1. Enter **key database name**, including **.kdb** extension
2. Enter **database password**

Enter 6 for **Create a Self-Signed Certificate**.

```
Key Management Menu

Database: /u/userid/test.kdb
Expiration: 2045/04/24 16:16:41
Type: FIPS

1 - Manage keys and certificates
2 - Manage certificates
3 - Manage certificate requests
4 - Create new certificate request
5 - Receive requested certificate or a renewal certificate
6 - Create a self-signed certificate
7 - Import a certificate
8 - Import a certificate and a private key
9 - Show the default key
10 - Store database password
11 - Show database record length

0 - Exit program

Enter option number (press ENTER to return to previous menu):
==> 6
```

NOTE: If this is a FIPS database, you will see **Type: FIPS** here; if it is not a FIPS database, depending on your version of z/OS, there will either be nothing shown here, or it will have **Type: non-FIPS**.

NOTE: If no expiration date was specified for the key database, it will show **Expiration: None**.

For the **Certificate Usage**, enter 2 for **User or server certificate**.

```
Certificate Usage

1 - CA certificate
2 - User or server certificate

Select certificate usage (press ENTER to return to menu):
==> 2
```

For the **Certificate Key Algorithm**, enter 1 for **Certificate with an RSA key**.

```
Certificate Key Algorithm

1 - Certificate with an RSA key
2 - Certificate with a DSA key
3 - Certificate with an ECC key

Select certificate key algorithm (press ENTER to return to menu):
==> 1
```

Enter the desired **RSA Key Size** (option 2 for **2048-bit key** is used in this example):

```
RSA Key Size

1 - 1024-bit key
2 - 2048-bit key
3 - 4096-bit key

Select RSA key size (press ENTER to return to menu):
==> 2
```

Enter the desired certificate type; in this example type 1 for **SHA-1** (the **default**) is used.

Note: A SHA-1 certificate can be used for TLSv1.3, but it must be at least 2048 in length.

```
Signature Digest Type

1 - SHA-1
2 - SHA-224
3 - SHA-256
4 - SHA-384
5 - SHA-512

Select digest type (press ENTER to return to menu):
==> 1
```

Options **2-5** are **SHA-2** certificates and are also supported by Connect:Direct but may not be supported by the remote node if it is back-level.

SHA-2 Certificate Compatible Software:	Minimum Version:
Connect:Direct for z/OS	All versions
Connect:Direct for Unix	4.1.0, 4.0.00 with 04Mar2011 maintenance
Connect:Direct for Windows	4.6.0, 4.5.01, 4.5.00 Patch 035, 4.4.01, 4.4.00 Patch 068
Sterling Secure Proxy	3.4.1 and later
Connect:Direct for HP NonStop	3.6.01 and later
Connect:Direct for i5/OS	3.7.0 and later

NOTE: Connect:Direct supports **SHA-2** certificates on all protocols; however, **SHA-2** ciphers require TLSv1.1 or TLSv1.2.

You will now need to enter information regarding the certificate, beginning with the **label**:

```
Select digest type (press ENTER to return to menu): 1
Enter label (press ENTER to return to menu):
===>
```

After each, hit **Enter**.

1. Enter **label**. This is used for both local and remote Secure+ nodes and is used to identify your certificate in the database.

NOTE: If this certificate is used for the local site certificate, this Label will be added to **Certificate Label** in the Secure+ PARMFILE.

2. Enter **Common name**. Connect:Direct does not validate Common name unless Client Authentication is selected.
3. Enter **Organizational unit** (optional).
4. Enter **Organization** (required).
5. Enter **City/Locality** (optional).
6. Enter **State/Province** (required).
7. Enter **Country/Region** (2 characters – required).
8. Enter **number of days certificate will be valid** or **let default**. << must be between 1 and 9999 days >>
9. After you see **Certificate created** hit **Enter**.

```
Select digest type (press ENTER to return to menu): 1
Enter label (press ENTER to return to menu): TestCert
Enter subject name for certificate
  Common name (required): Test_Cert
  Organizational unit (optional): Support
  Organization (required): IBM
  City/Locality (optional): Irving
  State/Province (optional): TX
  Country/Region (2 characters - required): US
Enter number of days certificate will be valid (default 365): 9999
Enter 1 to specify subject alternate names or 0 to continue: 0
Please wait .....
Certificate created.
Press ENTER to continue.
===>
```

You should now be back at the **Key Management Menu** panel.

Enter **1** for **Manage keys and certificates**.

```
Key Management Menu

Database: /u/userid/test.kdb
Expiration: 2045/04/24 16:16:41

1 - Manage keys and certificates
2 - Manage certificates
3 - Manage certificate requests
4 - Create new certificate request
5 - Receive requested certificate or a renewal certificate
6 - Create a self-signed certificate
7 - Import a certificate
8 - Import a certificate and a private key
9 - Show the default key
10 - Store database password
11 - Show database record length

0 - Exit program

Enter option number (press ENTER to return to previous menu):
==> 1
```

Enter **1** for your certificate label name

NOTE: If there are multiple labels, select the certificate label that was just created. Depending on how many there are, you may need to scroll down until the label just created is found.

```
Key and Certificate List

Database: /u/userid/test.kdb

1 - TestCert

0 - Return to selection menu

Enter label number (ENTER to return to selection menu, p for previous list):
==> 1
```

If this is to be the default certificate, enter **3** to **Set key as default** and press **Enter**; if not, skip this step.

```
Key and Certificate Menu

Label: TestCert

1 - Show certificate information
2 - Show key information
3 - Set key as default
4 - Set certificate trust status
5 - Copy certificate and key to another database/token
6 - Export certificate to a file
7 - Export certificate and key to a file
8 - Delete certificate and key
9 - Change label
10 - Create a signed certificate and key
11 - Create a certificate renewal request

0 - Exit program

Enter option number (press ENTER to return to previous menu): 3

Default key set.

Press ENTER to continue.
==>
```

Press **ENTER** to continue. You will be back at the **Key Management Menu** panel.

Enter **4** for **Set certificate trust** status, then enter **1** for **trusted**.

```
Key and Certificate Menu

Label: TestCert

1 - Show certificate information
2 - Show key information
3 - Set key as default
4 - Set certificate trust status
5 - Copy certificate and key to another database/token
6 - Export certificate to a file
7 - Export certificate and key to a file
8 - Delete certificate and key
9 - Change label
10 - Create a signed certificate and key
11 - Create a certificate renewal request

0 - Exit program

Enter option number (press ENTER to return to previous menu): 4

Enter 1 if trusted, 0 if untrusted (press ENTER to return to menu): 1

Record updated.

Press ENTER to continue.
==>
```

Export Your Certificate from Your Key Database

If you have just finished creating a certificate, you can simply continue from where you are. However, if you are just entering **gskkyman**, open the key database (option **2** for **Open database**, enter **key database name** including **.kdb** extension, and enter **database password**), enter option **1** (**Manage keys and certificates**), enter your certificate label number (typically **1**), and enter **6** to **Export certificate to a file**.

```
Key and Certificate Menu

Label: TestCert

1 - Show certificate information
2 - Show key information
3 - Set key as default
4 - Set certificate trust status
5 - Copy certificate and key to another database/token
6 - Export certificate to a file
7 - Export certificate and key to a file
8 - Delete certificate and key
9 - Change label
10 - Create a signed certificate and key
11 - Create a certificate renewal request

0 - Exit program

Enter option number (press ENTER to return to previous menu):
==> 6
```

Enter **2** to export file in **Base64 ASN.1 DER**.

Enter **export file name**. (this can be any name you choose and is a text file).

```
Export File Format

1 - Binary ASN.1 DER
2 - Base64 ASN.1 DER
3 - Binary PKCS #7
4 - Base64 PKCS #7

Select export format (press ENTER to return to menu): 2
Enter export file name (press ENTER to return to menu): TestCert.crt

Certificate exported.

Press ENTER to continue.
===>
```

Press **ENTER** to continue. You will be back at the **Key Management Menu** panel.

Enter **0** to exit out of gskkyman.

```
Label: TestCert

1 - Show certificate information
2 - Show key information
3 - Set key as default
4 - Set certificate trust status
5 - Copy certificate and key to another database/token
6 - Export certificate to a file
7 - Export certificate and key to a file
8 - Delete certificate and key
9 - Change label
10 - Create a signed certificate and key
11 - Create a certificate renewal request

0 - Exit program

Enter option number (press ENTER to return to previous menu): 0
$
===>
```

If a new database was created, use **chmod** to grant **read / write** permissions to the database:
chmod o=rw *your_database.kdb*

Enter **exit** and hit **Enter** to end **OMVS**.

Press **Enter** again to complete the exit from **OMVS**.

NOTE: If you need to copy the certificate to send to the remote, do not get out of **OMVS** yet.

```
Label: TestCert

1 - Show certificate information
2 - Show key information
3 - Set key as default
4 - Set certificate trust status
5 - Copy certificate and key to another database/token
6 - Export certificate to a file
7 - Export certificate and key to a file
8 - Delete certificate and key
9 - Change label
10 - Create a signed certificate and key
11 - Create a certificate renewal request

0 - Exit program

Enter option number (press ENTER to return to previous menu): 0
$ chmod o=rw test.kdb
$ exit
>>>> FSUM2331 The session has ended. Press <Enter> to end OMVS.
```

Import a Remote Certificate into Your Key Database

Copy the remote certificate, as a text file, to your USS home directory (where your database is located).

Enter **OMVS** and enter **gskkyman**.

```
IBM
Licensed Material - Property of IBM
5650-ZOS Copyright IBM Corp. 1993, 2015
(C) Copyright Mortice Kern Systems, Inc., 1985, 1996.
(C) Copyright Software Development Group, University of Waterloo,
1989.

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

IBM is a registered trademark of the IBM Corp.

$

==> gskkyman
```

Enter **2** for **Open database**.

Enter **key database name**, including **.kdb** extension

Enter **database password**

```
Database Menu

1 - Create new database
2 - Open database
3 - Change database password
4 - Change database record length
5 - Delete database
6 - Create key parameter file
7 - Display certificate file (Binary or Base64 ASN.1 DER)

11 - Create new token
12 - Delete token
13 - Manage token
14 - Manage token from list of tokens

0 - Exit program

Enter option number: 2
Enter key database name (press ENTER to return to menu): test.kdb
Enter database password (press ENTER to return to menu):
```

Select **7** for **Import a certificate**.

Enter **import file name** from the remote Secure+.

NOTE: If the certificate file name being imported is not in the same directory where gskkyman is pointing, you may have to provide a path as well as the filename (for example: /u/userid/test.kdb/Z52.crt).

Enter the **label** (this is only used to identify the certificate in the key database and is not actually used by Connect:Direct). Enter anything here, but it is recommended using something that will indicate whose certificate this is).

```
Key Management Menu

Database: /u/userid/test.kdb
Expiration: 2045/04/24 16:16:41

1 - Manage keys and certificates
2 - Manage certificates
3 - Manage certificate requests
4 - Create new certificate request
5 - Receive requested certificate or a renewal certificate
6 - Create a self-signed certificate
7 - Import a certificate
8 - Import a certificate and a private key
9 - Show the default key
10 - Store database password
11 - Show database record length

0 - Exit program

Enter option number (press ENTER to return to previous menu): 7
Enter import file name (press ENTER to return to menu): Z52.crt
Enter label (press ENTER to return to menu): Z52

Certificate imported.

Press ENTER to continue.
===>
```

Press **ENTER** to continue. You will be back at the **Key Management Menu** panel.

Now you need to make this certificate **trusted**.

Select **2** to **Manage Certificates**.

```
Key Management Menu

Database: /u/userid/test.kdb
Expiration: 2045/04/24 16:16:41

1 - Manage keys and certificates
2 - Manage certificates
3 - Manage certificate requests
4 - Create new certificate request
5 - Receive requested certificate or a renewal certificate
6 - Create a self-signed certificate
7 - Import a certificate
8 - Import a certificate and a private key
9 - Show the default key
10 - Store database password
11 - Show database record length

0 - Exit program

Enter option number (press ENTER to return to previous menu):
===> 2
```

Hit **Enter** to scroll through the certificates until you find the one you just imported...

```
Certificate List

Database: /u/userid/test.kdb

1 - Equifax Secure Certificate Authority
2 - Equifax Secure eBusiness CA-2
3 - VeriSign Class 1 Public Primary CA - G2
4 - VeriSign Class 2 Public Primary CA - G2
5 - VeriSign Class 3 Public Primary CA - G2
6 - VeriSign Class 4 Public Primary CA - G2
7 - VeriSign Class 1 Public Primary CA - G3
8 - VeriSign Class 2 Public Primary CA - G3
9 - VeriSign Class 3 Public Primary CA - G3

0 - Return to selection menu

Enter label number (ENTER for more labels, p for previous list):
==>
```

When you see the certificate you imported, select that number: (in this example, **3** is chosen)

```
Enter label number (ENTER for more labels, p for previous list):

Certificate List

Database: /u/userid/test.kdb

1 - VeriSign Class 4 Public Primary CA - G3
2 - VeriSign Class 3 Public Primary CA - G5
3 - Z52

0 - Return to selection menu

Enter label number (ENTER to return to selection menu, p for previous list):
==> 3
```

Select **2** to **Set certificate trust status**.

```
Certificate Menu

Label: Z52

1 - Show certificate information
2 - Set certificate trust status
3 - Copy certificate to another database/token
4 - Export certificate to a file
5 - Delete certificate
6 - Change label

0 - Exit program

Enter option number (press ENTER to return to previous menu):
==> 2
```


Select **1**.

```
Certificate Menu

Label: Z52

1 - Show certificate information
2 - Set certificate trust status
3 - Copy certificate to another database/token
4 - Export certificate to a file
5 - Delete certificate
6 - Change label

0 - Exit program

Enter option number (press ENTER to return to previous menu): 2

Enter 1 if trusted, 0 if untrusted (press ENTER to return to menu): 1

Record updated.

Press ENTER to continue.

==>
```

Hit **Enter**. You will be back at the **Certificate Menu** panel.

Build a Certificate Signing Request (CSR)

A Certificate Signing Request (CSR) is needed to request or purchase a certificate from a Certificate Authority (CA). Make sure the certificate requested is a X.509 Base64 format version 3 certificate.

Go into **OMVS**, start gskkyman and open your key database. Select **4** to **Create new certificate request**.

```
Key Management Menu

Database: /u/userid/test.kdb
Expiration: 2045/04/24 16:16:41

1 - Manage keys and certificates
2 - Manage certificates
3 - Manage certificate requests
4 - Create new certificate request
5 - Receive requested certificate or a renewal certificate
6 - Create a self-signed certificate
7 - Import a certificate
8 - Import a certificate and a private key
9 - Show the default key
10 - Store database password
11 - Show database record length

0 - Exit program

Enter option number (press ENTER to return to previous menu):

==> 4
```

For the **Certificate Key Algorithm**, enter **1** for **Certificate with an RSA key**.

```
Certificate Key Algorithm

1 - Certificate with an RSA key
2 - Certificate with a DSA key
3 - Certificate with an ECC key

Select certificate key algorithm (press ENTER to return to menu):

==> 1
```

Enter the desired **RSA Key Size**, (option **2** for **2048-bit key** is used in this example)

NOTE: Some older operating systems may not support **4096**.

```
Select certificate key algorithm (press ENTER to return to menu): 1

    RSA Key Size

    1 - 1024-bit key
    2 - 2048-bit key
    3 - 4096-bit key

Select RSA key size (press ENTER to return to menu):
==> 2
```

Enter the desired certificate type; in this example type **1** for **SHA-1** (the **default**) is used

```
Select RSA key size (press ENTER to return to menu): 2

    Signature Digest Type

    1 - SHA-1
    2 - SHA-224
    3 - SHA-256
    4 - SHA-384
    5 - SHA-512

Select digest type (press ENTER to return to menu):
==> 1
```

The following certificate information is needed:

1. Enter **request file name** (the name of the CSR file).
2. Enter **label**. This is used for both local and remote Secure+ nodes and is used to identify your certificate in the database.
3. Enter **Common name**. C:D z/OS Secure+ does not validate Common name unless Client Authentication is selected.
4. Enter **Organizational unit (optional)**.
5. Enter **Organization (required)**.
6. Enter **City/Locality (optional)**.
7. Enter **State/Province (required)**.
8. Enter **Country/Region (2 characters – required)**.
9. Enter **0** to continue.
10. After you see **Certificate created** hit **Enter**.

```
Select digest type (press ENTER to return to menu): 1
Enter request file name (press ENTER to return to menu): test.csr
Enter label (press ENTER to return to menu): TestCSR
Enter subject name for certificate
  Common name (required): Test_CSR
  Organizational unit (optional): Support
  Organization (required): IBM Corp
  City/Locality (optional): Irving
  State/Province (optional): TX
  Country/Region (2 characters - required): US

Enter 1 to specify subject alternate names or 0 to continue: 0

Please wait .....

Certificate request created.

Press ENTER to continue.
==>
```

You will be back at the **Key Management Menu**; enter **0** to **Exit program**.

```
Key and Certificate Menu

Label: TestCert

1 - Show certificate information
2 - Show key information
3 - Set key as default
4 - Set certificate trust status
5 - Copy certificate and key to another database/token
6 - Export certificate to a file
7 - Export certificate and key to a file
8 - Delete certificate and key
9 - Change label
10 - Create a signed certificate and key
11 - Create a certificate renewal request

0 - Exit program

Enter option number (press ENTER to return to previous menu): 0
$
```

Once you exit gsskyman, the CSR will be the file you just created (in this example, **test.csr**).

```
$ ls
Z51.crt      Z52.crt      test.csr      TestCert.crt
Z51.kdb      Z52.kdb      test.kdb
Z51.rdb      Z52.rdb      test.rdb
$
==>
```

The CSR file should look something like this:

```
-----BEGIN NEW CERTIFICATE REQUEST-----
MIICoDCCAYgCAQAwWzELMAkGA1UEBhMCVVMxCzAJBgNVBAGTA1RYMQ8wDQYDVQQH
EwZJcnZpbmcxDDAKBgNVBAoTA0FyaTEMMAoGA1UEC3VwMRIwEAYDVQQDEw1E
RUxwFVEVU1IiwgEiMA0GCSqGSIb3DQEBAQUAA4IBDwAwggEKAoIBAQC2f36Kw6LU
3fIx1FoPfBgp+R+NGNkCvIVUgSpuMTmsFQzijnJ4bvgQGDY8yYDHHnU6a3qcTj0d
8A+uJtaoOct9lV3u0NomsRpL/Q5e7glXKub3ZJ9KO/vi0hU13ZfClnTPiApSXylQ
5pVl5MiQsAyAwjAf0lYQUseoiQjSUS/0QDufv5Ae+R319T9bP16IvSH5CFUFV5mZ
iNh/QAdfNAVuzPB6oNC/shIOChS0lquEUB6On7JZCABVTtTdNiXK1ZyljX2loVWs
JJD3ETj2TL8ywZxd5BMLHSpD0NwsLlpUmdGwGzYS0xqJ0t8AfGEIqy/nmhSWvo7L
Y3q+zOjPS7izAgMBAAAGgADANBgkqhkiG9w0BAQsFAAOCAQEAs9dpUQG19a7KT+Me
11VYquDNutw8FmY1MsDIPKcajQ2W6JR+eosPb6VhUdF5n616P/P1WmuMgE9fmrwk
kiiTG1KGUcvHQ5bckyEyV0TJ4gj6FNjpdLrjDrSbS+A73KfYKapXxPHImG2D3OP1
JjYzUu/bob7FUaQ5pL04Sdangmh7vqli6JdoCdCBP8gK/osANCPUzg7VW8mRnLbn
EyUW1kBSXMsMw/uwcng34OFWFKSuwXQnKWxGteu5n7x9vr5ypnPcW4mWmHvY9h2D
PBGaP3DurzPaEpUS59DpRhCfb3DgCD2nLcaZIIur/DOE6/E08oC1dZtknOGDgz2T
H8/nGg==
-----END NEW CERTIFICATE REQUEST-----
```

This file will need either need to be copied as a text file to where it can be sent to a Certificate Authority, or you can display the contents of the file and copy and paste it to something like Notepad (since Notepad does not add formatting) and save it as a text file, then send it to the Certificate Authority.

Enter **exit** and press **Enter** to end **OMVS** session.

```
$ ls
Z51.crt      Z52.crt      test.csr      TestCert.crt
Z51.kdb      Z52.kdb      test.kdb
Z51.rdb      Z52.rdb      test.rdb
$
$ exit
>>>> FSUM2331 The session has ended. Press <Enter> to end OMVS.
```

Press **Enter** to end **OMVS**.

Import CA Signed Certificate From Certificate Signing Request (CSR)

After the Certificate Signing Request (CSR) is sent to a Certificate Authority (CA), a signed certificate, a root certificate and at least one intermediate certificate will be sent back from the CA (in some instance, the intermediate may need to be downloaded separately). These will need to be imported into the same keyring or key database where the CSR was built. In this example, a key database in gskkyman is used.

The order of importing the certificates is important; the root and intermediate certificates will need to be imported first (the root first), then the signed certificate will need to be “received.”

Go into **OMVS**, start gskkyman and open your key database.

Start with the root certificate.

From the **Key Management Menu**, select **7** to **Import a certificate**.

Enter **import file name** from the remote Secure+; in this example, **CSR_Root.pem** is used.

NOTE: If the certificate file name being imported is not in the same directory where gskkyman is pointing, you may have to provide a path as well as the filename (for example: /u/userid/test.kdb/CSR_Root.pem).

Enter the **label** (this is only used to identify the certificate in the key database and is not actually used by Connect:Direct). Enter anything here, but it is recommended using something that will indicate whose certificate this is; **CSR_Root** is used in this example.

```
Key Management Menu

Database: /u/userid/test.kdb
Expiration: 2045/04/24 16:16:41

1 - Manage keys and certificates
2 - Manage certificates
3 - Manage certificate requests
4 - Create new certificate request
5 - Receive requested certificate or a renewal certificate
6 - Create a self-signed certificate
7 - Import a certificate
8 - Import a certificate and a private key
9 - Show the default key
10 - Store database password
11 - Show database record length

0 - Exit program

Enter option number (press ENTER to return to previous menu): 7
Enter import file name (press ENTER to return to menu): CSR_Root.pem
Enter label (press ENTER to return to menu): CSR_Root

Certificate imported.

Press ENTER to continue.
==>
```

You should see “**Certificate imported**” if the import was successful.

Note: If you are importing the certificates “out of order” (for example, importing the intermediate before the root is imported), you will see the following:

```
Unable to import certificate.
Status 0x03353024 - Issuer certificate not found.
```

Now import the intermediate certificate. If there are multiple intermediate certificates, they will each have to be imported in the correct order.

From the **Key Management Menu**, select **7** to **Import a certificate**.

Enter **import file name** of the intermediate certificate received; in this example, **CSR_Inter.pem** is used.

NOTE: If the certificate file name being imported is not in the same directory where gskkyman is pointing, you may have to provide a path as well as the filename (for example: /u/userid/test.kdb/CSR_Root.pem).

Enter the **label** (this is only used to identify the certificate in the key database and is not actually used by Connect:Direct). Enter anything here, but it is recommended using something that will indicate whose certificate this is; **CSR_Inter** is used in this example.

```
Key Management Menu

Database: /u/userid/test.kdb
Expiration: 2045/04/24 16:16:41

1 - Manage keys and certificates
2 - Manage certificates
3 - Manage certificate requests
4 - Create new certificate request
5 - Receive requested certificate or a renewal certificate
6 - Create a self-signed certificate
7 - Import a certificate
8 - Import a certificate and a private key
9 - Show the default key
10 - Store database password
11 - Show database record length

0 - Exit program

Enter option number (press ENTER to return to previous menu): 7
Enter import file name (press ENTER to return to menu): CSR_Inter.pem
Enter label (press ENTER to return to menu): CSR_Inter

Certificate imported.

Press ENTER to continue.
===>
```

You should see **“Certificate imported”** if the import was successful.

If there are multiple intermediate certificates, repeat this. If you receive an error that the certificate was unable to be imported, you probably have the intermediates out of order; simply retry another intermediate.

Now the signed certificate needs to be received.

From the **Key Management Menu**, select **5** to **Receive requested certificate or a renewal certificate**.

Enter **import file name** of the signed certificate received from the CA; in this example, **CSR_Signed.pem** is used.

Note: You will not be asked to enter a label here. Receiving this signed certificate will associated this certificate with the private key created when the CSR was built.

```
Key Management Menu

Database: /u/userid/test.kdb
Expiration: 2045/04/24 16:16:41

1 - Manage keys and certificates
2 - Manage certificates
3 - Manage certificate requests
4 - Create new certificate request
5 - Receive requested certificate or a renewal certificate
6 - Create a self-signed certificate
7 - Import a certificate
8 - Import a certificate and a private key
9 - Show the default key
10 - Store database password
11 - Show database record length

0 - Exit program

Enter option number (press ENTER to return to previous menu): 5

Enter import file name (press ENTER to return to menu): CSR_Signed.pem
Certificate imported.

Press ENTER to continue.
==>
```

After the signed certificate has been successfully received, you should then be able, from the **Key Management Menu**, to go into option **1 – Manage Keys and certificates** and now see the site certificate from the CSR built previously.

Export a Remote Certificate from gskkyman

There may be an instance where certificates need to be exported from gskkyman so that they can be copied to either a keyring or to a key database on another system. This is slightly different than exporting your local site certificate.

NOTE: If all certificates need to be copied from gskkyman (for example, if moving certificates from a key database to a keyring), each certificate will have to be exported individually.

Start gskkyman, enter **2** for **Open database**, enter **key database name** (including **.kdb** extension) and enter **database password**.

On the **Key Management Menu**, select option **2 (Manage Certificates)**.

```
Key Management Menu

Database: /u/userid/test.kdb
Expiration: 2045/04/24 16:16:41

1 - Manage keys and certificates
2 - Manage certificates
3 - Manage certificate requests
4 - Create new certificate request
5 - Receive requested certificate or a renewal certificate
6 - Create a self-signed certificate
7 - Import a certificate
8 - Import a certificate and a private key
9 - Show the default key
10 - Store database password
11 - Show database record length

0 - Exit program

Enter option number (press ENTER to return to previous menu):
==> 2
```

Hit **Enter** to scroll through the list until you hit the **Label** of the certificate you wish to export.

Enter the number of the certificate **Label** you wish to export (in this example, **4** is selected).

```
Certificate List
Database: /u/userid/test.kdb
1 - Z62
2 - CDWin48
3 - newcd47
4 - testcert
0 - Return to selection menu
Enter label number (ENTER to return to selection menu, p for previous list):
==> 4
```

Enter option **4** (**Export certificate to a file**).

```
Certificate Menu
Label: testcert
1 - Show certificate information
2 - Set certificate trust status
3 - Copy certificate to another database/token
4 - Export certificate to a file
5 - Delete certificate
6 - Change label
0 - Exit program
Enter option number (press ENTER to return to previous menu):
==> 4
```

For the **Export File Format**, select **2** (**Base64 ASN.1 DER**).

Enter the name you want for the certificate file, then hit **Enter**.

```
Export File Format
1 - Binary ASN.1 DER
2 - Base64 ASN.1 DER
3 - Binary PKCS #7
4 - Base64 PKCS #7
Select export format (press ENTER to return to menu): 2
Enter export file name (press ENTER to return to menu): testcert.crt
Certificate exported.
Press ENTER to continue.
==>
```

When you see the message **Certificate Exported**, the file will be in your home directory in OMVS.

You can then copy that file as a text file to wherever it needs to go.

Copying z/OS Certificate from gskkyman

The certificate should already be exported as **Base64 ASN.1 DER** to a file; please refer to section **Export Your Certificate from Your Key Database** for assistance if needed.

There are three ways to copy the certificate out of OMVS once it has been exported from the key database:

1. Copy and Paste Certificate to Pre-Allocated File – this is the easiest way
2. Copy Certificate as HFS file using Connect:Direct
3. Copy and Paste Using Open Editor (in USS)

Copy and Paste Certificate to Pre-Allocated File

Pre-allocate a file with DCB=(RECFM=FB,LRECL=80,BLKSIZE=27920,DSORG=PS) – the pre-allocated file used in this example is **USERID.TESTCERT.CERT**.

While in **OMVS** (but not in gskkyman), enter **cat *your certificate file*** (this example shows **cat TestCert.crt**).

```
$ ls Test*.*
TestCert.crt
$
==> cat TestCert.crt
```

Select and copy the entire file, including lines containing **BEGIN CERTIFICATE** and **END CERTIFICATE**.

```
$ cat TestCert.crt
-----BEGIN CERTIFICATE-----
MIIDnTCCAoWgAwIBAgIIWIsGsAAI488wDQYJKoZIhvcNAQEFBQAwXDELMAkGA1UE
BhMCVVMxChZAJBgNVBAGTAIRYMQ8wDQYDVQQHEwZJcnZpbmcxDDAKBgNVBAoTA0lC
TTEQMA4GA1UECxxMHU3VwcG9ydDEPMA0GA1UEAxMGSGsk1UZXN0MB4XDTE3MTIwODIx
NDAwMFoXDTQ1MDQyNDIxNDAwMFowXDELMAkGA1UEBhMCVVMxChZAJBgNVBAGTAIRY
MQ8wDQYDVQQHEwZJcnZpbmcxDDAKBgNVBAoTA0lCTTEQMA4GA1UECxxMHU3VwcG9y
dDEPMA0GA1UEAxMGSGsk1UZXN0MIIIBjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKC
AQEAwC114vpYcKXkrRkKvqaAj99b8xi8JkTIIoJw1ZDx1chftrQGJ5ICvUMHKMJ0
Detvu2c5EaP+gERc31jarN0hdwUldP7uC+sIAu1f+k4FtoNZjs0jvPzGRDZP+FoN
l1YTzhmQrJrh11rVtX5zCw1f6GSOVqt6Fhh8PBvgZDeKpDYszuh45Pbqacx4rBv5
xKstNCP1RPLdLKAeggoq53U8djLnVfnGggj9oVcVtUr96pBaoPH69xGt2TnjfpDr
kBu125k59Lw6QuHBfKZdMpiq9KNcs+J5TSvdTF3/dKQrJlnpHz5IMb44qon1Rd2t
SY6zRfRMoctLGSHPy3OGRHdTwIDAQABo2MwYTAdbGnVHQ4EFgQUdP8MqdQg93+S
mLK1FGY0fHmFXBUwHwYDVR0jBBGwFoAUDP8MqdQg93+SmLK1FGY0fHmFXBUwDgYD
VR0PAQH/BAQDAgH2MA8GA1UdEwEB/wQFMAMBAf8wDQYJKoZIhvcNAQEFBQADggEB
ABrruhAUswKNXGZr1/FoAlF4paOzf7p7NVQTPyXGT06bdGmqepN0nOPcS6EEIm7w
4YxwZJlh5K971XrV62587S3r+EHpKNWfIuFcbGFSh/PBXLF09TBZv1D5VtbZx4h
blodpjdLY2fZmAR9YJipWj2zXd+oRxEmv08zRlyEjV+rUhFw3LmCenuKPEJ6Wzo9
G1qYMPcbMRjKcNapQ/cMxvEq8gTEFROBFObcPAWk4K1CWvgS8xt8MoI3fvLNJdaK
JnLRhDaLSgchp0Y01HEXdqjXYZQfs2927uLWTK1lyrdHbRKZ6EBONamZuTP6G6mBu
gHGWrnd1jzFZjgX0c4R9Wc0=
-----END CERTIFICATE-----

$
==>
```


Open (edit) your pre-allocated file and paste directly into the file.

```
EDIT          USERID.TESTCERT.CERT          Columns 00001 00072
Command ==>          Scroll ==> CSR
***** ***** Top of Data *****
000001 -----BEGIN CERTIFICATE-----
000002 MIIDszCCApugAwIBAgIIYEvl4wACbX0wDQYJKoZIhvcNAQELBQAwZzELMAkGA1UE
000003 BhMCVVMx CzAJBgNVBAGTAIRYMREwDwYDVQQHEwhSb2Nrd2FsbDEQMA4GA1UEChMH
000004 U3VwcG9ydDEPMA0GA1UECXMgQWx0cmFuMRUwEwYDVQQDDAxaNjFGSVBTX2NlcnQw
000005 HhcNMjEwMzEyMjIwNjI3WhcNNDIxMDA2MjIwNjI3WjBnMQswCQYDVQQGEwJVUzEL
000006 MAKGA1UECBMCFgxEtAPBgNVBACTCFJvY2t3YWxsMRAwDgYDVQQKEwdTdXBwb3J0
000007 MQ8wDQYDVQQLEwZBbHRyYW4xFTATBgNVBAMMDFo2MUZJUUFNfY2VydDCCASIdQYJ
000008 KoZIhvcNAQEBBQADggEPADCCAQoCggEBAJhtEsdSbXv0roanvesy8iznPMGfIoNt
000009 33m+sYAogrg0Y6KGUeV32Z06gEMZe3+0PXzliTAtKMfg+8BKBALyzPVSCHx3KyVf
000010 2y0R0Xn3ltYsOLa00sFrVnEE4rK3DzRFvPw3rM6zmtXk8X0Ydwc6KVCJrRjWlRk
000011 +w7kcfBK6o5885dEjCKJk38Xgao/zXtYJQi5fm5wuvHr9T0KDuAjt3SH+bmQqiHQ
000012 PvZTw0uyQaWVhF+8N2JjD5V/x7b4kqpp1bQp6ohNXdTYlSGus7df2Q9a2MumXcwB
000013 WvIa+6kIwXm1GcoTzBLhkuK1g9djmdejjG4jNi3sLt3HGRkfKbdkh8CAwEAAANj
000014 MGEwHQYDVROOBByEFDbf9Qsf7JIEqvBDT0AgMclrEagfMB8GA1UdIwQYMBaAFDbf
000015 9Qsf7JIEqvBDT0AgMclrEagfMA4GA1UdDwEB/wQEAwIB9jAPBgNVHRMBAf8EBTAD
000016 AQH/MA0GCSqGSIb3DQEBCwUAA4IBAQCQ+1iptHMRil7S9FDr5FQaakGvawWZTDJt
000017 DfRxGZ5JrAeIFTRx3151xZgzfnqP1y2E4oR0nuWTBnqINi8gWMARzcwAbMVax/jH
000018 nBkt4VYZKwYUEYfDKP5vah3EY54LASCQC3q9DgT5RRKFFqvUXGIgBceCmB3Haqm4
000019 IRzLI9Lli+3Atp7qIRFaQXTUTI3Ggpc0OzwW+wTvNCuCo+DPDA4qrURckS/HaciE
000020 A7bvdVBZ7NHQZ3nlsuxfcIbLdve6VUuyWUWSay5h4LaFvGO16blt0KvIcpYI+4jJ
000021 M3goMDTgeB9KNhaEZ3Es/WPdUxv/ec+lw12eOekvhcumJNxmQxZj
000022 -----END CERTIFICATE-----
***** ***** Bottom of Data *****
```

NOTE: While copying and pasting this, please make sure not to edit it in any way.

If using this certificate to connect to a C:D Windows or C:D Unix, paste the certificate into a text editor; something like Notepad is recommended since it does not add formatting.



The screenshot shows a Notepad window titled "Untitled - Notepad" with a menu bar (File, Edit, Format, View, Help). The text inside the window is the same certificate content as shown in the previous block, but without the line numbers and column indicators. The text is as follows:

```
-----BEGIN CERTIFICATE-----
MIIDszCCApugAwIBAgIIYEvl4wACbX0wDQYJKoZIhvcNAQELBQAwZzELMAkGA1UE
BhMCVVMx CzAJBgNVBAGTAIRYMREwDwYDVQQHEwhSb2Nrd2FsbDEQMA4GA1UEChMH
U3VwcG9ydDEPMA0GA1UECXMgQWx0cmFuMRUwEwYDVQQDDAxaNjFGSVBTX2NlcnQw
HhcNMjEwMzEyMjIwNjI3WhcNNDIxMDA2MjIwNjI3WjBnMQswCQYDVQQGEwJVUzEL
MAKGA1UECBMCFgxEtAPBgNVBACTCFJvY2t3YWxsMRAwDgYDVQQKEwdTdXBwb3J0
MQ8wDQYDVQQLEwZBbHRyYW4xFTATBgNVBAMMDFo2MUZJUUFNfY2VydDCCASIdQYJ
KoZIhvcNAQEBBQADggEPADCCAQoCggEBAJhtEsdSbXv0roanvesy8iznPMGfIoNt
33m+sYAogrg0Y6KGUeV32Z06gEMZe3+0PXzliTAtKMfg+8BKBALyzPVSCHx3KyVf
2y0R0Xn3ltYsOLa00sFrVnEE4rK3DzRFvPw3rM6zmtXk8X0Ydwc6KVCJrRjWlRk
+w7kcfBK6o5885dEjCKJk38Xgao/zXtYJQi5fm5wuvHr9T0KDuAjt3SH+bmQqiHQ
PvZTw0uyQaWVhF+8N2JjD5V/x7b4kqpp1bQp6ohNXdTYlSGus7df2Q9a2MumXcwB
WvIa+6kIwXm1GcoTzBLhkuK1g9djmdejjG4jNi3sLt3HGRkfKbdkh8CAwEAAANj
MGEwHQYDVROOBByEFDbf9Qsf7JIEqvBDT0AgMclrEagfMB8GA1UdIwQYMBaAFDbf
9Qsf7JIEqvBDT0AgMclrEagfMA4GA1UdDwEB/wQEAwIB9jAPBgNVHRMBAf8EBTAD
AQH/MA0GCSqGSIb3DQEBCwUAA4IBAQCQ+1iptHMRil7S9FDr5FQaakGvawWZTDJt
DfRxGZ5JrAeIFTRx3151xZgzfnqP1y2E4oR0nuWTBnqINi8gWMARzcwAbMVax/jH
nBkt4VYZKwYUEYfDKP5vah3EY54LASCQC3q9DgT5RRKFFqvUXGIgBceCmB3Haqm4
IRzLI9Lli+3Atp7qIRFaQXTUTI3Ggpc0OzwW+wTvNCuCo+DPDA4qrURckS/HaciE
A7bvdVBZ7NHQZ3nlsuxfcIbLdve6VUuyWUWSay5h4LaFvGO16blt0KvIcpYI+4jJ
M3goMDTgeB9KNhaEZ3Es/WPdUxv/ec+lw12eOekvhcumJNxmQxZj
-----END CERTIFICATE-----
```

This file can be sent to the remote as a text file via Connect:Direct or email.

Copy Certificate as HFS file Using Connect:Direct

Connect:Direct can be used to copy the exported certificate file as a HFS file to either to a z/OS file or to Windows file:

To copy to a z/OS file, specify DCB=(RECFM=FB,LRECL=80,BLKSIZE=27920,DSORG=PS) for the destination file:

```
HFSCOPYZ PROCESS PNODE=CDZ_node SNODE=CDZ_node
STEP01 COPY FROM (PNODE FILE='/u/userid/TestCert.crt' -
                  DATATYPE=TEXT PERMISS=777 DISP=SHR) -
TO (SNODE FILE=CD.CDZ.CERT DISP=NEW -
   DCB=(RECFM=FB,LRECL=80,BLKSIZE=27920,DSORG=PS))
```

This file can then be sent to a C:D Windows or C:D Unix or be used to import into another z/OS key database.

To copy to a Windows file, use the following:

```
HFSCOPYW PROCESS PNODE=CDZ_node SNODE=CDWin_node
STEP01 COPY FROM (PNODE FILE='/u/userid/TestCert.crt' -
                  DATATYPE=TEXT PERMISS=777 DISP=SHR) -
TO (SNODE FILE='C:\Certs\CDZ_cert.txt' DISP=NEW)
```

To copy to a Unix file, use the following:

```
HFSCOPYW PROCESS PNODE=CDZ_node SNODE=CDUnix_node
STEP01 COPY FROM (PNODE FILE='/u/userid/TestCert.crt' -
                  DATATYPE=TEXT PERMISS=777 DISP=SHR) -
TO (SNODE FILE='/path/certs/CDZ_cert.txt' DISP=NEW)
```

The file can then be sent to the remote as a text file via Connect:Direct or email.

Copy and Paste Using Open Editor (in USS)

NOTE: Open Editor is typically kept with the User Applications; if this is different in your location, you will need to find Open Editor and go to that location.

1. Type **U** to go into **User Apps** on the command line (or however Open Edition is defined at your site) and hit **Enter**.
2. Type **OE** on the Option line and hit **Enter** to enter Unix Systems Services.

CSG USER FUNCTIONS		
OPTION ==> OE		SCROLL ==> CSR
SM	SMP/E	- SMP/E Dialogs
IP	IPCS	- Interactive problem Control Facility
P	PDS	- PDS Application
R	RACF	- Resource Access Control Facility
CD	CD	- CSGCD - production Connect Direct
MSG	MSG	- Connect:Direct Message Lookup
UCD	UCD	- CD.HOST5200 Personal C:D IUI
OE	UNIX	- Unix System Services
DIST	DIST	- Connect:* Distribution System (Auth required)
XD	XDC	- Extended Debugging Controller
F	FLMGR	- File Manager for Z/OS
DT	DITTO	- Ditto for Mvs
DB	DEBUG	- Debug Tool Utility functions
E or M	ENTERPRISE	- Connect:Enterprise User Interface
MX	MXI	- MXI Version 4.3 GenLevel 050126
X	EXIT	- Terminate User Menu

3. Either **ISHELL (OPENMVS ISPF SHELL)** or **EDIT (CREATE OR CHANGE SOURCE DATA)** can be used:

A. ISHELL (IS) - OPENMVS ISPF SHELL

- 1) Type **IS** on the Option line and hit **Enter**.

OPENEDITION/MVS PRIMARY OPTION MENU		
OPTION ==> IS		SCROLL ==> CSR
		USERID - USERID
		TIME - 13:55
OB	BROWSE	- DISPLAY SOURCE DATA
OE	EDIT	- CREATE OR CHANGE SOURCE DATA
IS	ISHELL	- OPENMVS ISPF SHELL
X	EXIT	- exit openedition/mvs

- 2) If the Unix pathname is not correct, enter the correct pathname, then hit **Enter**.

UNIX System Services ISPF Shell		
Enter a pathname and do one of these:		
- Press Enter.		
- Select an action bar choice.		
- Specify an action code or command on the command line.		
Return to this panel to work with a different pathname.		
		More: +
<u>/u/userid/</u>		

- 3) The **Directory List** for the specified Unix pathname will be displayed.

```
File  Directory  Special_file  Commands  Help
-----
                        Directory List

Select one or more files with / or action codes.  If / is used also select an
action from the action bar otherwise your default action will be used.  Select
with S to use your default action.  Cursor select can also be used for quick
navigation.  See help for details.
EUID=10110016   /u/userid/
Type  Filename                                     Row 1 of 163
_ Dir      .
_ Dir      ..
_ Dir      .InstallAnywhere
_ File     .sh_history
_ File     a.out
_ File     asgard.crt
_ File     benzwin1.crt
_ File     benzwin2.crt

Command ==> _____
```

- 4) Scroll down to the file needing to be copied (i.e., the exported certificate filename) and type a **B** (Browse) on the Command line next to that file, then hit **Enter** (**E** for Edit can also be used but be careful not to edit the file).

```
File  Directory  Special_file  Commands  Help
-----
                        Directory List

Select one or more files with / or action codes.  If / is used also select an
action from the action bar otherwise your default action will be used.  Select
with S to use your default action.  Cursor select can also be used for quick
navigation.  See help for details.
EUID=10110016   /u/userid/
Type  Filename                                     Row 59 of 163
_ File     Test_File.zip
_ File     tempfile.txt
b File     TestCert.crt
_ File     testcert.p12
_ File     testcerx.crt
_ File     TestCSR
_ File     TestCSR1
_ File     testjeff.crt
_ File     test1.csr
_ File     test50.kdb
_ File     test50.rdb
_ File     unixtest.crt

Command ==> _____
```

B. EDIT (OE) – CREATE OR CHANGE SOURCE DATA

- 1) Type **OS** on the Option line and hit **Enter**.

```
OPENEDITION/MVS PRIMARY OPTION MENU

OPTION ==> OE                                SCROLL ==> CSR

USERID - USERID
TIME   - 13:55

OB  BROWSE  - DISPLAY SOURCE DATA
OE  EDIT    - CREATE OR CHANGE SOURCE DATA
IS  ISHELL  - OPENMVS ISPF SHELL
X   EXIT    - exit openedition/mvs
```

- 2) The **z/OS UNIX Directory List** for the specified Unix pathname will be displayed.

Menu Utilities View Options Help							
z/OS UNIX Directory List					Row 1 to 15 of 163		
Command ==>					Scroll ==> PAGE		
Pathname . : /u/userid							
EUID . . : 10110016							
Command	Filename	Message	Type	Permission	Audit	Ext	Fmat
_____	.		Dir	rwxr----	fff--		----
_____	..		Dir	rwxrwxr-x	fff--		----
_____	.sh_history		File	rw-----	fff--	--s-	----
_____	.InstallAnywher		Dir	rwx-----	fff--		----
_____	a.out		File	rwxr-xr-x	fff--	--s-	----
_____	asgard.crt		File	rwxr-xr-x	fff--	--s-	----
_____	benzwin1.crt		File	rwxr-xr-x	fff--	--s-	----
_____	benzwin2.crt		File	rwxr-xr-x	fff--	--s-	----
_____	cdwin46.crt		File	rwxr-xr-x	fff--	--s-	----
_____	cdwin47.crt		File	rwxr-xr-x	fff--	--s-	----

- 3) Scroll down to the file needing to be copied (i.e., the exported certificate filename) and type a **B** (Browse) on the Command line next to that file, then hit **Enter** (E for Edit can also be used but be careful not to edit the file).

Menu Utilities View Options Help							

z/OS UNIX Directory List					Row 1 to 15 of 163		
Command ==>					Scroll ==> PAGE		
Pathname . : /u/userid							
EUID . . : 10110016							
Command	Filename	Message	Type	Permission	Audit	Ext	Fmat

_____	test_file.zip		File	rwxr-xr-x	fff--	--s-	----
_____	tempfile.txt		File	rw-r--r--	fff--	--s-	----
b	TestCert.crt		File	rw-r--r--	fff--	--s-	----
_____	TestFile.txt		File	rw-----	fff--	--s-	----
_____	TestCSR		File	rw-r--r--	fff--	--s-	----
_____	Z46.crt		File	rw-----	fff--	--s-	----
_____	Z46.kdb		File	rw-----	fff--	--s-	----
_____	Z46.rdb		File	rw-----	fff--	--s-	----
_____	Z48.crt		File	rw-----	fff--	--s-	----
_____	Z48key.crt		File	rw-----	fff--	--s-	----
_____	Z50.crt		File	rw-----	fff--	--s-	----
_____	Z50.kdb		File	rw----rw-	fff--	--s-	----
_____	Z50.rdb		File	rw-----	fff--	--s-	----
_____	Z50PLX.crt		File	rw-----	fff--	--s-	----
_____	Z50PLX.kdb		File	rw-----	fff--	--s-	----

- ```
Menu Utilities Compilers Help

BROWSE /u/userid/TestCert.crt Line 0000000000 Col 001 080
Command ==> Scroll ==> PAGE
***** Top of Data *****
-----BEGIN CERTIFICATE-----
MIIDnTCCAwGwAwIBAgIIWIsGsAAI488wDQYJKoZIhvcNAQEFBQAwwXDELMAkGA1UE
BhmCVVMxMzA5BgNVBAgTAIRYMq8wDQYDVQQHEwZJcnZpbmcxDDAKBgNVBAoTA0lC
TTEQMA4GA1UECxMHU3VwcG9ydDEPMA0GA1UEAxMGSk1UZXRNOMB4XDTE3MTIwODIx
NDAMFfoXDQTlMDQyNDIeXNDAMFfowXDELMAkGA1UEBhmCVVMxMzA5BgNVBAgTAIRY
MQ8wDQYDVQQHEwZJcnZpbmcxDDAKBgNVBAoTA0lCTTEQMA4GA1UECxMHU3VwcG9y
dDEPMA0GA1UEAxMGSk1UZXRNOMIIBIJANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKc
AQEAWC114vpYcKKxrRxxKvqaAj99bx8i8JkTIIOJwlZDxlchftrQGJ5ICvUMHKMJ0
Detvu2c5Eap+gERC31jarN0hdwUlDP7uC+sIAulf+k4FtoNZjs0jvPzGRDZP+Fon
11YTzhmQrJhh1lrVtX5zcWlF6GSOVqt6FHH8PBvgZDeKpDYszuh45Pbqacx4rbv5
xEksTNCPLRpLdKAeggoq53U8djLnVfnGggj9oVcVtUr96pBaoPH69xGt2TnjfpDr
kBui25k59Lw6QuHBFBKzdMpiq9KNcs+J5TSvdTF3/dKQRjlnpHz5IMB44qonIrd2t
SY6zRFrmocTLGShpy3OGRHdTgwIDAQABO2MwYTAdbGNVHQ4EFgQUdp8MqdQg93+S
mLK1FGY0fHmFXBUwDVR0jBBgwFoAUDP8MqdQg93+SmLK1FGY0fHmFXBUwDgYD
VR0PAQH/BAQDAgH2MA8GA1UdEwEB/wQFMAMBaf8wDQYJKoZIhvcNAQEFBQAwdggEB
ABrruhAUswKNXGZr1/FoAlF4paOzf7p7NVQTPYXGT06bdGmqepN0nOPCs6EEIm7w
4YxwZJlh5K971XrV62587S3r+EHPkNWfyIufcbGFsH/PBXLfO9TBZv1D5vtbzx4h
blodpdjrLY2fZmAR9YJipWj2zXd+ORxEMv08zRIYeJv+rUhFrw3LmCenuKpeJ6Wo9
GlqYmpcbMRjkCNapQ/cMxvEq8gTEFROBFobcPAWk4KLcwvqS8xt8MoI3fvLNJdaK
JnLRhDaLSgchp0Y01HEXDqjXYZQfs2927uLWTK1yrdrHbRKZ6EBOnamZuTP6G6mBu
gHGWrnd1jZFZjgX0c4R9WC=
-----END CERTIFICATE-----
***** Bottom of Data *****
```

- ```

EDIT          USERID.TESTCERT.CERT          Columns 00001 00072
Command ==>          Scroll ==> CSR
*****
***** Top of Data *****
000001  -----BEGIN CERTIFICATE-----
000002  MIIDnTCCAoWgAwIBAgIIWiSgSAAI488wDQYJKoZIhvcNAQEFBQAwXDELMAkGA1UE
000003  BhMCVVMxChZABgNVBAGTAIRYMQ8wDQYDVQQHEwZJcnZpbmcxDDAKBgNVBAoTA0lC
000004  TTEQMA4GA1UECzMHU3VwcG9ydDEPMA0GA1UEAxiMGSk1UZXN0M0B4XDTE3MTIwODIx
000005  NDAMWFOxDTQ1MDQyNDIwNDAMWFOwXDELMAkGA1UEBhMCVVMxChZABgNVBAGTAIRY
000006  MQ8wDQYDVQQHEwZJcnZpbmcxDDAKBgNVBAoTA0lCTTEQMA4GA1UECzMHU3VwcG9y
000007  dDEPMA0GA1UEAxiMGSk1UZXN0M0IIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKC
000008  AQEAWC114vpYcKXkrRkKvqAaj99bx8i8JkTIIoJwlZDxlchftrQGJ5ICvUMHKMJ0
000009  Detvu2c5EaP+gERc31jarN0hdwUldP7uC+sIAulf+k4FtoNZjs0jvPzGRDZP+FoN
000010  l1YTzhmQrJhh1lrVtX5zCw1F6GSOVqt6FHh8PBvgZDeKpDYszuh45Pbqacx4rBv5
000011  xEksTNCPlRpLdKAeggoq53U8djLnVfnGggj9oVcVtUr96pBaoPH69xGt2TnjfpDr
000012  kBu125k59Lw6QuHbFKZdMpiq9KNcs+J5TSvdTF3/dKqRjlnphZ5IMb44qonIrd2t
000013  SY6zRfrMocLTGShpy3OGRHd7gIDAQABo2MwYTAdbBgNVHQ4EFgQUdP8mQdQg93+S
000014  mLK1FGY0fHmFXBUwHwYDVR0lTBBGwFoAUDP8MqdQg93+SmLK1FGY0fHmFXBUwDgYD
000015  VR0PAQH/BAQDAgH2MA8GA1UdEwEB/wQFMAMBAf8wDQYJKoZIhvcNAQEFBQADggEB
000016  ABrruhAuswKNXGZr1/FoAlF4paOzf7p7NVQTPyXGT06bdGmqepN0nOpCs6EEIm7w
000017  4YxwZJ1h5K971XrV62587S3r+EHpKNWfyIuFcbGFsH/PBXLf09TBZv1D5VtbZx4h
000018  bl opdjrLY2fZmAR9YJipWj2zXd+oRxEV08zRIyEjv+ruUhFrw3LmCenuKpEJ6Wo9
000019  GlqYMPcbMRjKcNapQ/cMxvEq8gTEFROBFObcPAWk4K1CWvqS8xt8MoI3fvLNJdaK
000020  JnLRhDaSgchp0Y01HEXhdqjXYZQfs2927uLWTK1yrdHbRKZ26EBOnamZuTP6G6mB
000021  gHGWrnd1jZfZjgX0c4R9Wc0=
000022  -----END CERTIFICATE-----
*****
***** Bottom of Data *****

```

Or to a Notepad text file (Notepad is recommended since it does not add formatting to the file).

```

-----BEGIN CERTIFICATE-----
MIIDnTCCAowGAwIBAgIIWisGsAAI488wDQYJKoZIhvcNAQEFBQAwXDELMAkGA1UE
BhMCVVMxMzA1BgNVBAGTA1RYMQ8wDQYDVQQHEwZJcnZpbmcxDDAKBgNVBAoTA01C
TTEQMA4GA1UECxmHU3VwcG9ydDEPMA0GA1UEAxMGSk1UZXN0MIIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCA
QEAWC114vpYcKXkrRxKvqaAj99bx8i8JkTIIoJw1ZDx1chftrQGJ5ICvUMHKMJ0
Detvu2c5EaP+gERc31jarN0hdwU1dP7uC+sIAu1f+k4FtoNZjs0jvPzGRDZP+FoN
l1YTzhmQrJhh11rVtX5zCw1F6GSOVqt6FHH8PBvgZDeKpDYszuh45Pbqacx4rBv5
xEksTNCPLRpLdKAeggoq53U8djLnVfnGggj9oVcVtUr96pBaoPH69xGt2TnjfpDr
kBui25k59Lw6QuHBFKzdMpiq9KNcs+J5TSvdTF3/dKqRjlnPhz5IMb44qonIrd2t
SY6zRFRmOoctLGShpy3OGRHdTgwIDAQABo2MwYTAdbGNVHQ4EFgQUdP8MqdQg93+S
mLK1FGY0fHmFXBUwHwYDVR0jBBgwFoAUDP8MqdQg93+SmLK1FGY0fHmFXBUwDgYD
VR0PAQH/BAQDAgH2MA8GA1UdEwEB/wQFMAMBAf8wDQYJKoZIhvcNAQEFBQADggEB
ABrruhAUsWKNXGZr1/FoA1F4paOzf7p7NVQTPYXGT06bdGmqepN0nOPcS6EEIm7w
4YxwZJ1h5K971XrV62587S3r+EHPkNWfyIuFcbGFsH/PBXLf09TBZv1D5VtbZx4h
blodpjrLY2fZmar9YJipWj2zXd+oRxEv08zRIyEjV+rUhFrw3LmCenuKpEJ6Wo9
G1qYMpcbMRjKcNapQ/cMxvEq8gTEFROBfObcPAWk4K1CWvqS8xt8MoI3fvLNJdaK
JnLRhDaLSgchp0Y01HEXdqjXYZQfs2927uLWTK1yrdHbRKZ6EBOnamZuTP6G6mBu
gHGWrnD1jZFZjgX0c4R9Wc0=
-----END CERTIFICATE-----

```

- Whether z/OS or Notepad file, save the file. This file can be sent to the remote as a text file via Connect:Direct or email.

Loading Private Key (For Use as Site Certificate) into gskkyman

Go to **TSO ISPF Option 6** and enter **OMVS** or enter **TSO OMVS** from the command line. Once in **OMVS**, do a **ls** command; you should see the file for the private key (key file) you want to load (key file should be a .p12 file).

```

$ ls
Z62.crt      Z60.crt      cdwin48.p12  test.rdb
Z61.kdb      Z60.kdb      test.csr     TestCert.crt
Z61.rdb      Z60.rdb      test.kdb
$
==>

```

Enter **gskkyman** and hit **Enter**.

```

IBM
Licensed Material - Property of IBM
5650-ZOS Copyright IBM Corp. 1993, 2015
(C) Copyright Mortice Kern Systems, Inc., 1985, 1996.
(C) Copyright Software Development Group, University of Waterloo, 1989.

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

IBM is a registered trademark of the IBM Corp.

$

==> gskkyman

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

```


This brings up the **Database Menu**:

```
Database Menu

1 - Create new database
2 - Open database
3 - Change database password
4 - Change database record length
5 - Delete database
6 - Create key parameter file
7 - Display certificate file (Binary or Base64 ASN.1 DER)

11 - Create new token
12 - Delete token
13 - Manage token
14 - Manage token from list of tokens

0 - Exit program

Enter option number:
==>
```

1. Enter **2** for **Open database** and hit **Enter**.
2. Enter **key database name**, including **.kdb** extension and hit **Enter**.
3. Enter database password and hit **Enter**.

```
Database Menu

1 - Create new database
2 - Open database
3 - Change database password
4 - Change database record length
5 - Delete database
6 - Create key parameter file
7 - Display certificate file (Binary or Base64 ASN.1 DER)

11 - Create new token
12 - Delete token
13 - Manage token
14 - Manage token from list of tokens

0 - Exit program

Enter option number: 2
Enter key database name (press ENTER to return to menu): test.kdb
Enter database password (press ENTER to return to menu):
```

This should bring up the **Key Management Menu** panel:

```
Key Management Menu

Database: /u/userid/test.kdb
Expiration: None
Type: non-FIPS

1 - Manage keys and certificates
2 - Manage certificates
3 - Manage certificate requests
4 - Create new certificate request
5 - Receive requested certificate or a renewal certificate
6 - Create a self-signed certificate
7 - Import a certificate
8 - Import a certificate and a private key
9 - Show the default key
10 - Store database password
11 - Show database record length

0 - Exit program

Enter option number (press ENTER to return to previous menu):
==>
```


1. Enter **8** for **Import a certificate and a private key**.
2. Enter **import file name** using the certificate file you just copied to HFS (in this example, **cdwin48.p12**).
3. Enter **import file password** which is the password created when the certificate was exported.
4. Enter the **label**. This can be anything you want; it is the label that will be used to identify this certificate to the key database.

```

1 - Manage keys and certificates
2 - Manage certificates
3 - Manage certificate requests
4 - Create new certificate request
5 - Receive requested certificate or a renewal certificate
6 - Create a self-signed certificate
7 - Import a certificate
8 - Import a certificate and a private key
9 - Show the default key
10 - Store database password
11 - Show database record length

0 - Exit program

Enter option number (press ENTER to return to previous menu): 8
Enter import file name (press ENTER to return to menu): cdwin48.p12
Enter import file password (press ENTER to return to menu):
Enter label (press ENTER to return to menu): cdwin48

Certificate and key imported.

Press ENTER to continue.
===>

```

5. Press **Enter** to continue. This returns you to the **Key Management Menu** panel.

The certificate now needs to be made trusted; select **1** to **Manage keys and certificates**.

```

Key Management Menu

Database: /u/userid/test.kdb
Expiration: 2045/04/24 16:16:41

1 - Manage keys and certificates
2 - Manage certificates
3 - Manage certificate requests
4 - Create new certificate request
5 - Receive requested certificate or a renewal certificate
6 - Create a self-signed certificate
7 - Import a certificate
8 - Import a certificate and a private key
9 - Show the default key
10 - Store database password
11 - Show database record length

0 - Exit program

Enter option number (press ENTER to return to previous menu):
===> 1

```

1. Select the number corresponding to the label you just imported (**2** is chosen in this example).

```

Key and Certificate List

Database: /u/userid/test.kdb

1 - TestCert
2 - cdwin48

0 - Return to selection menu

Enter label number (ENTER to return to selection menu, p for previous list):
===> 2

```

2. Select **4** to **Set certificate trust status** and hit **Enter**.
3. Select **1** (if trusted) and hit **Enter**.

```
Key and Certificate Menu
Label: cdwin48

1 - Show certificate information
2 - Show key information
3 - Set key as default
4 - Set certificate trust status
5 - Copy certificate and key to another database/token
6 - Export certificate to a file
7 - Export certificate and key to a file
8 - Delete certificate and key
9 - Change label
10 - Create a signed certificate and key
11 - Create a certificate renewal request

0 - Exit program

Enter option number (press ENTER to return to previous menu): 4
Enter 1 if trusted, 0 if untrusted (press ENTER to return to menu): 1
Record updated.
Press ENTER to continue.
===>
```

4. Press **Enter** to continue. You will be returned to the **Key and Certificate Menu** panel.

If using this key as an alternate site certificate, you can exit gskkyman at this point and go into your Secure+ PARMFILE and enter the label of this certificate on **Certificate Label** on the remote node that will be using it.

However, if you will be using this certificate as the default (primary) certificate for your local node, select **3** to **Set key as default** and hit **Enter**.

NOTE: Beginning in C:D z/OS 6.0, a default certificate can be used if no label is specified for **Certificate Label** in the Secure+ PARMFILE. To use that feature in the Secure+ PARMFILE, a default certificate must be defined in the keyring / key database.

```
Key and Certificate Menu
Label: cdwin48

1 - Show certificate information
2 - Show key information
3 - Set key as default
4 - Set certificate trust status
5 - Copy certificate and key to another database/token
6 - Export certificate to a file
7 - Export certificate and key to a file
8 - Delete certificate and key
9 - Change label
10 - Create a signed certificate and key
11 - Create a certificate renewal request

0 - Exit program

Enter option number (press ENTER to return to previous menu): 3
Default key set.
Press ENTER to continue.
===>
```

You can exit gskkyman at this point and go into your Secure+ PARMFILE and enter the label of this certificate for **Certificate Label** in the local node.

Exporting Private Key from gskkyman to Use as Site Certificate

On the mainframe, go to **TSO ISPF Option 6** and enter **OMVS** (or enter **TSO OMVS** on the command line) and hit **Enter**.

Enter **gskkyman** and hit **Enter**.

```
IBM
Licensed Material - Property of IBM
5650-ZOS Copyright IBM Corp. 1993, 2015
(C) Copyright Mortice Kern Systems, Inc., 1985, 1996.
(C) Copyright Software Development Group, University of Waterloo, 1989.

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

IBM is a registered trademark of the IBM Corp.

$

====> gskkyman
```

The **Database Menu** panel should be displayed:

```
Database Menu

1 - Create new database
2 - Open database
3 - Change database password
4 - Change database record length
5 - Delete database
6 - Create key parameter file
7 - Display certificate file (Binary or Base64 ASN.1 DER)

11 - Create new token
12 - Delete token
13 - Manage token
14 - Manage token from list of tokens

0 - Exit program

Enter option number:
====>
```

1. Enter **2** for **Open database**.
2. Enter **key database name**, including **.kdb** extension
3. Enter **database password** and hit **Enter**.

```
Database Menu

1 - Create new database
2 - Open database
3 - Change database password
4 - Change database record length
5 - Delete database
6 - Create key parameter file
7 - Display certificate file (Binary or Base64 ASN.1 DER)

11 - Create new token
12 - Delete token
13 - Manage token
14 - Manage token from list of tokens

0 - Exit program

Enter option number: 2
Enter key database name (press ENTER to return to menu): test.kdb
Enter database password (press ENTER to return to menu):
```

This will bring up the **Key Management Menu** panel.

Enter **1** for **Manage keys and certificates**.

```
Key Management Menu

Database: /u/userid/test.kdb
Expiration: 2045/04/24 16:16:41

1 - Manage keys and certificates
2 - Manage certificates
3 - Manage certificate requests
4 - Create new certificate request
5 - Receive requested certificate or a renewal certificate
6 - Create a self-signed certificate
7 - Import a certificate
8 - Import a certificate and a private key
9 - Show the default key
10 - Store database password
11 - Show database record length

0 - Exit program

Enter option number (press ENTER to return to previous menu):
==> 1
```

Select the number corresponding to the **Label** for your site certificate being exported and hit **Enter**.

NOTE: If there are numerous labels, you may have to hit **Enter** until you see the label you need.

```
Key and Certificate List

Database: /u/userid/test.kdb

1 - TestCert
2 - Z52SCC

0 - Return to selection menu

Enter label number (ENTER to return to selection menu, p for previous list):
==> 1
```

Select **7 - Export certificate and key to a file** and hit **Enter**.

```
Key and Certificate Menu

Label: TestCert

1 - Show certificate information
2 - Show key information
3 - Set key as default
4 - Set certificate trust status
5 - Copy certificate and key to another database/token
6 - Export certificate to a file
7 - Export certificate and key to a file
8 - Delete certificate and key
9 - Change label
10 - Create a signed certificate and key
11 - Create a certificate renewal request

0 - Exit program

Enter option number (press ENTER to return to previous menu):
==> 7
```

Select **3 - Binary PKCS #12 Version 3** and hit **Enter**.

```
Export File Format

1 - Binary PKCS #12 Version 1
2 - Base64 PKCS #12 Version 1
3 - Binary PKCS #12 Version 3
4 - Base64 PKCS #12 Version 3

Select export format (press ENTER to return to menu):
==> 3
```

Enter **export file name** you wish to call the file (key file) being created (in this example, *testcert.p12* is used).

Enter **export file password** which will be used for this key file.

NOTE: This password will be used whenever importing this key file.

Re-enter **export file password**.

Enter **0** for export encryption.

NOTE: If the key database is a FIPS database, option **1** for **strong encryption** must be selected.

```
Export File Format

1 - Binary PKCS #12 Version 1
2 - Base64 PKCS #12 Version 1
3 - Binary PKCS #12 Version 3
4 - Base64 PKCS #12 Version 3

Select export format (press ENTER to return to menu): 3
Enter export file name (press ENTER to return to menu): testcert.p12
Enter export file password (press ENTER to return to menu):
Re-enter export file password:
Enter 1 for strong encryption, 0 for export encryption: 0

Certificate and key exported.

Press ENTER to continue.
==>
```

You should see the **Certificate and key imported** message if it is successful.

Hit **Enter** to continue, then enter **0** to exit gskkyman.

If you do a **ls** command, you should see the file you just created.

```
$ ls
Z51.crt      Z52.crt      test.csr      testcert.p12
Z51.kdb      Z52.kdb      test.kdb      tempfile.txt
Z51.rdb      Z52.rdb      test.rdb
$
==>
```

Exit out of **OMVS**.

Now the certificate file can be copied as a binary file to another z/OS system or to a Unix or Windows system to be used as a site certificate.

The following are examples of processes to copy the certificate file from HFS (USS) to a Windows, Unix or z/OS system:

Copy from HFS to a C:D Windows node:

```
HFSCOPYZ PROCESS PNODE=CDZ_node SNODE=CDWin_node
STEP01 COPY FROM (PNODE FILE='/u/userid/testcert.p12' -
                  DATATYPE=BINARY PERMISS=777 DISP=SHR) -
                TO (SNODE FILE='C:\Certs\testcert.p12' -
                  SYSOPTS="datatype(binary) xlate(no)" -
                  DISP=NEW)
```

Copy from HFS to a C:D Unix node:

```
HFSCOPYZ PROCESS PNODE=CDZ_node SNODE=CDUnix_node
STEP01 COPY FROM (PNODE FILE='/u/userid/testcert.p12' -
                  DATATYPE=BINARY PERMISS=777 DISP=SHR) -
                TO (SNODE FILE='/path/certs/testcert.p12' -
                  SYSOPTS=":datatype=binary:xlate=no:" -
                  DISP=NEW)
```

Copy from HFS to C:D z/OS node:

```
HFSCOPY1 PROCESS PNODE=CDZ_node SNODE=CDZ_node NOTIFY=&SYSUID
STEP01 COPY FROM (PNODE FILE='/u/userid/testcert.p12' -
                  DATATYPE=BINARY PERMISS=777 DISP=SHR) -
                TO (SNODE DSN=hlq.TESTCERT.P12,DISP=NEW -
                  DCB=(LRECL=0,BLKSIZE=23040,RECFM=U))
```

Depending on the systems involved, going directly from Windows or Unix to HFS may not work or be allowed. If not allowed, you can use the above example to copy it from HFS to C:D z/OS node, then copy from the mainframe to Windows or Unix using one of the following examples:

Copy from C:D z/OS to C:D Windows:

```
WINCOPY1 PROCESS PNODE=CDZ_node SNODE=CDWin_node NOTIFY=&SYSUID
STEP01 COPY FROM (PNODE DSN=hlq.TESTCERT.P12 DISP=SHR) -
                TO (SNODE FILE='C:\Certs\testcert.p12', DISP=NEW -
                  SYSOPTS="datatype(binary) xlate(no)")
```

Copy from C:D z/OS to C:D Unix:

```
WINCOPY1 PROCESS PNODE=CDZ_node SNODE=CDUnix_node NOTIFY=&SYSUID
STEP01 COPY FROM (PNODE DSN=hlq.TESTCERT.P12 DISP=SHR) -
                TO (SNODE FILE='/path/certs/testcert.p12', DISP=NEW -
                  SYSOPTS=":datatype=binary:xlate=no:")
```

Now go to the system where this was copied to and import the key. If loading into gskkyman on z/OS, refer to section **Loading Private Key (For Use as Site Certificate) into gskkyman**; if loading into Windows or Unix, see section **Importing Private Key Into IKEYMAN**.

Using IBM Key Manager (IKEYMAN)

IKEYMAN is a key-management utility included with C:D Windows or C:D Unix that is used on Windows and Unix platforms to create key databases, public and private key pairs, and certificate requests. When using IKEYMAN to create a certificate for use in Connect:Direct, make sure that you create the key database, create the certificate, and add the remote certificate to the key database.

NOTE: The Keystore is the same as the Key Database.

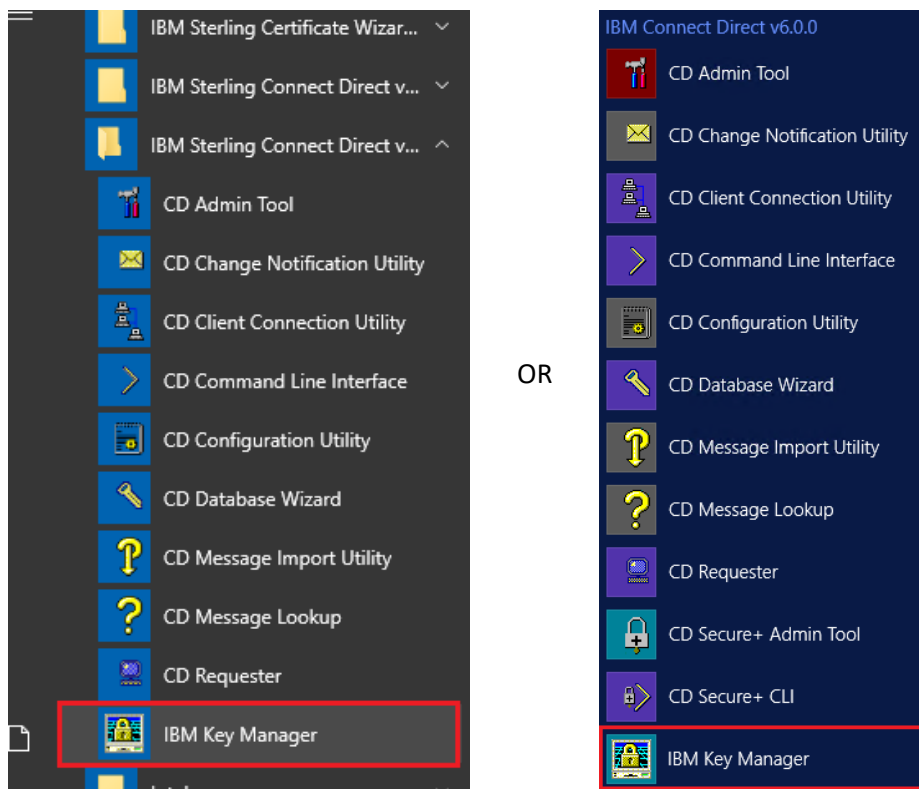
For further information about the IKEYMAN, refer to the [IBM Key Manager User's Guide](#), available here: **IKEYMAN Users Guide**.

NOTE: If you are using Connect:Direct Unix and do not have access to an X11 Graphical application (and therefore cannot bring up the IKEYMAN GUI), you will need to use the Command Line Interface to use IKEYMAN; please go to section **Using IKEYMAN From the Command Line Interface (CLI)**.

Creating a Key Database (Keystore) Using IKEYMAN

1. Start the **IKEYMAN GUI**.

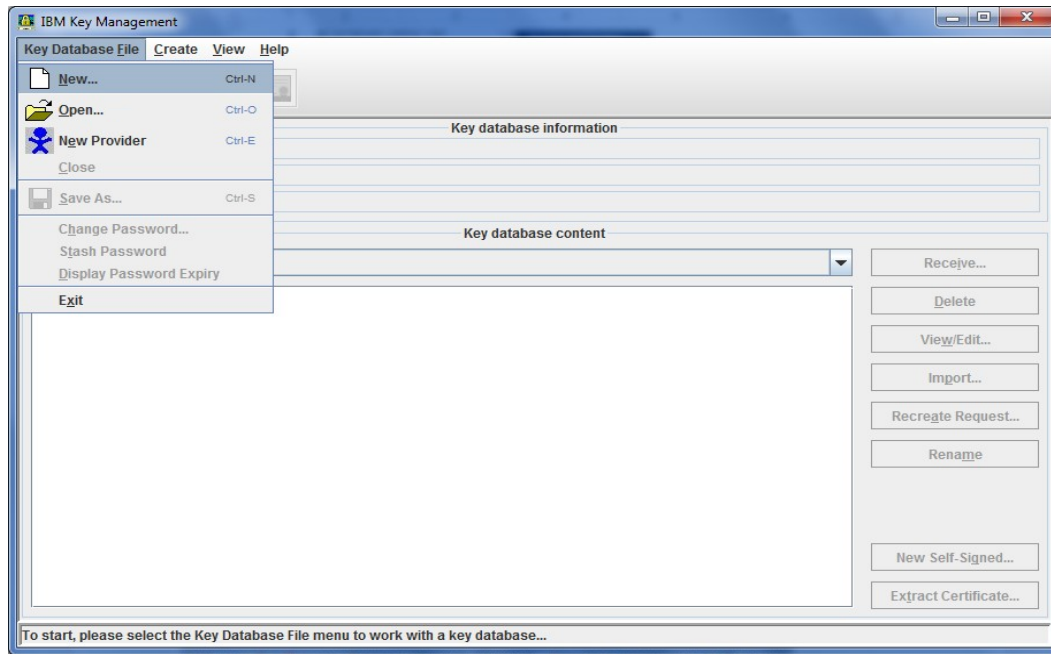
If using C:D Windows 4.7 or later, you would go to Start menu and open All Programs >> IBM Sterling Connect Direct v4.7.0 >> **IBM Key Manager**. In this example, the C:D Windows GUI version is used.



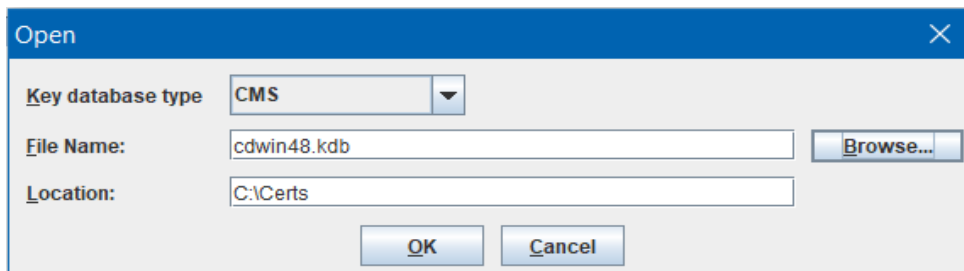
Note: If using C:D Unix 4.1 or later, the **ikeycmd** executable is located in the following folder, depending on the C:D Unix version that is installed and the C:D installed path:

/<cd_install_path>/jre/ibm-java-[i386|x86_64]-[70|80]/jre/bin/

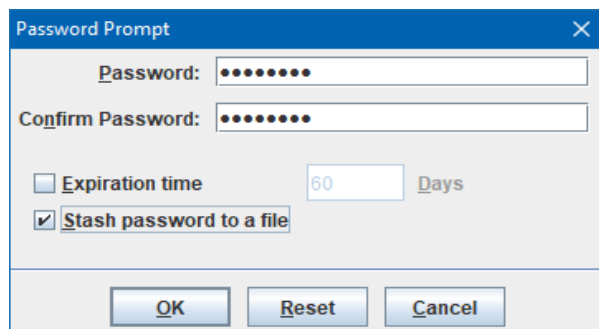
2. Select the **Key Database File** menu, select **New**.



3. Input the following into the New dialog:
Key database type: **CMS**
File Name: Input a Filename with a **.kdb** extension
Location: Enter the path for the file name to be saved
Select **OK**.



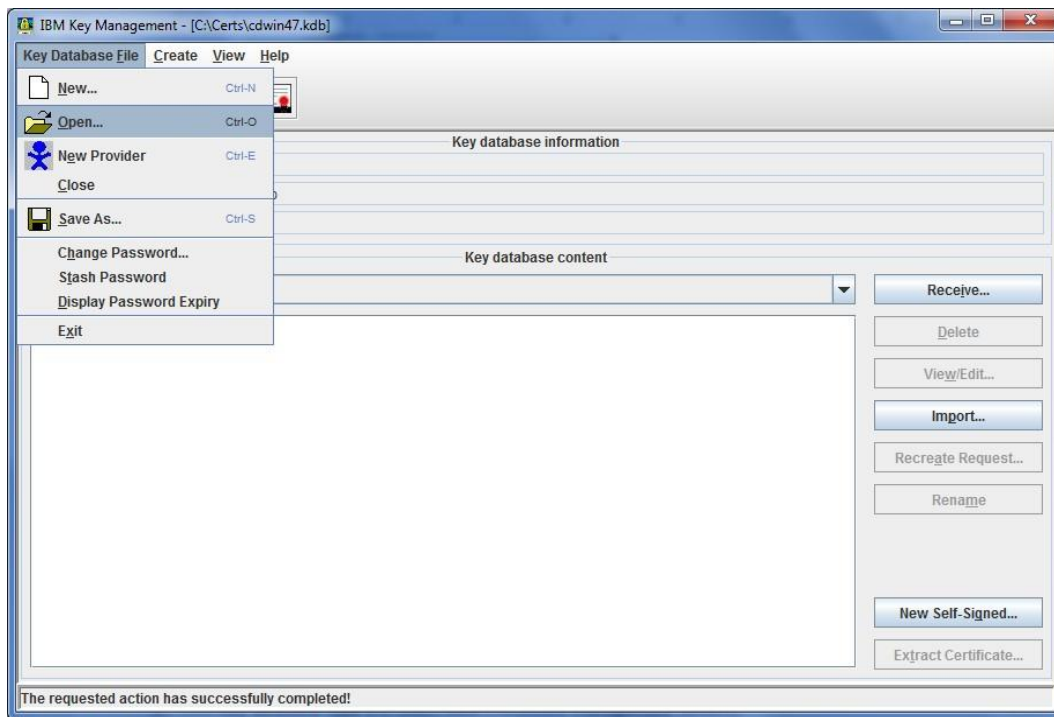
4. Enter a **Password** and **Confirm Password**. Select **Stash password to a file**. Select **OK**.
If you want the database to have an expiration date, select **Expiration time** and enter a value for **Days** (maximum value is 7300).



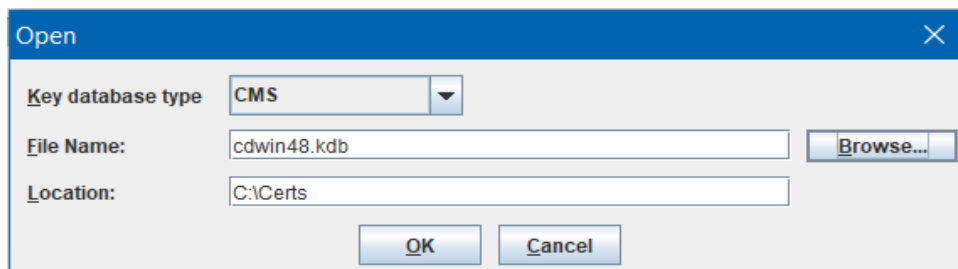
NOTE: Remember this password; if you forget this password, it is not recoverable.

Creating Certificates using IKEYMAN

1. If a new Key Database File has just been created and you are currently in that Key Database, skip to **step 7**. If not, start the **IKEYMAN GUI** (see page 47) and from the **Key Database File** menu, select **Open**. The Open window displays:



2. Select **Key database type** and select **CMS** (Certificate Management System).
3. Select **Browse** to navigate to the directory that contains the key database files.
4. Select the key database file where the certificate will be saved (**cdwin48.kdb** is used here).



5. Select **OK**. The Password Prompt window displays.
6. Type the password that was selected when the key database was created and select **OK**. The name of the key database file displays in the **File Name** field.
NOTE: You must have the passphrase for the key database; if you do not, it is not recoverable.
7. Either go to the **Create** menu (in pull-down menu) and select **New Self-Signed Certificate** or select the **New Self-Signed** button on the lower right to bring up the **Create New Self-Signed Certificate** window.
8. Enter the Certificate Information:
 - a. **Key Label** can be whatever label you want for this certificate. For example, **cdwin_label**. This value is used to identify the certificate in the database.

- b. **Version** should be **X509 V3** (X509 is the format used by Connect:Direct) - REQUIRED
- c. **Key Size** can be **1024, 2048** or **4096** (some systems may not support **4096**) - REQUIRED
- d. **Signature Algorithm** - encryption for the certificate; change to **SHA256WithRSA** - REQUIRED
NOTE: For C:D Windows 4.7 or C:D Unix 4.1 and earlier, keep default of **SHA1WithRSA**.
- e. **Common Name** - used with Client Authentication with Common Name included - OPTIONAL
- f. **Organization, Locality, State/Province, Zipcode, Country or region, and Subject Alternative Names** are all optional and merely provide additional information regarding the certificate
- g. **Validity Period** can be coded up to 7300 or accept the default of **365** - REQUIRED

NOTE: While this information is optional, either **Common Name** or **Organization** MUST be coded.

9. Select **OK**.

10. The **Personal Certificates** list shows the label of the self-signed personal certificate you created (an asterisk (*) next to a certificate means that it is the default certificate for your KEYSTORE).

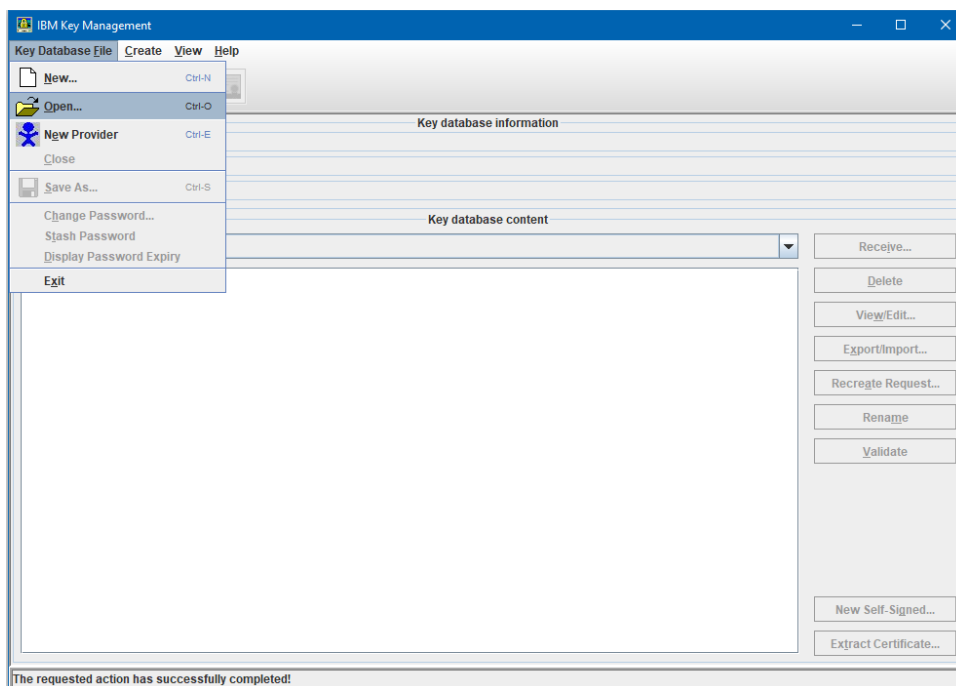
Building a Certificate Signing Request (CSR) Using IKEYMAN

To obtain a certificate signed by a Certificate Authority (CA), a Certificate Signing Request (CSR) is needed. However, this is handled differently in IKEYMAN than a self-signed certificate. The CSR will need to be generated, sent to the CA, then the certificates received from the CA will need to be received into your keystore to complete the creation of the keycert (the private key is built when the CSR is built, but the keycert isn't built until the certificates from the CA are added). The root and intermediate certificates received from the CA will also need to be added to the keystore to complete the keycert.

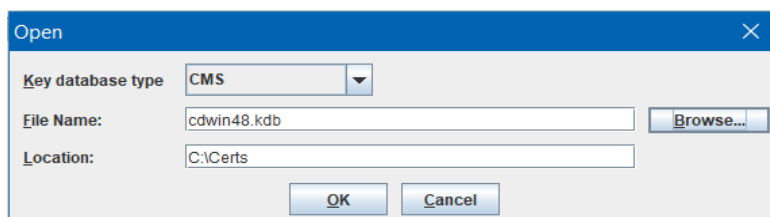
The IKEYMAN GUI instructions will be the same in either C:D Windows or C:D Unix.

Creating the Certificate Signing Request (CSR)

1. If you are already in your Key Database (or keystore), skip to **step 7**. If not, start the **IKEYMAN GUI** (see page 47) and from the **Key Database File** menu, select **Open**. The Open window displays:



2. Select **Key database type** and select **CMS** (Certificate Management System).
3. Select **Browse** to navigate to the directory that contains the key database files.
4. Select the key database file where the certificate will be saved (**cdwin48.kdb** is used here).

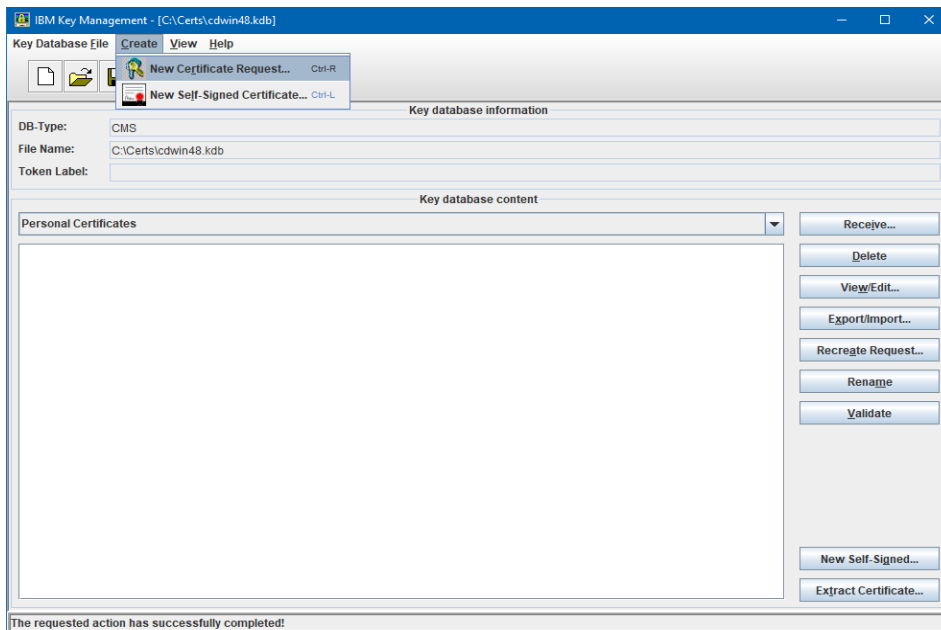


5. Select **OK**. The Password Prompt window displays.
6. Type the password that was selected when the key database was created and select **OK**. The name of the key database file displays in the **File Name** field.

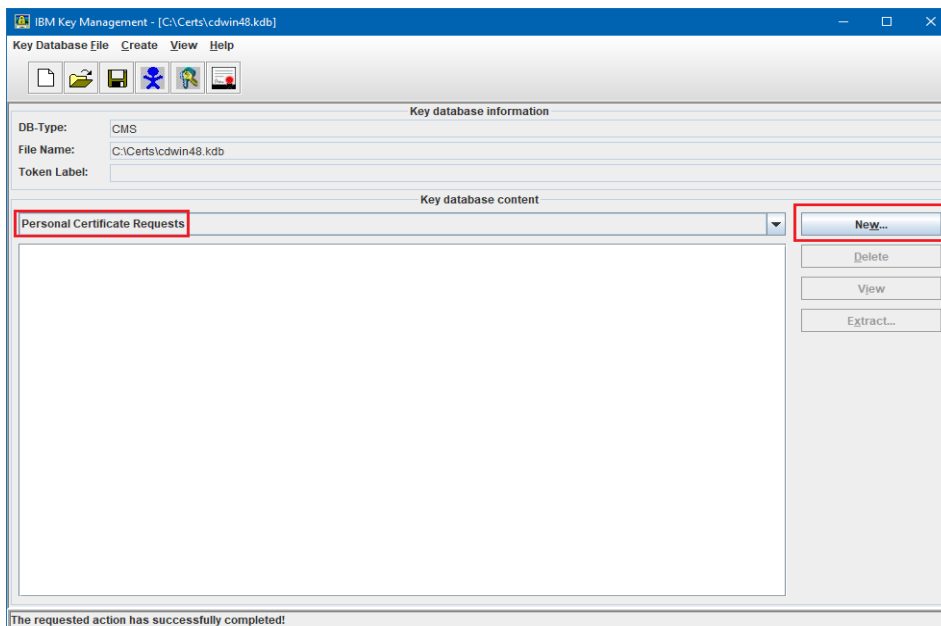
NOTE: You must have the passphrase for the keystore (key database); if you do not, it is not recoverable and a new keystore will need to be created.

7. There are three ways to get to the **Create New Key and Certificate Request** panel:

a. **Create** menu (in pull-down menu) and select **New Certificate Request**:



b. Select the **Personal Certificate Requests** view, then click the **New** button:



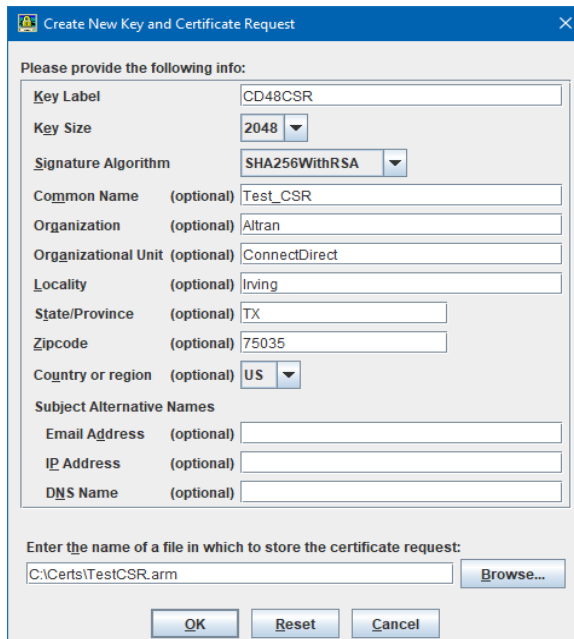
c. Hit **Ctrl-R**.

8. On the **Create New Key and Certificate Request** panel, enter the Certificate Information:

- Key Label** can be whatever label you want for this certificate. In this example, **CD48CSR**. This value is used to identify the certificate request in the keystore.
- Key Size** can be **1024**, **2048** or **4096** (some systems may not support **4096**) - REQUIRED
- Signature Algorithm** - encryption for the certificate (keep default of **SHA1WithRSA**) –REQUIRED
NOTE: For C:D Windows 4.8 or C:D Unix 4.2 and later, use **SHA256WithRSA**.
- Common Name** – used with Client Authentication with Common Name included - OPTIONAL
- Organization, Locality, State/Province, Zipcode, Country or region, and Subject Alternative Names** are all optional and merely provide additional information regarding the certificate

- f. At the bottom, enter the name for the CSR file that will be created. In this example, **C:\Certs\TestCSR.arm** was used. If you have multiple CSRs being created in your environment, It is a good idea to have a naming convention for this file; for example, a name with yymmdd, such as TestCSR_20181025.arm.

NOTE: This is the file that will be sent to your CA to have the certificate signed.



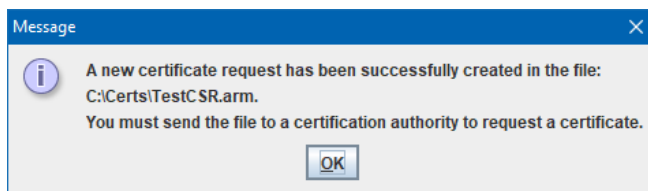
The dialog box titled "Create New Key and Certificate Request" contains the following fields and options:

- Key Label:** CD48CSR
- Key Size:** 2048
- Signature Algorithm:** SHA256WithRSA
- Common Name (optional):** Test_CSR
- Organization (optional):** Altran
- Organizational Unit (optional):** ConnectDirect
- Locality (optional):** Irving
- State/Province (optional):** TX
- Zipcode (optional):** 75035
- Country or region (optional):** US
- Subject Alternative Names:**
 - Email Address (optional):**
 - IP Address (optional):**
 - DNS Name (optional):**
- Enter the name of a file in which to store the certificate request:** C:\Certs\TestCSR.arm

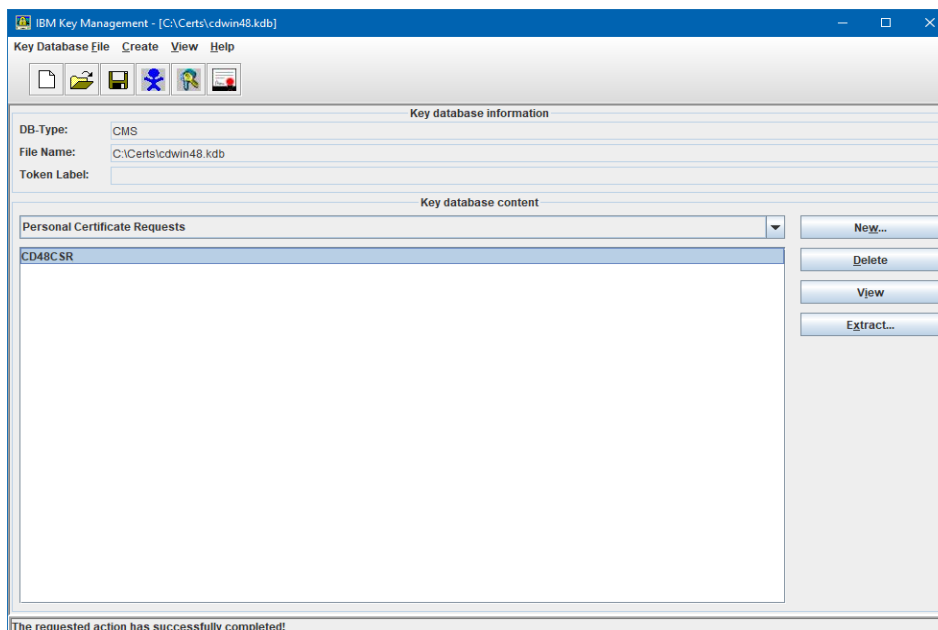
Buttons at the bottom: OK, Reset, Cancel, and a Browse... button next to the file name field.

9. Select **OK**.

10. You should see a pop-up with a confirmation message. Click **OK** to continue.



11. The **Personal Certificate Requests** list should now show the label of the CSR you created:

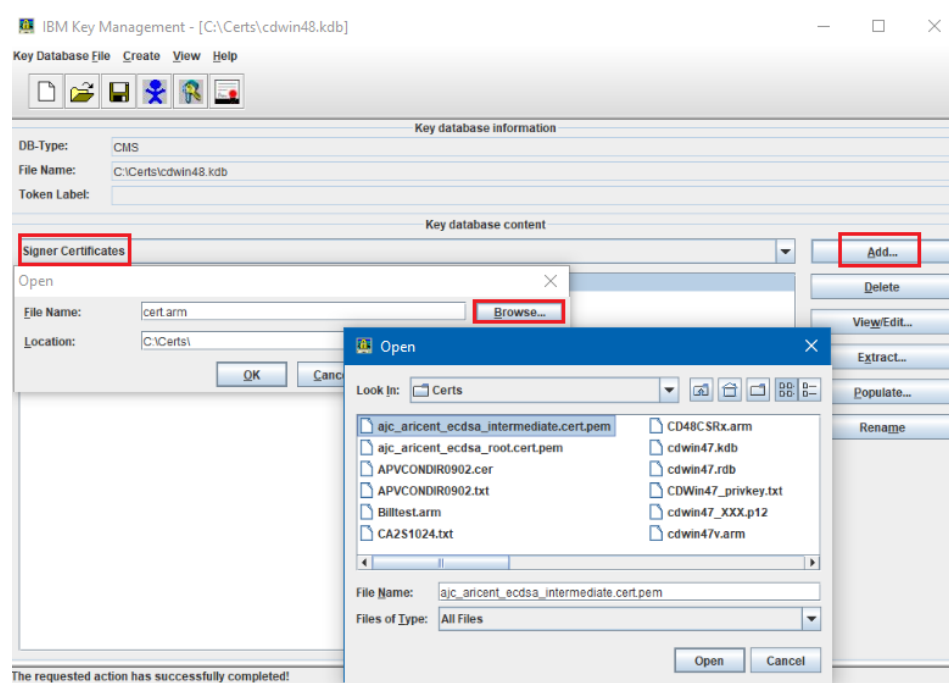


12. Send the request file you just created to your Certificate Authority (CA) to be signed.
From the CA, request an Apache style, x509, base64 encoded, PEM format signed server (public) certificate. Also, get the Root and Intermediate certificates from the CA.
13. You should receive a root, signed, and at least one intermediate certificate from the CA. The root and intermediate certificates should be sent to your trading partners (the signed server certificate is typically not sent to the trading partner unless there is a reason it is needed).

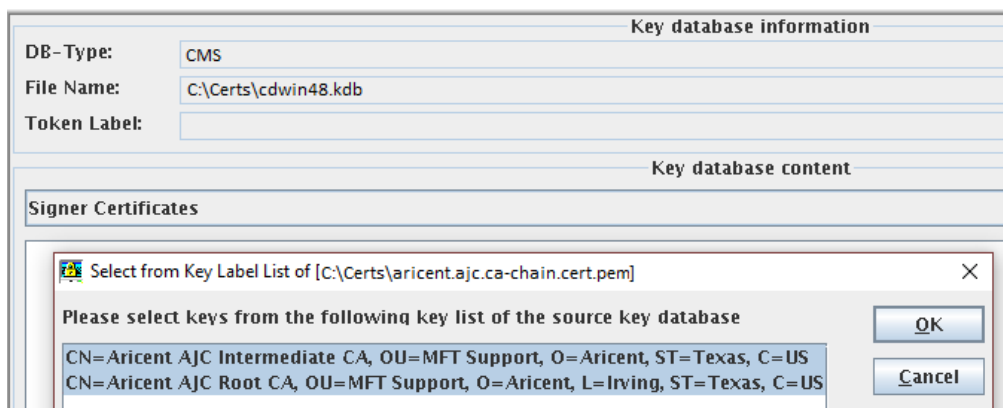
Adding the Intermediate and Root Certificates to the Keystore

1. Select the **Signer Certificates** view and click on **Add**. Navigate to the location where you stored the root and intermediate certificates you received from the CA. Select the file and click on **Open**, then click **OK**.

NOTE: If you received a file from your CA that contains something like “ca-chain.cert” in the name, select that file; this is a bundled file that contains both intermediate and root certificates from the CA.

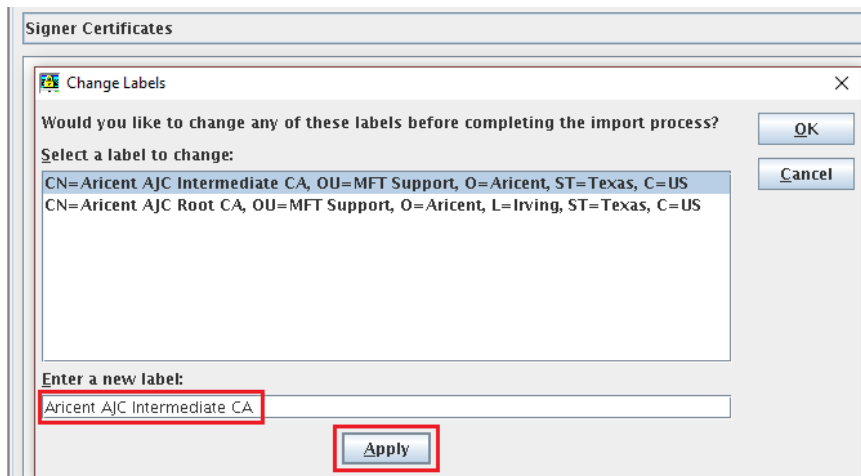


2. Click **Open** to select the correct file, then click **OK**.
3. If your file is a chained certificate file, you will see something like the following (if not a chained certificate file, skip to **Step 8**):

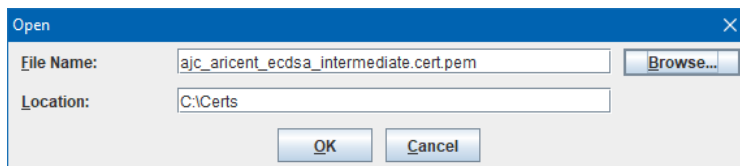


4. Select the Root and all Intermediates and click **OK**.

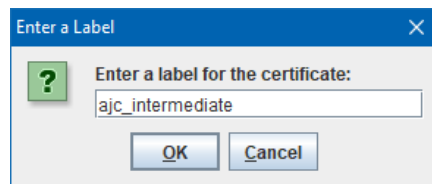
- The “default” label will be the CN for each certificate. You can change the label for the certificate as it will appear in the **Signer Certificates** view. Select each certificate in turn and provide your own label, or you can accept the default label. Click on **Apply** when done.



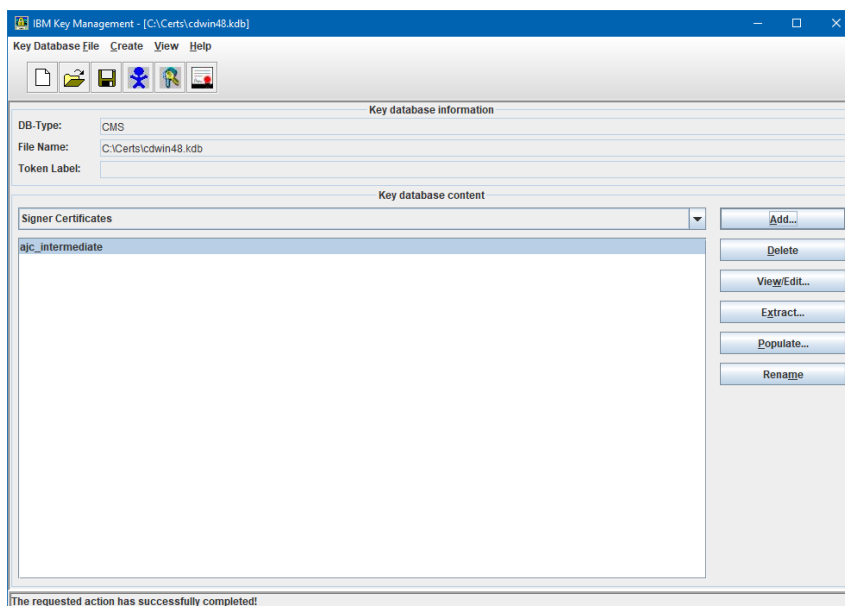
- Enter a label for the certificate being added. Click on **Apply** when done.
- Repeat this for the root certificate and any additional intermediate certificates you have received, then skip to section **Receive the Signed Certificate to Create the KeyCert**.
- On the **Open** popup, verify that the File Name is correct, then click **OK**.



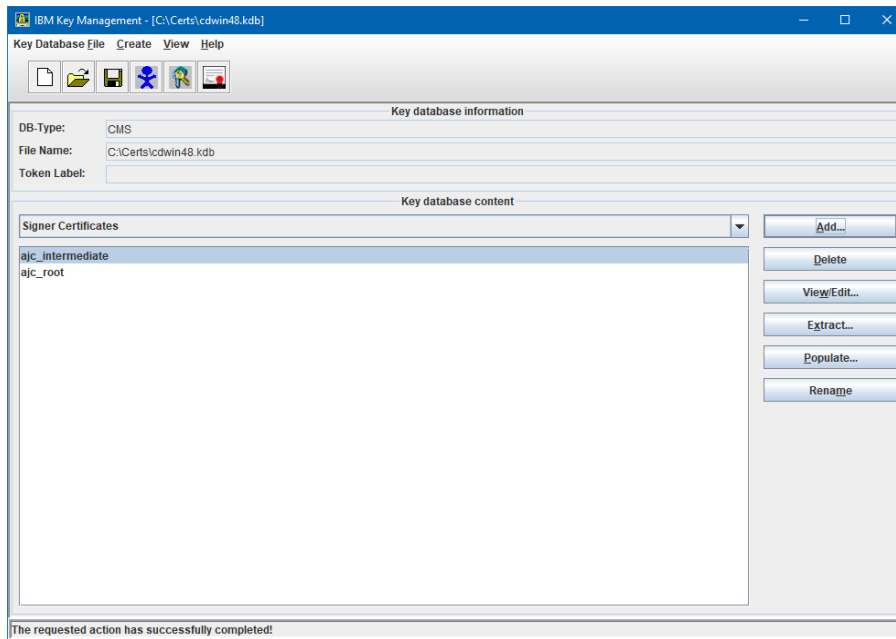
- Enter a label for this certificate. Click **OK** when done.



- The new certificate should now be in the Signer Certificates view.

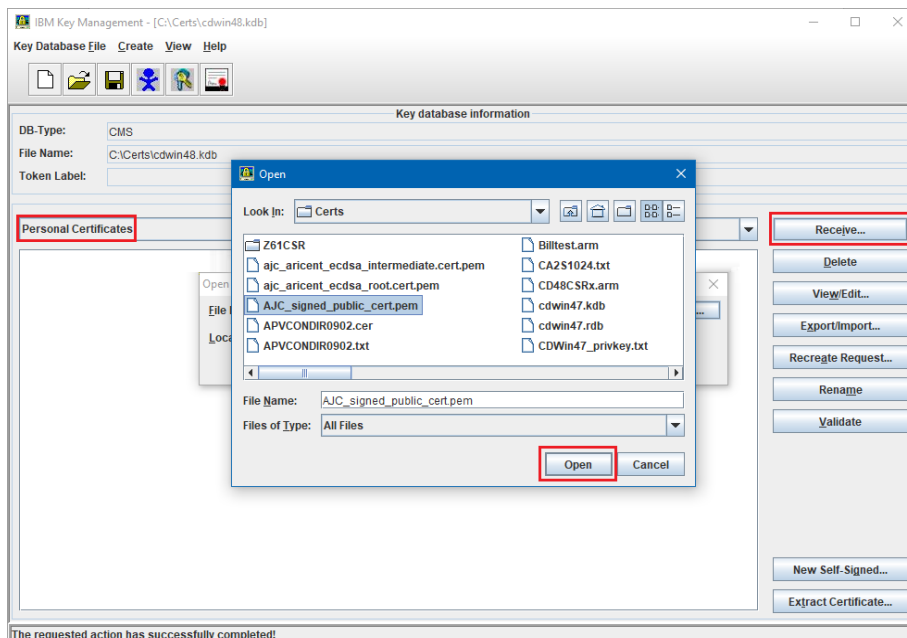


- Repeat this for all intermediate certificates and the root certificate. After you are finished, the root and all intermediate certificates should now be seen in the **Signer Certificates** view.

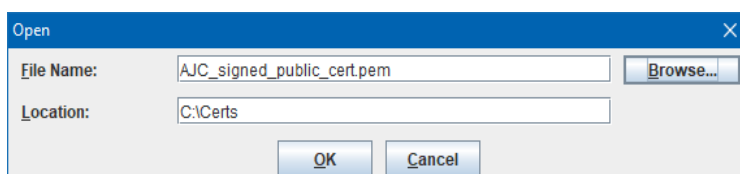


Receive the Signed Certificate to Create the KeyCert

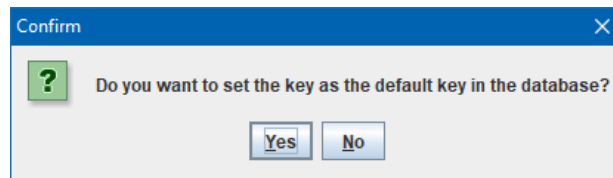
- Select the **Personal Certificates** view. Click on **Receive**, then on the **Open** pop-up select **Browse** and navigate to the folder where the signed (public) server certificate file received from the CA is located. Select the signed certificate file and click **Open**.



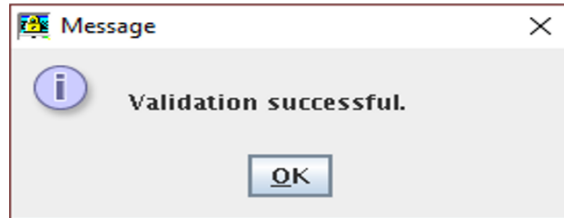
- On the Open popup, verify the correct file has been selected, then click **OK**.



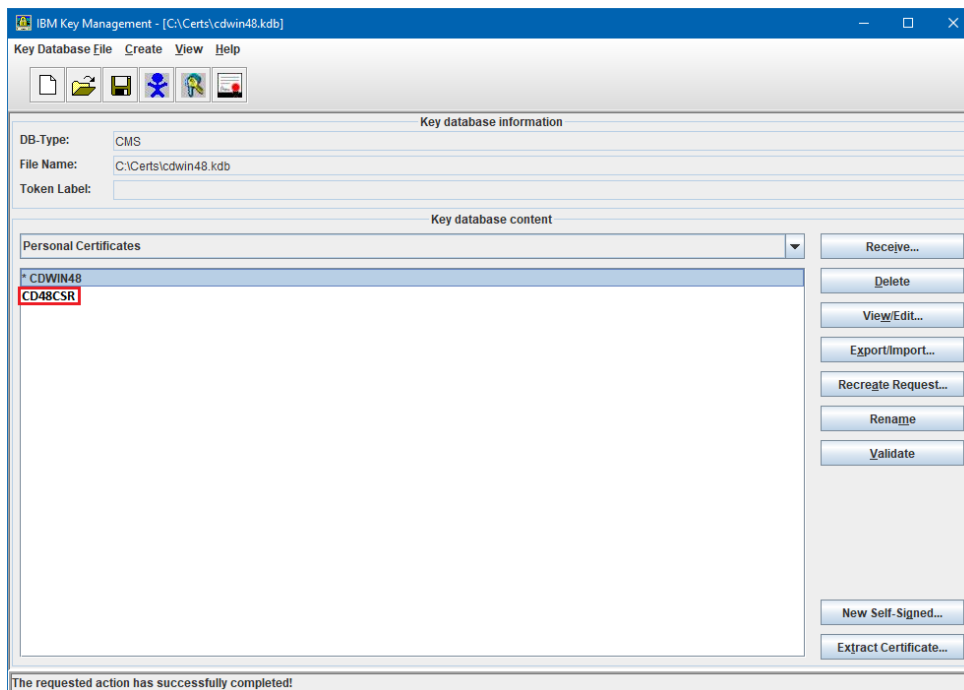
3. Confirm if you want to make this key the default key for your key database (keystore)



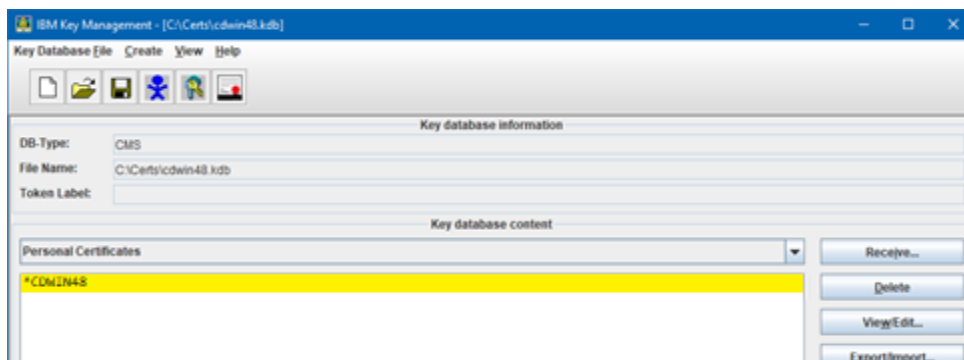
4. If all certificates have been added correctly, you should see the following:



5. You should now see the label used when creating the CSR initially in the list on the Personal Certificates view. The asterisk (*) at the beginning of a label indicates the default key for that keystore.



Note: If the certificate name is in a yellow banner, that means the keystore does not have the complete certificate chain for this certificate, as in the following:

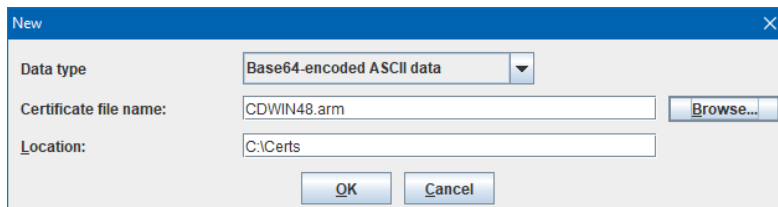


Extracting CA Certificates using IKEYMAN

Before you can send your C:D Windows certificate to the z/OS trading partner, you will need to extract it from IKEYMAN to a file that can then be sent to your partner.

Note: If you are already in the key database, please start with **Step 8**.

1. Start the iKeyman GUI.
Go to Start menu and open All Programs >> IBM Sterling Connect Direct v4.7.0 >> **IBM Key Manager**
2. From the **Key Database File** menu, select **Open**. The Open window displays.
3. Select **Key database type** and select **CMS** (Certificate Management System).
4. Select **Browse** to navigate to the directory that contains the key database files.
5. Select the key database file to which you want to add the certificate, for example **cdwin48.kdb**.
6. Select **Open**. The Password Prompt window displays.
7. Type the password you set when you created the key database and select **OK**.
The name of your key database file displays in the **File Name** field.
8. In the **Key database content** field, select the appropriate certificate type (**Personal or Signer Certificates**) and select the certificate you want to extract.
9. Select **Extract certificate** (button at lower-right). The Extract a Certificate to a File window displays.
10. Select the **Data type** of the certificate as **Base64-encoded ASCII data** for a file with the **.arm** extension.
11. Type the certificate file name and location where you want to store the certificate or select **Browse** to select the name and location.

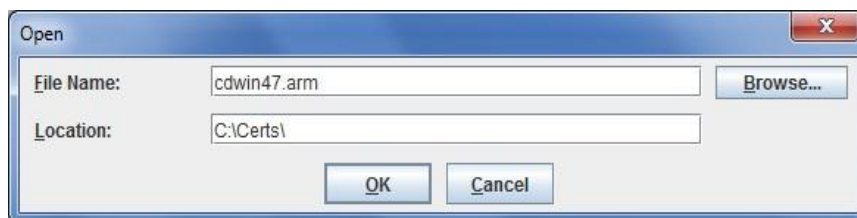


12. Select **OK**. The certificate is written to the file you specified. Send this file to your trading partner.

Add CA Certificates using IKEYMAN

These instructions are for public certificates, not private certificates.

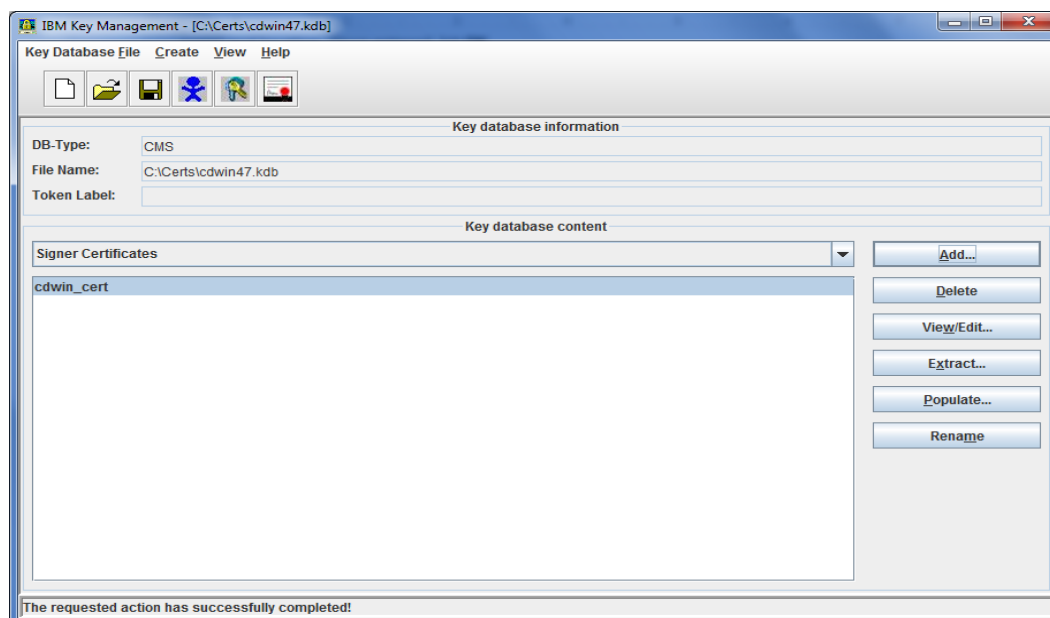
1. Start the IKEYMAN GUI.
Go to Start menu and open All Programs >> IBM Sterling Connect Direct v4.7.0 >> **IBM Key Manager**
2. From the **Key Database File** menu, select **Open**. The Open window displays.
3. Select **Key database type** and select **CMS** (Certificate Management System).
4. Select **Browse** to navigate to the directory that contains the key database files.
5. Select the key database file to which you want to add the certificate, for example **cdwin47.kdb**.
6. Select **Open**. The Password Prompt window displays.
7. Type the password you set when you created the key database and select **OK**.
The name of your key database file displays in the **File Name** field.
8. In the **Key database content** field, select **Signer Certificates**.
9. Select **Add**. Enter the File Name and Location of the certificate being added or select **Browse** to select the name and location. Once entered, select **OK**.



10. Enter a label for this certificate; the label must be unique to this database.



11. The **Signer Certificates** list shows the label of the certificate you added.



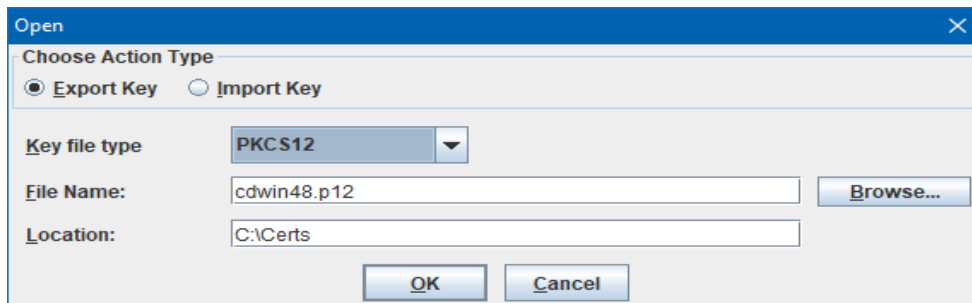
12. The certificate has been added to the key database.

Defining Private Key in IKEYMAN to Use as Site Certificate in gskkyman or RACF

You can use IKEYMAN to build a private key and certificate that can then be exported from IKEYMAN and imported into either the gskkyman key database or RACF (or whatever security application is used) keyring to be used as the default certificate.

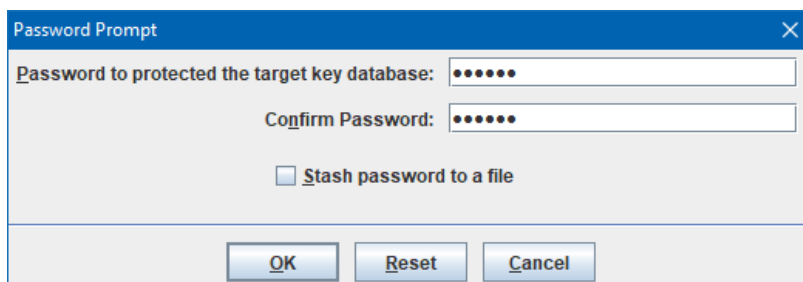
Note: If you are already in the key database, please start with **Step 8**.

1. Start the IKEYMAN GUI.
Go to Start menu and open All Programs >> IBM Sterling Connect Direct v4.8.0 >> **IBM Key Manager**
2. From the **Key Database File** menu, select **Open**. The Open window displays.
3. Select **Key database type** and select **CMS** (Certificate Management System).
4. Select **Browse** to navigate to the directory that contains the key database files.
5. Select the key database file to which you want to add the certificate, for example **cdwin47.kdb**.
6. Select **Open**. The Password Prompt window displays.
7. Type the password you set when you created the key database and select **OK**.
The name of your key database file displays in the **File Name** field.
8. In the **Key database content** field, select **Personal Certificates** and select the certificate label.
NOTE: If you have not already created this private key, please refer to section **Creating Certificates using IKEYMAN** to create the private key.
9. Select **Export/Import** (button at right).
10. A window displays to choose whether you want to Export or Import the key – select **Export Key**.
11. Fill out the following:
 - a. Select the **Key file type** as **PKCS12**.
 - b. Select a file name; the default extension is .p12. In this example, **cdwin48.p12** is used.
 - c. Enter a location where you want to store the key or select **Browse** to select the name and location.



12. Select **OK**. On the **Password Prompt**, enter the Password for this key and confirm the password.

NOTE: REMEMBER THIS PASSWORD! This will be used as the **import file password** when imported into the keyring or key database. **IF FORGOTTEN, THIS PASSWORD CANNOT BE RECOVERED!**



13. Select **OK**. The private key is written to the file (that is, the key file) and location you specified.

This key file needs to be sent to the mainframe as a binary file (if using gskkyman, it can be sent directly to USS via an HFS copy). In this example, this key is being imported into a key database in gskkyman.

Copy from C:D Windows to HFS:

```
HFSCOPYZ PROCESS PNODE=CDWin_node SNODE=CDZ_node
STEP01 COPY FROM (PNODE FILE='C:\Certs\cdwin48.p12'
                  SYSOPTS="datatype(binary) xlate(no)")
TO (SNODE FILE='/u/userid/cdwin48.p12'
   DATATYPE=BINARY PERMISS=777 DISP=NEW)
```

Copy from C:D Unix to HFS:

```
HFSCOPYZ PROCESS PNODE=CDUnix_node SNODE=CDZ_node
STEP01 COPY FROM (PNODE FILE='/path/certs/cdwin48.p12'
                  SYSOPTS=":datatype=binary:xlate=no:")
TO (SNODE FILE='/u/userid/cdwin48.p12'
   DATATYPE=BINARY PERMISS=777 DISP=NEW)
```

Depending on the systems involved, going directly from Windows or Unix to HFS may not work or be allowed. In that case, copy the key file from C:D Windows or C:D Unix to C:D z/OS as a binary file, then copy that binary file to HFS:

Copy from C:D Windows to C:D z/OS:

```
ZOSCOPY1 PROCESS PNODE=CDWin_node SNODE=CDZ_node
STEP01 COPY FROM (PNODE FILE='C:\Certs\cdwin48.p12')
                  SYSOPTS="datatype(binary) xlate(no)"
TO (SNODE FILE=hlq.CDWIN48.P12
   DISP=NEW)
```

Copy from C:D Unix to C:D z/OS:

```
ZOSCOPY1 PROCESS PNODE=CDUnix_node SNODE=CDZ_node
STEP01 COPY FROM (PNODE FILE='/path/certs/cdwin48.p12')
                  SYSOPTS=":datatype=binary:xlate=no:"
TO (SNODE FILE=hlq.CDWIN48.P12
   DISP=NEW)
```

Copy from C:D z/OS to HFS:

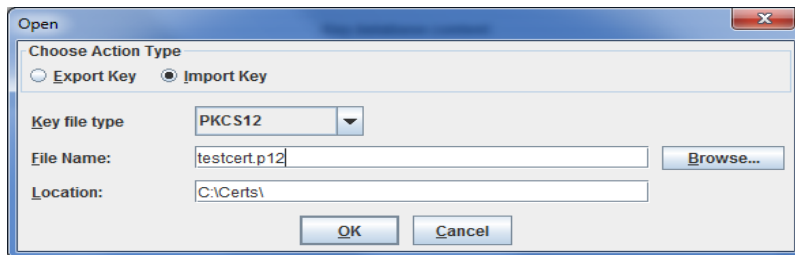
```
HFSCOPY2 PROCESS PNODE=CDZ_node SNODE=CDZ_node
STEP01 COPY FROM (PNODE FILE=hlq.CDWIN48.P12 DISP=SHR) -
TO (SNODE FILE='/u/userid/cdwin48.p12' -
   DATATYPE=TEXT PERMISS=777 DISP=NEW)
```

For information on how to load the key file into gskkyman, see **Loading Private Key (For Use as Site Certificate) into gskkyman** on page 39.

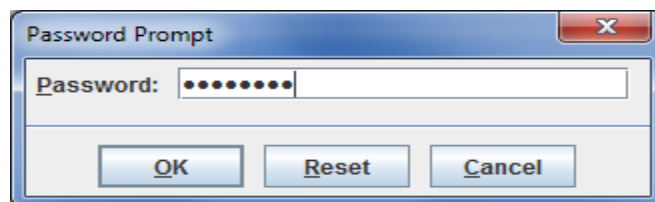
Importing Private Key Into IKEYMAN

Note: If you are already in IKEYMAN and have opened the key database, please start with **Step 8**.
If your certificate is in .PEM format (format used prior to C:D Windows 4.7 and C:D Unix 4.2), go to section **Import Certificate into CMS KeyStore** on page 140.

1. Start the IKEYMAN GUI.
(For example, go to Start menu and open All Programs >> IBM Sterling Connect Direct v4.7.0 >> **IBM Key Manager**)
2. From the **Key Database File** menu, select **Open**. The Open window displays.
3. Select **Key database type** and select **CMS** (Certificate Management System).
4. Select **Browse** to navigate to the directory that contains the key database files.
5. Select the key database file to which you want to add the certificate, for example **cdwin47.kdb**.
6. Select **Open**. The Password Prompt window displays.
7. Type the password you set when you created the key database and select **OK**.
The name of your key database file displays in the **File Name** field.
8. In the **Key database content** field, select **Personal Certificates** and select the certificate label.
9. Select **Export/Import** (button at right). **NOTE:** Some versions may just have **Import** button.
10. A window displays to choose whether you want to Export or Import the key; select **Import Key**.
11. Fill out the following:
 - d. Select the **Key file type** as **PKCS12**.
 - e. Select a file name – the default extension is .p12. In this example, **testcert.p12** is used.
 - f. Select a location where you saved the key or select **Browse** to select the name and location.



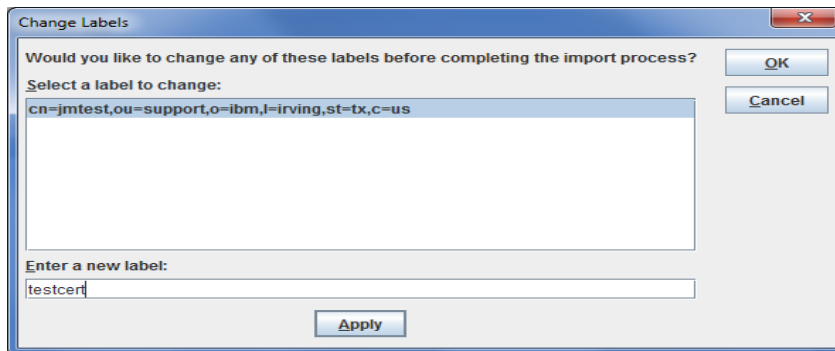
12. Select **OK**. On the **Password Prompt**, enter the **Password** for this key.



13. Select **OK**.
14. You will now see a popup to Change Labels:

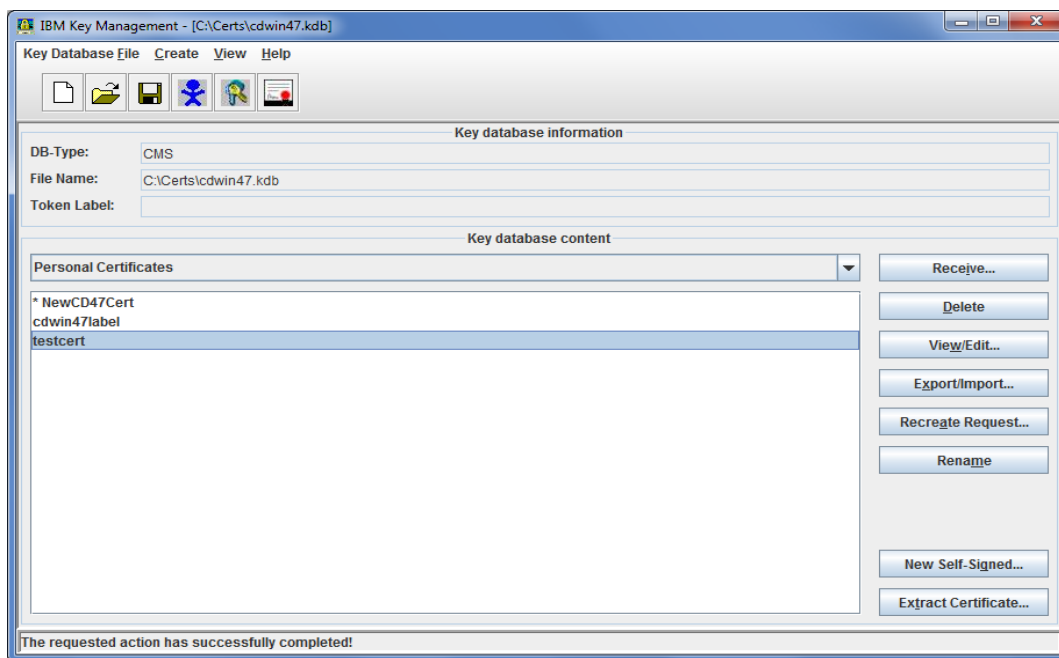


15. Since the default label is information from the certificate, it is recommended this be changed to a more meaning Label by selecting the label in the main box, then adding a new label in the **Enter a new label** box at the bottom.

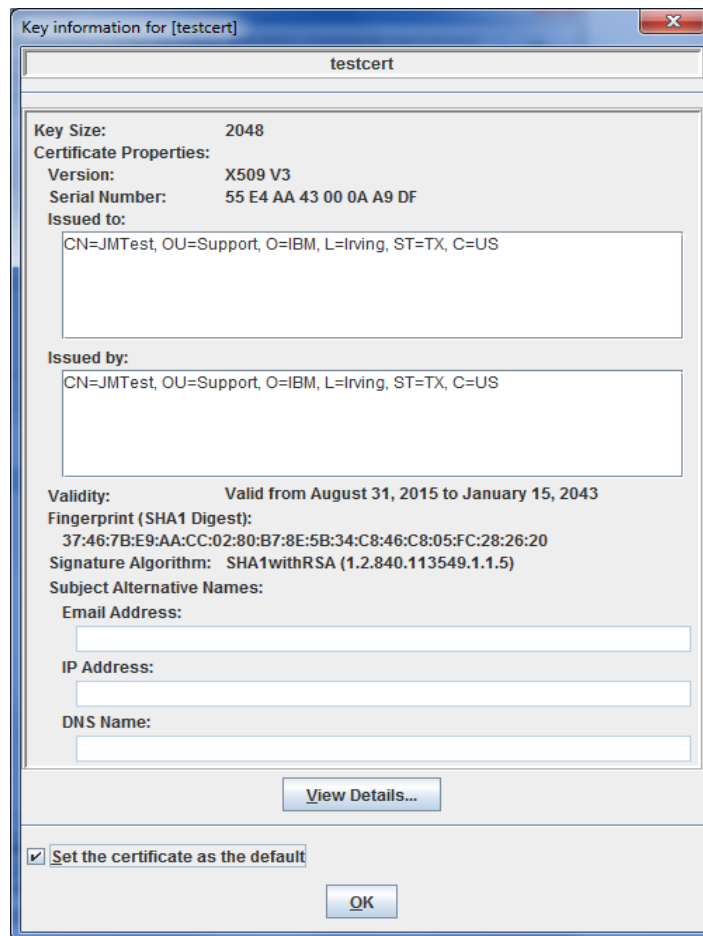


16. Select **Apply**. The label in the main box will change to the new label.
17. Select **OK**.

The new certificate that was just added should now appear in the **Personal Certificates** window.

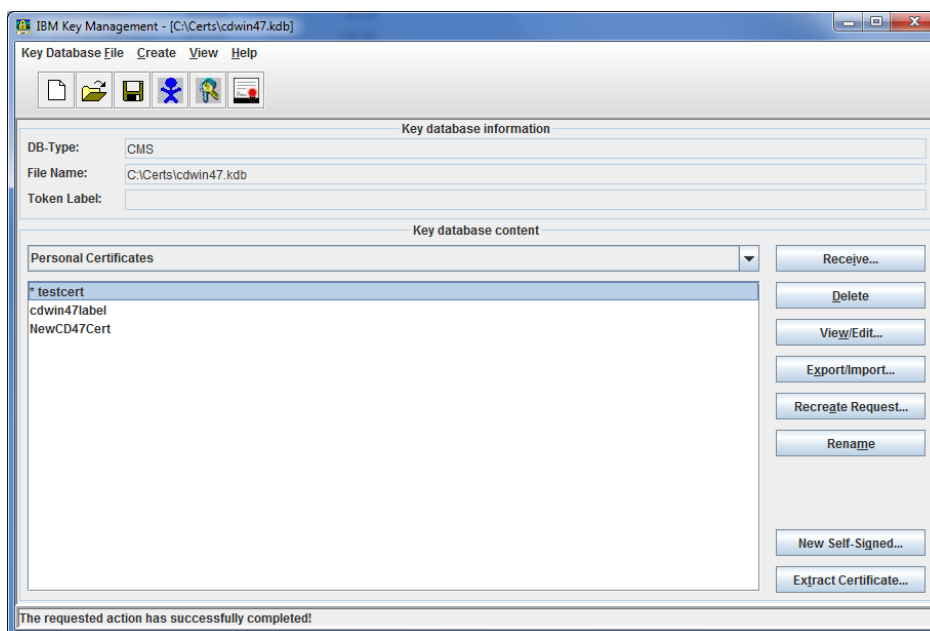


If this new certificate is to be used as the default site certificate, double-click on it; a pop-up will appear like the following:



Check the box at the bottom **Set the certificate as the default**, then select **OK**.

This new certificate will now appear with an asterisk (*) beside it, indicating that it is the default certificate.



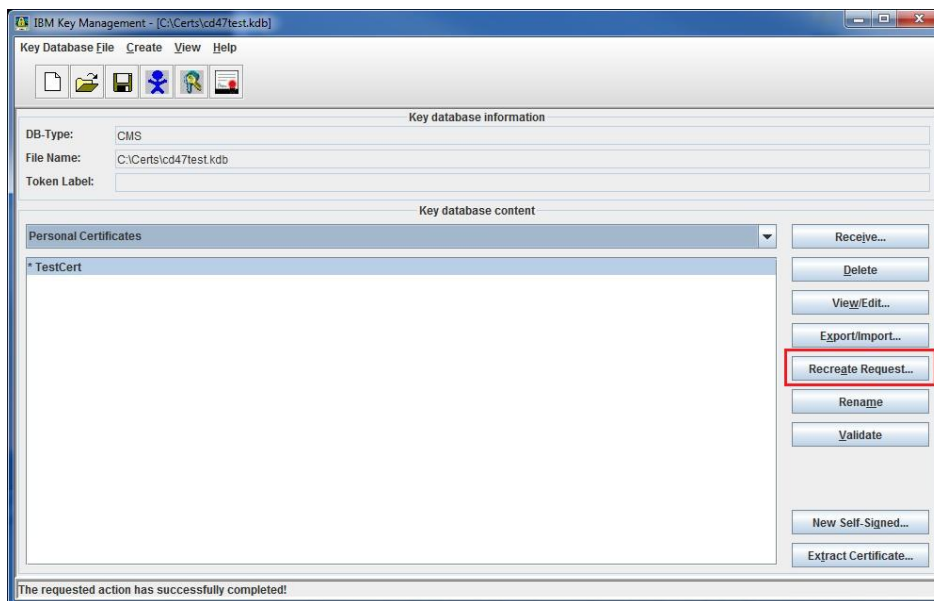
The certificate is ready to use.

Renew an Existing Key Certificate

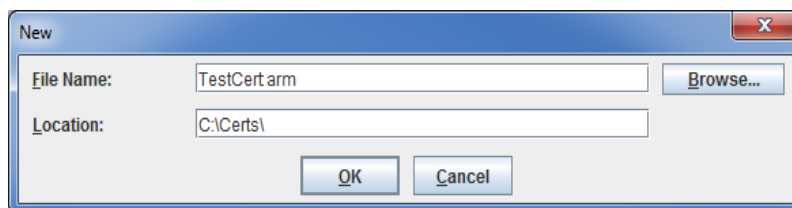
NOTE: This will only work if the Key Certificate is not expired; if already expired, you will need to start over with a new Certificate Signing Request. Also, a self-signed certificate cannot be renewed.

Start IKEYMAN and open your keystore (see steps 1-7 under **Importing Private Key Into IKEYMAN** on page 62 for help in doing this).

On the right side of the panel, select **Recreate Request**.



In the pop-up, name your file (keep the .arm extension) and the location where the file will be saved:



Select **OK**.

The .arm file created will contain something like the following:

```
-----BEGIN NEW CERTIFICATE REQUEST-----
MIICvkCCAAyCAQAweTElMAkGA1UEBhMCVVMxDjAMBgNVBBETBTclMDM4MQswCQYDVQQIEwJlUWDER
MA8GA1UEBxMIQW55d2hlcmUxEjAQBgNVBAoTCU1CTSBDb3JwLjEQA4GA1UECXMHU3VwcG9ydDEU
MBIGA1UEAwwLQ29tbW9uX05hbWUwgGElMA0GCSqGSIb3DQEBAQUAA4IBDwAwggEKAoIBAQCd6P64
klLHJJJeWOAfAlhnd/ws4UTpFhyqw6GGpLeHtHCGOWfZycIDpu8IUEjo12fs6bErhuxnAXs2Ox+n1
L6Xe9zsFc11OF33Nzxfj3jjkqZouS/97A/GMHZb3Kx52ODEAMkyDtL3culpwR15oP4wcMpamPwoB
nxAREhetfrIMIDdPS0k+6hqSmX/DGkeN1R6AtRaCUDt8t2vdc1qzA4uIAVRlWdmxNsIkUXghFw5a
oQ1Bocjps7qi5r/ZKAo7O2w01hQeEl3uQLwS3OMTvd6nKk0K9yPnEx5+QKCs+SzrpsMSdlwWqL6
DYks9OVQGqiTnrgJSAAMFw22WD18+h/zAgMBAAGgADANBgkqhkiG9w0BAQsFAAOCAQEAJP3XE2Tf
6InizjdBQKxOJuMb2TB/SkMmGVVtdChJIgnkBRM6wa4x42t8AJsn/GlkIPsNpdsGxboZ2v/zRs7I
nc2WKJMCrCnZY6B6AP/FAKKnfdDuzrFOjnfCDO0NqiAcjAlno7I0FDVQN+/1dBQK15AIViOUeSA5
kk2EssivqWARf1VhrHr3CilA32m7un64czFN0mBifR4YiO5J4Xotttrvz/eBE3oBDX48FHrmEQOEi
T5TzFWtbTpWha7bG6YVWiiWdHLvMmxM2avQgLy8nzhLwAtpDgspuYRW9Y8Z9vtfmcUMWvwMWfmGK
KCkn5244ybrot2L80ePu0TI4UsZvbA==
-----END NEW CERTIFICATE REQUEST-----
```

Send this “Renew Certificate Request” file to your Certificate Authority (CA). From the CA, request an Apache style, x509, base64 encoded, PEM format signed server (public) certificate. If needed, also get the updated Root and Intermediate certificates from the CA.

Once you receive the certificate(s) back from your Certificate Authority (CA), import the certificate(s) into IKEYMAN.

Using IKEYMAN From the Command Line Interface (CLI)

The **ikeycmd** information here is condensed from the [IBM Key Manager User's Guide](#), available here: **IKEYMAN Users Guide**.

The **ikeycmd** executable is located in the following folder, depending on the C:D Unix version that is installed and the C:D installed path:

```
/<cd_install_path>/jre/ibm-java-[i386|x86_64]-[70|80]/jre/bin/
```

NOTE: You **MUST** know the password for the Keystore database to use these commands (this password was set when the Keystore was created, typically at CDU Secure+ install). If forgotten, the Keystore password is **not** recoverable and a new Keystore database will need to be created.

This section covers the following topics using the Command Line Interface:

1. Create a Key Database
2. Create a Certificate Signing Request (CSR)
3. Create a Self-Signed Certificate
4. Add Certificate to Key Database
5. Extract Certificate from Key Database
6. Viewing Certificates in the Keystore
7. Renew Existing Key Certificate That Is About to Expire
8. Updating Secure+ to Use the New Key Certificate

Create a Key Database

Use the following command to create a key database using the Command Line Interface (CLI):

```
ikeycmd -keydb -create -db /<cd_install_path>/ndm/secure+/certificates/cdkeystore.kdb -pw  
keystore_password -type cms
```

The key database (or keystore) name can be anything, but it should have an extension of **.kdb**.

NOTE: Remember the password you select; if forgotten or lost, the password is not recoverable, and a new key database will have to be created. **It must be TYPE CMS.**

Create a Certificate Signing Request (CSR)

NOTE: The **-label** and **-file** name values used in the following command are examples; it is strongly recommended that the values chosen be descriptive to explicitly identify the object being created. The value for **-size** can be up to 4096, though some systems may not support this size.

```
ikeycmd -certreq -create -db /<cd_install_path>/ndm/secure+/certificates/cdkeystore.kdb -label  
new_keycert_yyyymmdd -file  
/<cd_install_path>/ndm/secure+/certificates/new_certreq_yyyymmdd.pem -pw keystore_password  
-dn "CN=cdu_server_name.your_domain.com, OU=your_department, O=your_company, L=your_city,  
ST=your_state(2-char), POSTALCODE=your_zipcode, C=your_country(2-char)" -size 2048 -sig_alg  
SHA256WithRSA
```

NOTE: If running **CDU 4.2.0.4 iFix005** or earlier, use value **-sig_alg SHA1WithRSA** instead of **SHA256WithRSA**. If you are not sure what version you are running, issue the following to determine the exact CDU version:

```
/<cd_install_path>/etc/cdver
```

Optional fields that you may want to add are:

- expire** (optional: default is 365 – can be 0 to 7300)
- default_cert** (optional – if omitted, will not be the default certificate)

If you wish to use an ECDSA certificate, then you can use the following **-size** and **-sig_alg** combination:

- size 256 -sig_alg SHA256WithECDSA**
- size 384 -sig_alg SHA384WithECDSA**
- size 512 -sig_alg SHA512WithECDSA**

Ensure that you tell your CA that you are using ECDSA. The signing process is different for ECDSA.

Optional **Subject Alternative Names** can also be added and used with the CSR.

Add one or more to the end of the command string:

- san_dnsname** *<dns_name_of_server>*
- san_emailaddr** *<email_of_server_admin>*
- san_ipaddr** *<ip_address_of_server>*

Send the ***new_certreq_yyyymmdd.pem*** file (or whatever you chose to call it) to your Certificate Authority (CA). From the CA, request an Apache style, x509, base64 encoded, PEM format signed server (public) certificate. Also, get the Root and Intermediate certificates from the CA.

Create a Self-Signed Certificate

NOTE: The **-label** and **-file** name values used in the following command are examples; it is strongly recommended that the values chosen be descriptive to explicitly identify the object being created. The value for **-size** can be up to 4096, though some systems may not support this size.

By default, this command creates a self-signed X509 certificate in the identified key database. A self-signed certificate has the same issuer name as its subject name.

```
ikeycmd -cert -create -db /<cd_install_path>/ndm/secure+/certificates/cdkeystore.kdb -label  
new_keycert_yyyymmdd -file  
/<cd_install_path>/ndm/secure+/certificates/new_certreq_yyyymmdd.pem -pw keystore_password  
-dn "CN=cdu_server_name.your_domain.com, OU=your_department, O=your_company, L=your_city,  
ST=your_state(2-char), POSTALCODE=your_zipcode, C=your_country(2-char)" -size 2048 -sig_alg  
SHA256WithRSA
```

NOTE: If running **CDU 4.2.0.4 iFix005** or earlier, use value **-sig_alg SHA1WithRSA** instead of **SHA256WithRSA**. If you are not sure what version you are running, issue the following to determine the exact CDU version:

```
/<cd_install_path>/etc/cdver
```

If using an ECDSA certificate, the following **-size** and **-sig_alg** combination can be used:

- size 256 -sig_alg SHA256WithECDSA**
- size 384 -sig_alg SHA384WithECDSA**
- size 512 -sig_alg SHA512WithECDSA**

Optional fields that may be added are:

- expire** (optional: default is 365 – can be 0 to 7300)
- default_cert** (optional – if omitted, will not be the default certificate)

Optional **Subject Alternative Names** can also be added and used with the CSR. Add one or more to the end of the command string:

- san_dnsname <dns_name_of_server>**
- san_emailaddr <email_of_server_admin>**
- san_ipaddr <ip_address_of_server>**

Add Certificate to Key Database

Any root and intermediate certificates received from the remote trading partners for their Connect:Direct servers will need to be added to the key database.

For these instructions, please note the following:

/<cd_install_path>/ndm/secure+/certificates/ -

the path where your key database is located and/or your CA certificate is located. If different than this, add the correct path to the command. If the CA certificate is not located in the same path as the key database, change the command accordingly.

cdkeystore.kdb -

this is whatever key database (keystore) you are using. If your key database has a different name, add the correct name to the command.

keystore_password -

the password of the key database you are using.

- A. If you have received a certificate from a CA (that is, as a result of a CSR you created), receive the Signed Server (Public) Certificate into the key database to create the Key Certificate.

```
ikeycmd -cert -receive -db /<cd_install_path>/ndm/secure+/certificates/cdkeystore.kdb -file  
/<cd_install_path>/ndm/secure+/certificates/signed_cert_from_CA.pem -pw keystore_password  
-format ascii -default_cert yes
```

The **label** used for the Key Certificate will be the same as the label used for creating the CSR.

signed_cert_from_CA.pem -

the name of the CA certificate file received from the CA in PEM format.

NOTE: The keystore **cdkeystore.kdb** in folder **/<cd_install_path>/ndm/secure+/certificates/** should have been created during the installation of C:D Unix. Only change the ikeycmd commands if needed.

- B. If the certificate(s) received from the trading partner are in separate files, you can specify your own labels:

```
ikeycmd -cert -add -db /<cd_install_path>/ndm/secure+/certificates/cdkeystore.kdb -file  
trusted_cert_from_TP.txt -label your_choice_of_label -pw keystore_password -trust enable
```

This needs to be issued for each certificate being added to the key database.

trusted_cert_from_TP.txt -

the name of the certificate file received from your trading partner (probably a text file).

- C. If the certificates received from the trading partner are bundled in a single file, the **CN** value of the certificates will be used for the labels.

```
ikeycmd -cert -add -db /<cd_install_path>/ndm/secure+/certificates/cdkeystore.kdb -file  
trusted_cert_file_from_TP.txt -pw keystore_password -trust enable
```

trusted_cert_file_from_TP.txt -

the name of the certificate file received from your trading partner (probably a text file).

NOTE: The **-label** and **-file** values used in the above commands are examples; it is strongly recommended that the values chosen be descriptive to explicitly identify the object being created.

Extract Certificate from Key Database

Your root and intermediate certificates will need to be sent to your trading partners. Do not send the signed server (public) certificate.

To extract a certificate from the key database to send to the trading partner, issue the following for each certificate that is being extracted:

```
ikeycmd -cert -extract -db /<cd_install_path>/ndm/secure+/certificates/cdkeystore.kdb -label  
your_choice_of_label -target /<cd_install_path>/ndm/secure+/certificates/<current_key_certificate  
_label>.pem -pw keystore_password -format ascii
```

NOTE: The **-target** value used in the above command indicates the name of the file created and the folder into which the certificate will be extracted; this is an example. Make sure to remember the name of the file and the folder where the file is being placed.

Viewing Certificates in the Keystore

To view all certificates in the Keystore:

```
ikeycmd -cert -list -db /<cd_install_path>/ndm/secure+/certificates/cdkeystore.kdb -pw  
<keystore_password>
```

To view details of the default key certificate:

```
ikeycmd -cert -getdefault -db /<cd_install_path>/ndm/secure+/certificates/cdkeystore.kdb -pw  
<keystore_password>
```

To view details of any certificate in the Keystore.

```
ikeycmd -cert -details -db /<cd_install_path>/ndm/secure+/certificates/cdkeystore.kdb -label  
certificate_label -pw <keystore_password>
```

Renew Existing Key Certificate

NOTE: This will only work if the Key Certificate is not expired; if already expired, you will need to start over with a new Certificate Signing Request. Also, a self-signed certificate cannot be renewed.

1. View the details of the current default key certificate and note the label of the key certificate.

```
ikeycmd -cert -getdefault -db /<cd_install_path>/ndm/secure+/certificates/cdkeystore.kdb -pw  
<keystore_password>
```

2. Recreate the Certificate Signing Request from the existing key certificate.

```
ikeycmd -certreq -recreate -db /<cd_install_path>/ndm/secure+/certificates/cdkeystore.kdb -label  
<current_key_certificate_label> -target /<cd_install_path>/ndm/secure+/certificates/  
<current_key_certificate_label.pem> -pw <keystore_password>
```

3. Send the 'renew certificate request' file to your Certificate Authority (CA). From the CA, request an Apache style, x509, base64 encoded, PEM format signed server (public) certificate. If needed, also get the updated Root and Intermediate certificates from the CA.

4. Receive the renewed Signed Server (Public) Certificate into the Keystore to update the Key Certificate:

```
ikeycmd -cert -receive -db /<cd_install_path>/ndm/secure+/certificates/cdkeystore.kdb -file  
/<cd_install_path>/ndm/secure+/certificates/<renewed_signed_cert_from_CA.pem> -pw  
<keystore_password> -format ascii -default_cert yes
```

5. Verify the new validity date of the key certificate.

```
ikeycmd -cert -getdefault -db /<cd_install_path>/ndm/secure+/certificates/cdkeystore.kdb -pw  
<keystore_password>
```

6. Look for the "Valid: From:" line in the output and verify the validity date.

Updating Secure+ to Use the New Key Certificate

Launch the Secure+ Command Line interface.

```
/<cd_install_path>/ndm/bin/spcli.sh
```

Execute this command to update the 'LocalNode' record.

```
update localnode KeyCertLabel=<key certificate label>;
```

Execute this command to update a 'RemoteNode' record.

```
update remotenode Name=nodename KeyCertLabel=<key certificate label>;
```

Migrating to New Site Certificate (or Using an Alternate Site Certificate)

When migrating to a new certificate, the old certificate can be left in the keyring / key database; however, if the old certificate is left, the new certificate would have to have a different label name than the old one. There are several reasons for leaving the old certificate:

1. The old certificate is being used until full testing is complete to verify the new certificate (this also applies to old certificates about to expire but are still being used until the new certificate is installed and used).
2. The new certificate may be an upgrade (for example, going from a SHA-1 to a SHA-2 certificate) and some trading partners may not be able to switch over immediately.
3. One or more of your trading partners require keeping the old certificate.

How much work you choose to do will determine if you choose to have both certificates in the keyring / key database.

Removing the Old Certificate and Adding the New Certificate

From a Secure+ perspective, the easiest way to migrate the certificates is to remove the old certificate from your keyring / key database and add the new certificate with the same label name. This would require no changes to the Secure+ PARMFILE and a simple refresh of the Secure+ environment (in the C:D z/OS IUI, panel ADMIN;S, option RF).

However, you would need to ensure that all your remote Secure+ trading partners have your new certificate loaded into their perspective systems and that they are aware of the precise day and time that you are going to start using the new site certificate. If they do know when you do the switch and have not loaded the new certificate correctly in their systems, you will have problems with the certificates not matching.

Keeping Both the Old and New Certificates

If you decide to keep the old certificate in your keyring / key database along with the new certificate, there are two different ways to do this:

- A. Load the new certificate with a new certificate label name in your keyring / key database; however, do not change the **Certificate Label** in the Local node definition in the Secure+ PARMFILE. For each remote trading partner that has successfully loaded and verified the new certificate, enter the new certificate label name in the **Certificate Label** in place of the asterisk (*) in the remote node definition for that trading partner. Once you have all your remote trading partners using the new certificate, update your Secure+ PARMFILE:
 - 1) Change the Secure+ PARMFILE local node to point to the new certificate label in the **Certificate Label**.

- 2) Change the **Certificate Label** back to an asterisk (*) in all your Secure+ PARMFILE remote node definitions.
 - 3) Do a **Save Active** to save the changes to your Secure+ PARMFILE (or a **Save As** and create a new PARMFILE if this is not the active PARMFILE being used).
- B. Load the new certificate with a new certificate label name in your keyring / key database. Update all your Secure+ PARMFILE remote node definitions with the old certificate label name in the **Certificate Label** field in place of the asterisk (*). In the Secure+ PARMFILE local node definition change the **Certificate Label** to point to the new certificate instead of the old one. Then as your remote Secure+ trading partners confirm that they have your new certificate loaded and available change the **Certificate Label** in the remote node definition from the old certificate label name to an asterisk (*). Make sure that you save the Secure+ PARMFILE each time you change a remote node definition.

Whichever method is chosen, when all remote Secure+ trading partners have migrated to the new certificate, you can remove the old certificate from your keyring / key database.

Should you need to have both certificates be used, you can select which will be the “default” for your system and be loaded in the **Certificate Label** in the local node definition of your Secure+ PARMFILE. Then you can add the old certificate label name in the **Certificate Label** field of each remote node that needs to keep using the old certificate.

NOTE: It is up to you and your remote trading partners to negotiate which certificate will be used. This may require a decision that a set cutoff date be implemented, such as if the old certificate is expiring and can no longer be used after the expiration date. After the cutoff date, the Secure+ transfers will no longer work without the new certificate loaded correctly.

Regardless of which option you decide to take you must ensure that the remote trading partner has your new certificate and has it loaded correctly.

Managing the Secure+ Parameter File (PARMFILE) on C:D z/OS

The Connect:Direct Secure+ Parameter File (PARMFILE) is required to run Secure+ and contains information that determines the protocol and encryption method used during security enabled Connect:Direct operations. The Secure+ PARMFILE must contain one local node record and a remote node record for each trading partner using Connect:Direct Secure+ to perform a secure connection. The local node record is the default for the PARMFILE and should define the default security and protocol settings for that PARMFILE. Each remote node record defines the specific security and protocol used by a trading partner and can override the settings in the local node record.

NOTE: When saving a Secure+ PARMFILE, the **Save Active** option can only be used on a Secure+ PARMFILE that is currently active (i.e., defined in the C:D initialization parameters on the **SECURE.DSN=** parameter). If a Save Active is attempted on any other PARMFILE, the changes will either be applied to the active PARMFILE or result in a hashing error. If there is no active PARMFILE, an error will occur.

C:D z/OS 6.0 added new functional authority C:D z/OS to enable users to be restricted from accessing the Secure+ Admin Tool. This is done through either the AUTH file or the Stage-2 Security Exit. C:D z/OS 6.0 added function **SECURE+ COMMANDS** to the AUTH file and added functionality to **SECURE+ ADMIN** and to functions **S#WNCR** (Write to Secure+ PARMFILE) and **S#RNCR** (Read Secure+ PARMFILE) in BYTE08 of the Functional Authorities defined in the Stage-2 Security Exit. If **SECURE+ ADMIN** is set to **N**, or if **S#WNCR** is not coded in the Stage-2 Security Exit, then the userid will not be able to access the Secure+ Admin Tool, whether by entering the **S** (Execute Secure Plus Commands) panel or by entering **SA** from the ADMIN panel.

In the AUTH file, if **SECURE+ ADMIN** is **N** but **SECURE+ COMMANDS** is **Y**, then the remainder of the **S** panel will still be available; however, if both are set to **N**, then the **S** panel will not be available. If **SECURE+ ADMIN** is set to **Y**, **SECURE+ COMMANDS** will be forced to **Y**.

In the Stage-2 Security Exit, if **S#RNCR** is coded, then Secure+ options **CR** (Certificate Expiration Validation Command) and **RF** (Refresh Secure Plus Environment Command) are available; if **S#WNCR** is coded, then access to the Secure+ PARMFILE is allowed.

If a userid does not have access to the Secure+ Admin Tool, it will not be able to create a Secure+ PARMFILE.

Manually Create Connect:Direct Secure+ PARMFILE

NOTE: The Secure+ PARMFILE changed significantly beginning in C:D z/OS 5.2, particularly with the removal of support for the STS protocol. Therefore, if migrating from a pre-C:D z/OS 5.2, it is important to make sure that the IUI libraries and panels are migrated to the new release.

Beginning with C:D z/OS 5.2, the Secure+ PARMFILE is encrypted using ICSF (Integrated Cryptographic Service Facility); therefore, **ICSF must be active before Secure+ will initialize**. Prior to C:D z/OS 5.2, Certicom was used to encrypt the Secure+ PARMFILE (which was also used for the STS protocol). Starting in C:D z/OS 5.2, **if ICSF is not active, or the C:D userid does not the proper permissions to ICSF, C:D Secure+ will not initialize**.

Go into the IUI and go to panel **ADMIN;S** and select option **SA**.

SC.ZOS.JMCGE52	Execute Secure Plus Commands	14:55
CMD ==> SA		
CR - Execute Certificate Expiration Validation Command		
RF - Execute Refresh Secure Plus Environment Command		
SA - Execute Secure Plus Admin Tool		

NOTE: The Secure+ Admin Tool can also be brought up by going to **ADMIN;SA** and hitting **Enter**.

This will bring up the Secure+ Admin Tool.

```
File  Edit  Help
-----
Secure+ Admin Tool: Main Screen
Option ==> _____ Scroll CSR

Table Line Commands are:

U Update node      H View History      D Delete node
I Insert node      V View node        CL Clone node

Node Filter : * _____

Secure+                               External Client
LC Node Name      Type Protocol Override Encryption Auth Auth
-----
***** BOTTOM OF DATA *****
```

There are two ways to manually create a Secure+ PARMFILE:

- 1) Create the PARMFILE from the NETMAP (Quick Start)
- 2) Creating Each Node Manually (Without Using the NETMAP)

NOTE: If the PARMFILE is not very large (that is, there are not many nodes in the PARMFILE), it is recommended to use option **2)** and manually create the local and remote nodes, as it presents fewer problems. If the PARMFILE contains many nodes, using the NETMAP method will be much quicker.

Creating the PARMFILE from the NETMAP (Quick Start)

Either type **FO** on the command line and hit **Enter**, or go up to the pull-down menu, select **File**, then select option **2**:

```
File  Edit  Help
-----
2 1. New      (FN)
  2. Open    (FO)
  *. Close   (FC)
  4. Info    (FI)
  *. Report  (FR)
  *. Save Active (FS)
  *. Save as  (FA)
  *. Unload  (FX)
  9. Exit    (FE)

e+ Admin Tool: Main Screen
Scroll CSR

ble Line Commands are:

View History      D Delete node
View node        CL Clone node

Secure+                               External Client
LC Node Name      Type Protocol Override Encryption Auth Auth
-----
***** BOTTOM OF DATA *****
```

Type the name of your current Connect:Direct NETMAP, then hit **Enter**.

```
Secure+ Admin Tool: File Selection

Option ---> _____

Enter file name for: INPUT SECURE PARM FILE

File
Name: CD.HOST62.NETMAP Browse
---

File System Type:
1 1. MVS Cancel
---
```

Select **Yes** and hit **Enter**.

You will see something like the following: hitting **Enter** again will remove the **007** message panel.

You will also note that all SNA nodes in your NETMAP (except for your Local node) have been removed.

```

File  Edit  Help
-----
SC.ZOS.JMCGE52      Secure+ Admin Tool: Main Screen      Row 1 to 9 of 25
Option ==>          Scroll CSR

                        Table Line Commands are:

U Update node        H View History        D Delete node
I Insert node        V View node          CL Clone node

Node Filter : *

LC Node Name        Type  Secure+ Protocol Override Encryption  External Client Auth  Auth
-----
CDWIN48             R   *           N           *           *       N
CDWIN60             R   *           N           *           *       N
CDWIN61             R   *           N           *           *       N
SC.ALT.CD62T        R   *           N           *           *       N
SC.ZOS.CD61T        R   *           N           *           *       N
SC.ZOS.JMCD60T      R   *           N           *           *       N
SC.ZOS.JMCD61T      R   *           N           *           *       N
SC.ZOS.JMCGE62      L   Disabled   Y           Y           N       N
SC.ZOS.JMCGE62T     R   *           N           *           *       N

```

Placing a **U** on the line next to the local node (on the **LC** line) then hitting **Enter** will bring up the following:

```

                        Secure+ Create/Update Panel
Option ==>

Node Name:  SC.ZOS.JMCGE62      Type:  L      (Local or Remote)
-----
| Security Options | EA Parameters | SSL/TLS Parameters |
| ---            | --           | ---                |
-----
Secure+ Protocol:      Security Mode (Yes , No , Default to Local)
Enable SSL             N      Enable FIPS             N
Enable TLS 1.0         N      Enable SP800-131a Transition N
Enable TLS 1.1         N      Enable SP800-131a Strict   N
Enable TLS 1.2         N      Enable NSA Suite B 128 bit N
Enable TLS 1.3         N      Enable NSA Suite B 192 bit N

Auth Timeout:         120      Enable Override         Y

Alias Names:          TCP Information:
_____              IPaddr: _____
_____              Port:      _____
_____

                        OK      Cancel
                        --      ---

```

You can then edit the local node. Once the changes are made, place cursor on **OK** and hit **Enter**.

NOTE: Prior to C:D z/OS 6.1, the **Enable TLS 1.3** protocol will not be present.

You can then either update (**U**) each remote node, as needed, or delete (**D**) each node that is not needed.

You can now skip to section **Saving the New Secure+ PARMFILE**.

Creating Each Node Manually (Without Using the NETMAP)

The other option is to enter each node manually without the NETMAP.

When entering the Secure+ Admin Tool, you should see the following:

```
File  Edit  Help
-----
Secure+ Admin Tool: Main Screen
Option ==> _____ Scroll CSR

Table Line Commands are:

U Update node      H View History      D Delete node
I Insert node      V View node        CL Clone node

Node Filter : *

LC Node Name      Secure+      External Client
Type Protocol Override Encryption Auth Auth
-----
***** BOTTOM OF DATA *****
```

Go to **Edit** and select **1** for **Create/Update Record**, as in the following:

```
File  Edit  Help
-----
1  1. Create/Update Record (EM)  Main Screen
Optio *. Delete Record (ED)      Scroll CSR
      3. Options (EO)

ds are:

U Update node      H View History      D Delete node
I Insert node      V View node        CL Clone node

Node Filter : *

LC Node Name      Secure+      External Client
Type Protocol Override Encryption Auth Auth
-----
***** BOTTOM OF DATA *****
```

This will bring up the following:

```
Secure+ Create/Update Panel
Option ==>

Node Name:  NEW.LOCAL.NODE      Type:  L      (Local or Remote)
-----
| Security Options | EA Parameters | SSL/TLS Parameters |
| ---            | --          | ---                |
-----
Secure+ Protocol:      Security Mode (Yes , No , Default to Local)
Enable SSL             N      Enable FIPS             N
Enable TLS 1.0         N      Enable SP800-131a Transition N
Enable TLS 1.1         N      Enable SP800-131a Strict   N
Enable TLS 1.2         N      Enable NSA Suite B 128 bit  N
Enable TLS 1.3         N      Enable NSA Suite B 192 bit  N

Auth Timeout:         120      Enable Override         Y
Alias Names:
____
____
____
TCP Information:
IPaddr: _____
Port:      _____

OK      Cancel
--      ---
```

Now place your cursor on **SSL/TLS Parameters** and hit **Enter**. This will bring up:

NOTE: Beginning with C:D z/OS 6.2, you will see the above **red** comment in the PARMFILE (this text is not present prior to CD z/OS 6.2). Beginning in C:D z/OS 6.2, Data Encryption will be forced on all Secure+ transitions.

The **Cipher Suites** can be left as is until it is known which will be used. It is recommended that the ciphers be coded on the local node entry, and only coded on the remote if that remote needs a different cipher.

Secure+ Admin Tool: "Certificate Label" information

Option ---> _____

Enter the label of the x.509 certificate for use within the box on the next page. The individual lines within the box will be concatenated to form a single string.

CERT.LABEL

NOTE: If running Connect:Direct z/OS 6.0 or later, this can be left blank and Secure+ will assume you are using a default certificate. You can then add the correct certificate later if needed. However, if left blank, you must have a default certificate defined in your keyring / key database before attempting a Secure+ transmission.

Hit **Enter** to return to the **Secure+ Create/Update Panel**.

```

Secure+ Create/Update Panel
Option ==>

Node Name:  NEW.LOCAL.NODE          Type:  L          (Local or Remote)
-----
| Security Options      | EA Parameters      | SSL/TLS Parameters |
| ---                  | --                | ---                |
-----

Enable Client Auth      N          (Yes , No , Default to Local)
Enable Data Encrypt     N          (Ignored, Forced to Y)

Certificate Label       | CERT.LABEL        |
Cipher Suites          | 0035002F00050004000A000900020001 |
Certificate Pathname    | *                  |
Certificate Common Name |                    |
-----

                                OK          Cancel
                                --          ---

```

You will need to add a keyring or key database name; place your cursor on **Certificate Pathname** and hit **Enter**.

```

Secure+ Admin Tool: "Certificate Path" information
Option ---> _____
More: +

This is a scrollable panel. Please page down for
more information and modifiable fields.

In the first box below, enter the Unix path name of the .KDB file containing
the x.509 certificates for use by Connect:Direct.

In the second box below, enter the password used when creating the .KDB file.

-----
| KEYRING _____ |
| _____ |
| _____ |
| _____ |
| _____ |
| _____ |
| _____ |
| Certificate _____ |
| Path: _____ |
| _____ |

```

If entering a keyring name, enter the name without a path and hit **Enter** to return to the **Secure+ Create/Update Panel**.

If entering a key database name, enter the name along with the path (e.g., /u/userid/key_database.kdb), then scroll down to the bottom and add the key database password on **Certificate Pass Phrase**.

<u>Certificate</u>	_____
<u>Pass Phrase:</u>	_____

NOTE: If you do not know what the correct keyring or key database is at this time, simply add any name as a keyring name; it can be altered later.

Hit **Enter** to return to the **Secure+ Create/Update Panel**.

Any settings that are made on the Local node will serve as a “default” for the entire system. Because of this, it is recommended that Secure+ be disabled on the Local (that is, all protocols be set to **N**) and **Enable Override** set to **Y**. See the ***Connect:Direct Secure+ Implementation Guide*** for an explanation of all the settings.

Place your cursor on **OK** and hit **Enter** to save.

Hitting **OK** saves the Local node to the PARMFILE (although the PARMFILE itself will need to be saved before it can be used) and a new node with **Type: R** will be opened. You can then fill out and define the remote node:

Secure+ Create/Update Panel

Option ==>

Node Name: NEW.REMOTE.NODE

Type: R

(Local or Remote)

Security Options

EA Parameters

SSL/TLS Parameters

--

Secure+ Protocol:

Security Mode (Yes , No , Default to Local)

Enable SSL

D

Enable FIPS

D

Enable TLS 1.0

D

Enable SP800-131a Transition

D

Enable TLS 1.1

D

Enable SP800-131a Strict

D

Enable TLS 1.2

Y

Enable NSA Suite B 128 bit

D

Enable TLS 1.3

N

Enable NSA Suite B 192 bit

D

Auth Timeout:

120

Enable Override

N

Alias Names:

TCP Information:

IPaddr:

Port:

OK

Cancel

--

Here **Enable TLS 1.2** is set to **Y**. As you complete each node and hit **OK**, a new node will open; this will continue until you hit **Cancel**. If you wish to enter another node later, you can then place an **I** (insert) next to any node to insert a new node into the PARMFILE. Also, unless specifically needed, it is recommended that Remote Node entries have **Enable Override** set to **N**.

NOTE: Once you go through the initial building of the Local and Remote nodes and finally hit **Cancel**, any additional inserting of remote nodes will not bring up a new remote node to edit when you hit **OK**.

With the Local node and Remote node entered in this example, once **Cancel** is hit, **Main Panel** now shows the following with the Remote set to **TLS 1.2**:

File Edit Help

Secure+ Admin Tool: Main Screen

Row 1 to 2 of 2

Option ==>

Table Line Commands are:

U Update node

H View History

D Delete node

I Insert node

V View node

CL Clone node

Node Filter : *

LC Node Name

Secure+

External Client

Type

Protocol

Override

Encryption

Auth

Auth

NEW.LOCAL.NODE

L

Disabled

Y

N

N

N

NEW.REMOTE.NODE

R

TLSV12

N

*

*

*

BOTTOM OF DATA

Saving the New Secure+ PARMFILE

After you have inserted and edited your local and remote nodes, go to the pull-down menu and select **File**, then select option **7 (Save As)**:

File Edit Help		
<u>7</u>	1. New (FN)	e+ Admin Tool: Main Screen
	2. Open (FO)	Row 1 to 2 of 2
	3. Close (FC)	Scroll CSR
	4. Info (FI)	ble Line Commands are:
	5. Report (FR)	
	*. Save Active (FS)	View History D Delete node
	7. Save as (FA)	View node CL Clone node
	*. Unload (FX)	
	9. Exit (FE)	

LC Node Name	Type	Secure+ Protocol	Override	Encryption	External Auth	Client Auth
NEW.LOCAL.NODE	L	Disabled	Y	N	N	N
NEW.REMOTE.NODE	R	*	*	*	*	*
***** BOTTOM OF DATA *****						

If the PARMFILE has not been set up correctly, particularly if the Local Node has not been configured, you will get errors that will not allow the PARMFILE to be saved. If there are only Warnings, then you can continue and save the PARMFILE (**F3** to get out of the comments).

You will be asked to enter a Pass Phrase for your Secure+ PARMFILE. Enter any combination of at least 32 upper- and lower-case alphabetic characters and numbers.

NOTE: Do not worry about remembering this Pass Phrase; it is used to randomly encrypt the PARMFILE, and you will not need to use or access this Pass Phrase after this.

Option ---> _____

Secure Admin Tool: Pass Phrase Generation

Please enter a Pass Phrase that
contains upper, lower, numeric, and
alphabetic character data, at least 32 bytes long.

<----- Ruler ----->

1 2 3 4 5
12345678901234567890123456789012345678901234567

Enter the desired name for your Secure+ PARMFILE on the **File Name:** line, then hit **Enter**.

```
Secure+ Admin Tool: File Selection

Option ---> _____

Enter file name for: OUTPUT SECURE PARM FILE

File
Name: CD.SECURE.PARMFILE Browse
                                     ---
File System Type:
1 1. MVS Cancel
                                     ---
```

Enter the correct C:D z/OS LINKLIB(s) on the STEPLIB; if you need to enter more than three datasets on the STEPLIB step, select option **2** to Edit the JCL, then add the remaining datasets to the STEPLIB in the JCL.

Make sure that you enter the name of your Secure+ Access File at the bottom of the panel.

NOTE: While not a requirement, it is recommended that the name of the Access File be similar (that is, the same high-level qualifier) as the Secure+ PARMFILE to make identification easier.

```
Secure+ Admin Tool: "Save As" information

Option ---> _____

What do you want to do with the generated JCL? (PF3/PF12 to cancel)
3 1. Browse 2. Edit 3. Submit Make Pass Phrase
                                     ----

Job statement information. Verify before proceeding.
=====> //JOB CD1F JOB (CDLEVL), 'CD-SECP',MSGCLASS=X,CLASS=A,NOTIFY=&SYSUID
=====> //*
=====> //*
=====> //*

Mgmt. Class _____ Volume Serial DASD01
Stg. Class _____
Data Class _____

Product Load library information. Verify before proceeding.
=====> //STEPLIB DD DISP=SHR,DSN=CD.SDGALINK
=====> //*
=====> //*

Access file Dsname
=====> CD.SECURE.ACCFILE
```

If option **3** is selected, the JCL will be generated and submitted. If you select option **2**, the JCL will be generated but opened for edit before it is submitted. If changes are needed, do so, then type **SUB** on the command line and hit **Enter** to submit.

NOTE: IMPORTANT!! If you choose to edit the JCL, the Access file is built as soon as the JCL is generated; once you are in the JCL, the Access File has already been created. Therefore, if you change the name of your Access file in the JCL, you will get errors.

You should receive a zero return code, meaning the PARMFILE was saved successful.

At this point, you can then shut down Connect:Direct, go into the INITPARMs and add the parameter

`SECURE.DSN=your_parmfile_name`

Once you save your INITPARMs, you can then bring up Connect:Direct.

Cipher Filtering and TLSv1.3 (CD z/OS 6.1 or Greater)

Beginning in Connect:Direct for z/OS 6.1, two important features have been added to the Secure+ PARMFILE:

1. **Support for TLSv1.3** (which requires z/OS 2.4 or greater)
2. **Cipher Filtering / Cipher Sorting**

Support for TLSv1.3

There are two things to configure the Secure+ PARMFILE entries for TLSv1.3 support:

First, on the **Security Options** panel, **Enable TLS 1.3** has been added.

```
Secure+ Create/Update Panel
Option ==>
Node Name:  LOCAL.NODE      Type:  L      (Local or Remote)
-----
| Security Options | EA Parameters | SSL/TLS Parameters |
| ---            | --          | ---                |
-----
Secure+ Protocol:
Enable SSL          N      Enable FIPS          N
Enable TLS 1.0      N      Enable SP800-131a Transition N
Enable TLS 1.1      N      Enable SP800-131a Strict   N
Enable TLS 1.2      N      Enable NSA Suite B 128 bit  N
Enable TLS 1.3      Y      Enable NSA Suite B 192 bit  N
Auth Timeout:      120      Enable Override          Y
Alias Names:
-----
TCP Information:
IPaddr:
Port:
-----
OK      Cancel
--      ---
```

Note: TLSv1.3 does not support FIPS; if FIPS is coded and TLSv1.3 is used, FIPS will be ignored.

Second, on the **SSL/TLS Parameters** panel, either on the Local entry or the Remote entry, the correct cipher(s) need to be coded.

Place your cursor on **Cipher Suites** and hit **Enter**.

```
Secure+ Create/Update Panel
Option ==>
Node Name:  LOCAL.NODE      Type:  L      (Local or Remote)
-----
| Security Options | EA Parameters | SSL/TLS Parameters |
| ---            | --          | ---                |
-----
Enable Client Auth  N      (Yes , No , Default to Local)
Enable Data Encrypt N
-----
Certificate Label   | CERT_LABEL |
Cipher Suites       | FFFF       |
Certificate Pathname| /u/userid/key_database.kdb |
Certificate Common Name |          |
-----
OK      Cancel
--      ---
```

You will see the following:

```
Option --->

      Cipher Filtering:None          Cipher Sorting:Strongest
      Update the order field below to enable and order Cipher Suites

  0  All Available Cipher Suites      Enabled Cipher Suites
  ==  =====
  1  TLS_AES_256_GCM_SHA384          TLS_AES_256_GCM_SHA384
  2  TLS_AES_128_GCM_SHA256          TLS_AES_128_GCM_SHA256
  ---  TLS_ECDHE_ECDSA_W_AES_256_GCM_SHA384
  ---  TLS_ECDHE_ECDSA_W_AES_256_CBC_SHA384
  ---  TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA
  ---  TLS_ECDHE_ECDSA_W_AES_128_GCM_SHA256
  ---  TLS_ECDHE_ECDSA_W_AES_128_CBC_SHA256
  ---  TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA

More:  +
```

Note: Even though this node only has TLS 1.3 turned on, all ciphers are displayed.

In Connect:Direct z/OS 6.1, only two ciphers are supported for TLSv1.3:

TLS_AES_256_GCM_SHA384 (hexadecimal cipher code **1302**)

TLS_AES_128_GCM_SHA256 (hexadecimal cipher code **1301**)

If neither of these ciphers are coded by both sides of the transfer, TLSv1.3 will not be performed.

Hit F3 to return to the **SSL/TLS Parameters** panel; you should now see the following:

```
Secure+ Create/Update Panel

Option ==>

Node Name:  LOCAL.NODE      Type:  L      (Local or Remote)
-----
| Security Options | EA Parameters | SSL/TLS Parameters |
| ---            | --          | ---                |
-----
Enable Client Auth      N      (Yes , No , Default to Local)
Enable Data Encrypt     N
-----
Certificate Label       | CERT_LABEL      |
Cipher Suites           | 13021301        |
Certificate Pathname     | /u/userid/key_database.kdb |
Certificate Common Name  |                  |
-----
                        OK      Cancel
                        --      ---
```

If multiple protocols are chosen, additional ciphers should be selected for protocols other than TLSv1.3.

Cipher Filtering / Cipher Sorting

Due to the growing number of ciphers that are supported in Secure+, Cipher Filtering was added to CD z/OS 6.1. With Cipher Filtering turned on, only the ciphers supported by the protocols you select will be displayed when selecting ciphers.

Note: When creating a new Secure+ PARMFILE, Cipher Filtering is turned on by default. If it is not desired, it will need to be turned off.

To turn Cipher Filtering on or off, go to the **Secure+ Admin Tool Main Screen**, select **Edit** from the pull-down menu, then choose **3. Options**.

```

File  Edit  Help
-----
Option  3  1. Create/Update Record (EM)      Screen      Row 1 to 2 of 2
         2. Delete Record (ED)              Scroll CSR
         3. Options (EO)
re:

U Update node      H View History      D Delete node
I Insert node      V View node      CL Clone node

Node Filter : *

LC Node Name      Secure+      External Client
Type Protocol Override Encryption Auth Auth
-----
LOCAL.NODE        L Disabled Y N N N
REMOTE.NODE1      R TLSV12 N Y * *
REMOTE.NODE2      R TLSV13 N Y N N
***** BOTTOM OF DATA *****

```

This brings up the **Secure+ Admin Tool Edit options** panel.

```

Secure+ Admin Tool Edit options

Option --->

Node      Show FLASH Panel at
Name Filter: *      Startup: N (Yes or No)

Confirm      Cipher Filtering: P (Protocol or None)
Prompt: Y (Yes or No)      Cipher Sorting : S (Strongest or Weakest)

```

Two changes on this panel new to CD z/OS 6.1:

- (1) **Cipher Filtering:** To turn on, select **P** (Protocol); to turn off, select **N** (None)
- (2) **Cipher Sorting:** Select **S** (Strongest) to display ciphers from the strongest descending to the weakest
Select **W** (Weakest) to display ciphers from the weakest ascending to the strongest

The defaults are Cipher Filtering **P** and Cipher Sorting **S**.

With Cipher Filtering set to Protocol, if you go back into the Local node just edited with only TLSv1.3 set to **Y** and look at the ciphers, you will now see:

```

Option --->

Cipher Filtering:Protocol      Cipher Sorting:Strongest
Update the order field below to enable and order Cipher Suites

O All Available Cipher Suites      Enabled Cipher Suites
== =====
1 TLS_AES_256_GCM_SHA384      TLS_AES_256_GCM_SHA384      More: +
2 TLS_AES_128_GCM_SHA256      TLS_AES_128_GCM_SHA256
_
_
_
_
_
_

```

Note: Notice now the panel says **Cipher Filtering:Protocol**.

Converting Pre-C:D z/OS 5.2 Secure+ PARMFILE to C:D z/OS 5.2 or Later

When upgrading to C:D z/OS 5.2 Secure+ from a previous release, the Secure+ PARMFILE and Access file must be converted to the new file format. The 5.2 Secure+ Admin tool cannot open a Secure+ PARMFILE created in a release prior to 5.2.

The **DGASCONV** utility performs the conversion by allocating a new Secure+ PARMFILE and using the old PARMFILE as input. Due to the possibility of secure passwords with Strong Password Encryption (SPE) being enabled in the previous release, this utility can also use DGADTQFX and IDCAMS to convert the TCQ/TCX, copy the NETMAP and copy the AUTH file.

NOTE: Secure+ PARMFILES created prior to Connect:Direct OS/390 4.5 used IDEAL encryption. If these files were used in later releases, warning message SITA905W was received during startup, indicating that the PARMFILE is not encrypted with the TDES algorithm. To check this, do a REPRO of the **Access file** to a flat file. Browse the flat file, turn on HEX mode, scroll to column 156 (x'9C'), then examine the fullword that begins at column 156 (x'9C'):

```
00000080 - TDES encrypted
000000D4 - IDEAL encrypted
```

If this value is 000000D4, then you will need to encrypt the PARMFILE from IDEAL to TDES before continuing with this conversion utility; go to section **Encrypting the Connect:Direct Secure+ PARMFILE with TDES** for instructions on how to convert to TDES.

Sample JCL, **DGAJCONV**, in the **hlq.SDGAJCL** data set should be tailored to your environment and executed to perform this conversion. Use the instructions within the DGAJCONV JCL to assist with this JCL tailoring.

The execution of **DGAJCONV** will produce a report in the output DD called **REPORT**. Use this report to determine the scope of the changes made by the utility and to make recommended manual updates to any action produced in the report.

Once completed, the new Secure+ PARMFILE must be specified on the **SECURE.DSN** initialization parameter before the PARMFILE can be used.

This conversion utility also produces a diagnostic trace to the DD called **TRACEO**, which is commented out in the sample JCL. If you do not want to execute the diagnostic trace, leave the DD as a comment.

NOTE: Going from a pre-C:D z/OS 5.2 version to C:D z/OS 5.2 or later, the names of the cipher suites within the Secure+ PARMFILE changed for an **SSL_** prefix to a **TLS_** prefix. These cipher names are only used to identify them within Connect:Direct Secure+; System SSL looks at the hexadecimal cipher code. For example, the pre-C:D z/OS 5.2 cipher **SSL_RSA_AES_256_SHA**, which has a hex code of x'2F', is the same as the C:D 5.2 cipher **TLS_RSA_WITH_AES_256_CBC_SHA** which, as of z/OS 2.1, has a hex code of x'002F'.

The following are examples of running the DGASCONV Secure+ Conversion Utility:

1. **Strong Password Encryption (SPE) Used**
2. **Strong Password Encryption (SPE) NOT Used**

NOTE: If the Secure+ PARMFILE has a member called **.PASSWORD**, then SPE is being used; if **.PASSWORD** is not present in the Secure+ PARMFILE, then SPE is not being used.

Strong Password Encryption (SPE) Used

You will need to run the full version of the DGASCONV utility, which also makes changes to the AUTH, NETMAP and TCQ/TCX files. The instructions in the DGASCONV utility describe the changes needed.

If the Secure+ PARMFILE has not been converted the TDES encryption, this conversion utility will not work (since ICSF does not support IDEAL).

The following is an example on the DGASCONV JCL converting a C:D z/OS 5.1 Secure+ PARMFILE to the new C:D z/OS 5.2 PARMFILE:

```
//SDGA CONV JOB (LEVEL),'PARMFILE CONV',CLASS=A,MSGLEVEL=(1,1),
//          NOTIFY=userid,MSGCLASS=X,TIME=1440,REGION=0M
/*JOBPARM LINES=999999
//*
//DEFINE1 EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
DELETE (hlq.CD52.CONV.ACCESS) CLUSTER
DELETE (hlq.CD52.CONV.PARMFILE) CLUSTER
DELETE (hlq.CD52.CONV.TCQ) CLUSTER
DELETE (hlq.CD52.CONV.TCX) CLUSTER
DELETE (hlq.CD52.CONV.AUTH) CLUSTER
DELETE (hlq.CD52.CONV.NETMAP) CLUSTER
SET MAXCC = 0
DEFINE CLUSTER -
  (NAME(hlq.CD52.CONV.ACCESS) -
  RECORDS(1 4) -
  VOLUMES(volser) -
  INDEXED -
  FREESPACE(0 0) -
  NOIMBED -
  KEYS(18 2) -
  RECORDSIZE(500 500) -
  NOREPLICATE -
  SHAREOPTIONS(1 3)) -
DATA -
  (CONTROLINTERVALSIZE(4096) -
  NAME(hlq.CD52.CONV.ACCESSSS.DATA)) -
INDEX -
  (CONTROLINTERVALSIZE(512) -
  NAME(hlq.CD52.CONV.ACCESS.INDEX))
*
DEFINE CLUSTER -
  (NAME(hlq.CD52.CONV.PARMFILE) -
  RECORDS(1 4) -
  VOLUMES(volser) -
  INDEXED -
  FREESPACE(0 0) -
  NOIMBED -
  KEYS(18 2) -
  RECORDSIZE(1292 4086) -
  NOREPLICATE -
  SHAREOPTIONS(3 3)) -
DATA -
  (CONTROLINTERVALSIZE(4096) -
  NAME(hlq.CD52.CONV.PARMFILE.DATA)) -
```



```

INDEX -
  (CONTROLINTERVALSIZE (512) -
    NAME (hlq.CD52.CONV.PARMFILE.INDEX) )
*
DEFINE CLUSTER -
  (NAME (hlq.CD52.CONV.TCX) -
    RECORDS (1) -
    VOLUMES (volser) -
    NUMBERED -
    REUSE -
    RECORDSIZE (1017 1017) -
    SHAREOPTIONS (3 3) ) -
  DATA -
    (CONTROLINTERVALSIZE (1024) -
      NAME (hlq.CD52.CONV.TCX.DATA.COMP) )
*
DEFINE CLUSTER -
  (NAME (hlq.CD52.CONV.TCQ) -
    RECORDS (4000) -
    VOLUMES (volser) -
    NUMBERED -
    REUSE -
    RECORDSIZE (1529 1529) -
    SHAREOPTIONS (3 3) ) -
  DATA -
    (CONTROLINTERVALSIZE (1536) -
      NAME (hlq.CD52.CONV.TCQ.DATA.COMP) )
*
DEFINE CLUSTER -
  (NAME (hlq.CD52.CONV.AUTH) -
    RECORDS (100 4) -
    VOLUMES (volser) -
    INDEXED -
    FREESPACE (15 10) -
    NOIMBED -
    KEYS (26 6) -
    RECORDSIZE (512 2048) -
    NOREPLICATE -
    SHAREOPTIONS (2) ) -
  DATA -
    (CONTROLINTERVALSIZE (4096) -
      NAME (hlq.CD52.CONV.AUTH.DATA.COMP) ) -
  INDEX -
    (CONTROLINTERVALSIZE (512) -
      NAME (hlq.CD52.CONV.AUTH.INDEX.COMP) )
*
DEFINE CLUSTER -
  (NAME (hlq.CD52.CONV.NETMAP) -
    RECORDS (50 4) -
    VOLUMES (volser) -
    INDEXED -
    FREESPACE (0 0) -
    NOIMBED -
    KEYS (18 2) -
    RECORDSIZE (100 4086) -
    NOREPLICATE -
    SHAREOPTIONS (3 3) )

```

```

DATA                                     -
  (CONTROLINTERVALSIZE(4096)           -
    NAME(hlq.CD52.CONV.NETMAP.DATA) )  -
INDEX                                   -
  (CONTROLINTERVALSIZE(512)            -
    NAME(hlq.CD52.CONV.NETMAP.INDEX) )
/*
//REPRO      EXEC PGM=IDCAMS
//OAUTH      DD DISP=OLD,DSN=hlq.OLD51.AUTH
//AUTHFILE   DD DISP=OLD,DSN=hlq.CD52.CONV.AUTH
//ONETMAP    DD DISP=OLD,DSN=hlq.OLD51.NETMAP
//NETMAP     DD DISP=OLD,DSN=hlq.CD52.CONV.NETMAP
//SYSPRINT   DD SYSOUT=*
//SYSIN      DD *
      REPRO INFILE(OAUTH)   OUTFILE(AUTHFILE)
      REPRO INFILE(ONETMAP) OUTFILE(NETMAP)
/*
//BLDTCQ     EXEC PGM=DGADTQFX,PARM=DETAIL
//STEPLIB    DD DISP=SHR,DSN=hlq.HOST5200.SDGALINK
//SYSOUT     DD SYSOUT=*
//TCQIN      DD DISP=SHR,DSN=hlq.OLD51.TCQ
//TCXIN      DD DISP=SHR,DSN=hlq.OLD51.TCX
//TCQOUT     DD DISP=SHR,DSN=hlq.CD52.CONV.TCQ
//TCXOUT     DD DISP=SHR,DSN=hlq.CD52.CONV.TCX
//TCQINRPT   DD SYSOUT=*
/*
//CONVERT    EXEC PGM=DGASCONV
//STEPLIB    DD DISP=SHR,DSN=hlq.HOST5200.SDGALINK
/*  OPARMFIL = OLD S+ PARMFILE
/*  OACCESS  = OLD S+ ACCESS FILE
/*  PARMFILE = NEW S+ PARMFILE
/*  ACCESS   = NEW S+ ACCESS FILE
/*  AUTHFILE = NEW AUTH FILE
/*  TCXOUT   = NEW TCX FILE
/*  TCQOUT   = NEW TCQ FILE
/*  NETMAP   = NEW NETMAP FILE
/*  REPORT   = S+ CONVERSION REPORT
/*  TRACEO   = TRACE OUTPUT (OPTIONAL)
//OPARMFIL   DD DISP=SHR,DSN=hlq.OLD51.PARMFILE
//OACCESS    DD DISP=SHR,DSN=hlq.OLD51.ACCESS
//PARMFIL    DD DISP=OLD,DSN=hlq.CD52.CONV.PARMFILE
//ACCESS     DD DISP=OLD,DSN=hlq.CD52.CONV.ACCESS
//AUTHFILE   DD DISP=OLD,DSN=hlq.CD52.CONV.AUTH
//TCXOUT     DD DISP=OLD,DSN=hlq.CD52.CONV.TCX
//TCQOUT     DD DISP=OLD,DSN=hlq.CD52.CONV.TCQ
//NETMAP     DD DISP=OLD,DSN=hlq.CD52.CONV.NETMAP
//REPORT     DD SYSOUT=*
/*TRACEO     DD SYSOUT=*
//

```

This should complete with RC=0000. If there are problems and the debugging is needed, you can uncomment the TRACEO DD and resubmit the JCL.

Strong Password Encryption (SPE) NOT Used

The steps creating, REPROing, and building the AUTH, NETMAP and TCQ/TCX files can be removed. However, the CONVERT step still needs to have the AUTH, NETMAP and TCQ/TCX files referenced. That is, these DDs must be present with valid files; the DDs cannot be commented out or deleted.

Since these files will not be affected in any way, you can reference your current C:D 5.2 AUTH, NETMAP and TCQ/TCX files rather than having to create new ones just for this utility.

The following is an example on the DGASCONV JCL converting a C:D z/OS 5.1 Secure+ PARMFILE to the new C:D z/OS 5.2 PARMFILE:

```
//SDGA CONV JOB (LEVEL), 'PARMFILE CONV', CLASS=A, MSGLEVEL=(1,1),
//          NOTIFY=userid, MSGCLASS=X, TIME=1440, REGION=0M
/*JOBPARM LINES=999999
//*
//DEFINE1 EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
DELETE (hlq.CD52.CONV.ACCESS) CLUSTER
DELETE (hlq.CD52.CONV.PARMFILE) CLUSTER
SET MAXCC = 0
DEFINE CLUSTER
(NAME (hlq.CD52.CONV.ACCESS)
RECORDS(1 4)
VOLUMES(volser)
INDEXED
FREESPACE(0 0)
NOIMBED
KEYS(18 2)
RECORDSIZE(500 500)
NOREPLICATE
SHAREOPTIONS(1 3))
DATA
(CONTROLINTERVALSIZE(4096)
NAME (hlq.CD52.CONV.ACCESSSS.DATA))
INDEX
(CONTROLINTERVALSIZE(512)
NAME (hlq.CD52.CONV.ACCESS.INDEX))
*
DEFINE CLUSTER
(NAME (hlq.CD52.CONV.PARMFILE)
RECORDS(1 4)
VOLUMES(volser)
INDEXED
FREESPACE(0 0)
NOIMBED
KEYS(18 2)
RECORDSIZE(1292 4086)
NOREPLICATE
SHAREOPTIONS(3 3))
DATA
(CONTROLINTERVALSIZE(4096)
NAME (hlq.CD52.CONV.PARMFILE.DATA))
INDEX
```

```

        (CONTROLINTERVALSIZE(512)                -
        NAME(hlq.CD52.CONV.PARMFILE.INDEX))
// *
// CONVERT EXEC PGM=DGASCONV
// STEPLIB DD DISP=SHR,DSN=hlq.HOST5200.SDGALINK
// * OPARMFIL = OLD S+ PARMFILE
// * OACCESS = OLD S+ ACCESS FILE
// * PARMFILE = NEW S+ PARMFILE
// * ACCESS = NEW S+ ACCESS FILE
// * AUTHFILE = NEW AUTH FILE
// * TCXOUT = NEW TCX FILE
// * TCQOUT = NEW TCQ FILE
// * NETMAP = NEW NETMAP FILE
// * REPORT = S+ CONVERSION REPORT
// * TRACEO = TRACE OUTPUT (OPTIONAL)
// OPARMFIL DD DISP=SHR,DSN=hlq.OLD51.PARMFILE
// OACCESS DD DISP=SHR,DSN=hlq.OLD51.ACCESS
// PARMFILE DD DISP=OLD,DSN=hlq.CD52.CONV.PARMFILE
// ACCESS DD DISP=OLD,DSN=hlq.CD52.CONV.ACCESS
// AUTHFILE DD DISP=OLD,DSN=hlq.CD52.AUTH
// TCXOUT DD DISP=OLD,DSN=hlq.CD52.TCX
// TCQOUT DD DISP=OLD,DSN=hlq.CD52.TCQ
// NETMAP DD DISP=OLD,DSN=hlq.CD52.NETMAP
// REPORT DD SYSOUT=*
// *TRACEO DD SYSOUT=*
//

```

This should complete with RC=0000. If there are problems and the debugging is needed, you can uncomment the **TRACEO DD** and resubmit the JCL.

NOTE: The AUTHFILE, TCXOUT, TCQOUT and NETMAP DDs can be your current C:D z/OS 5.2 files since they will not be updated. But it is NOT recommended that these be dummied, as it will probably cause problems with both your new Secure+ PARMFILE and with your new instance of C:D z/OS 5.2.

Making a New Pass Phrase for the Connect:Direct Secure+ PARMFILE

Occasionally, the Connect:Direct Secure+ PARMFILE may fail with an error like the following:

```
RCDECYP - CANNOT DECRYPT OLD PARMFILE
RECORD DECRYPT ERROR
```

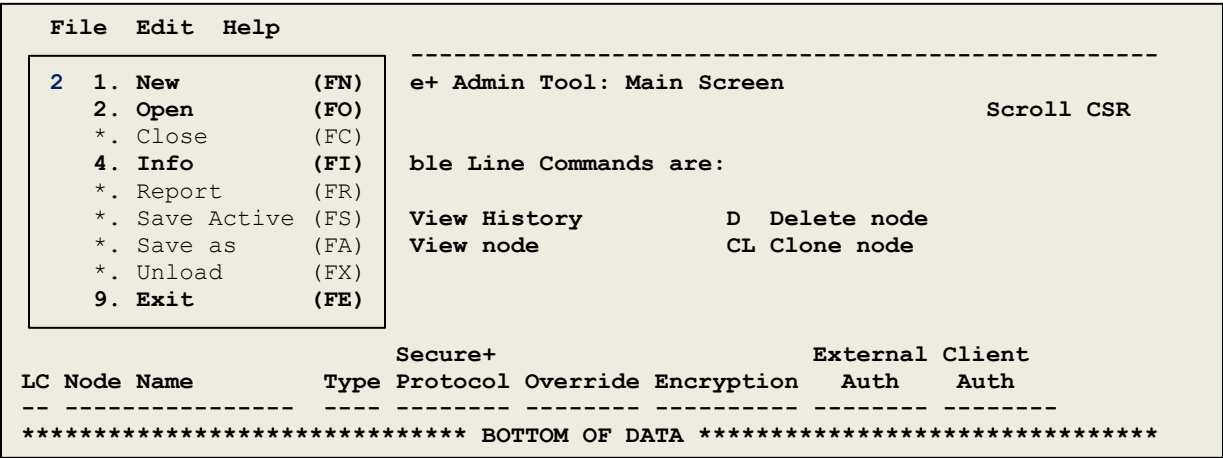
While this will typically occur when doing the Secure+ conversion from pre-C:D z/OS 5.2 to C:D z/OS 5.2, it is not exclusive to this conversion.

Basically, this error means that the keys that exist in your Access file are a longer length than the conversion program (DGASCONV) is expecting. When you see this error, you will need to “re-key” or make a new pass phrase for your PARMFILE before being able to successfully run the conversion job.

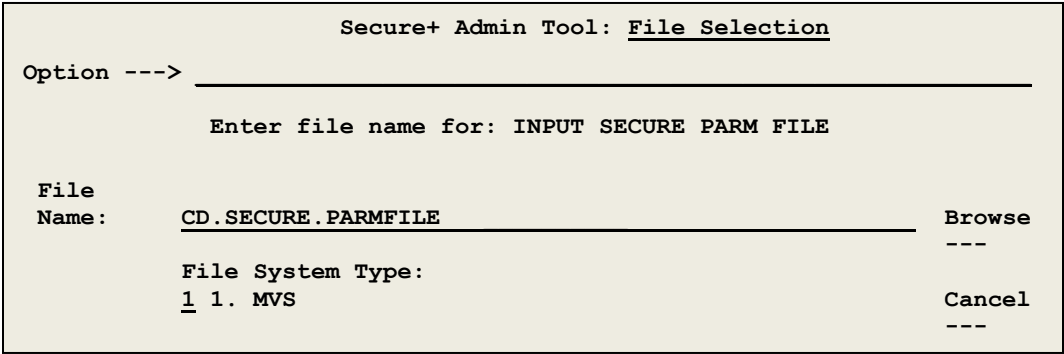
CAUTION: Before you begin this procedure, make copies of the Secure+ PARMFILE and the Access file (using **Save As** in the Secure+ Admin Tool) to enable you to back out of this change if necessary.

To make a new pass phrase for your PARMFILE:

- 1. Go into the UI and open the Secure+ Admin Tool (**ADMIN;S**, option **SA**)
- 2. To display the Secure+ Admin Tool File Selection panel, select **File** and type **2**.



- 3. Type the name of your Secure+ PARMFILE, or if you are unsure, enter a partial prefix followed by an asterisk (*), select **Browse**, press **Enter**, then type **S** beside the file in the list and press **Enter**.



4. From the File menu, select option **7 (Save as)** and press **Enter**. If you receive warnings, hit **F3** to bypass.

File Edit Help			-----	
<u>7</u>	1. New	(FN)	e+ Admin Tool: Main Screen	Row 1 to 7 of 7
	2. Open	(FO)		Scroll CSR
	3. Close	(FC)		
	4. Info	(FI)	ble Line Commands are:	
	5. Report	(FR)		
	*. Save Active	(FS)	View History	D Delete node
	7. Save as	(FA)	View node	CL Clone node
	*. Unload	(FX)		
	9. Exit	(FE)		
E.PARMFILE				

LC	Node Name	Type	Secure+ Protocol	Override	Encryption	External Auth	Client Auth
---	---	---	---	---	---	---	---
---	CDWIN48	R	*	N	*	*	N
---	CDWIN60	R	*	N	*	*	N
---	SC.ZOS.CD61T	R	*	N	*	*	N
---	SC.ZOS.JMCD61T	R	*	N	*	*	N
---	SC.ZOS.JMCGE62	L	Disabled	Y	Y	N	N
---	SC.ZOS.JMCGE62T	R	*	N	*	*	N

5. Select the name for the new Secure+ PARMFILE, then hit **Enter**.

```

Secure+ Admin Tool: File Selection

Option ---> _____

Enter file name for: OUTPUT SECURE PARM FILE

File
Name: CD.SECURE.NEWPASS.PARMFILE Browse
      _____
File System Type:
1 1. MVS Cancel
      _____

```

6. On the **Save As** information panel, do the following:
- Select **1, 2** or **3** (doesn't matter which, but you need something entered), but do not hit **Enter** yet.
 - Go to the top right and select the text string **Make Pass Phrase** and press **Enter**.

```

Secure+ Admin Tool: "Save As" information

Option ---> _____

What do you want to do with the generated JCL? (PF3/PF12 to cancel)
3 1. Browse 2. Edit 3. Submit Make Pass Phrase
-----

Job statement information. Verify before proceeding.
=====> //JOBCD1F JOB (CDLEVL),'CD-SECP',MSGCLASS=X,CLASS=A,NOTIFY=&SYSUID
=====> //*
=====> //*
=====> //*

Mgmt. Class _____ Volume Serial DASD02
Stg. Class _____
Data Class _____

Product Load library information. Verify before proceeding.
=====> //STEPLIB DD DISP=SHR,DSN=CD.SDGALINK
=====> //*
=====> //*

Access file Dsname
=====> CD.SECURE.NEWPASS.ACCFILE

```

7. The panel will state that a passphrase already exists and ask you if you want to make a new one; select **OK** and hit **Enter**.

Command Prompt

Secure+ Admin Tool: Confirmation Prompt

Option ---> _____

A passphrase currently exists. Are you sure you want to make a new one?
(Current private keys will be re=keyed)

OK Cancel

8. Type a **Pass Phrase**; this must be a combination of numbers and upper and lower case letters.

NOTE: You will not need to remember this phrase; it is only used to encrypt the PARMFILE.

When done, press **Enter**.

Option ---> _____

Secure Admin Tool: Pass Phrase Generation

Please enter a Pass Phrase that
contains upper, lower, numeric, and
alphabetic character data, at least 32 bytes long.

|-----|

<----- Ruler ----->

1 2 3 4 5
12345678901234567890123456789012345678901234567

9. Once you hit **Enter**, you will again be at the **"Save As" information** panel; go to the bottom and make sure the Access file name is correct (change the name for the Access file if necessary). Also make sure your STEPLIB is correct.

```
Secure+ Admin Tool: "Save As" information
Option ---> _____

What do you want to do with the generated JCL? (PF3/PF12 to cancel)
3 1. Browse 2. Edit 3. Submit Make Pass Phrase
-----

Job statement information. Verify before proceeding.
=====> //JOB CD1H JOB (CDLEVL), 'CD-SECP', MSGCLASS=X, CLASS=A, NOTIFY=&SYSUID
=====> // *
=====> // *
=====> // *

Mgmt. Class _____ Volume Serial DASD03
Stg. Class _____
Data Class _____

Product Load library information. Verify before proceeding.
=====> //STEPLIB DD DISP=SHR, DSN=CD.SDGALINK
=====> // *
=====> // *

Access file Dsname
=====> CD.SECURE.NEWPASS.ACCFILE
```

10. To submit the Save as-generated JCL, select option **3-Submit** and press **Enter**. If you want to view or edit the JCL before submitting, choose **2-Edit** and press **Enter**.

NOTE: IMPORTANT!! If you choose to edit the JCL, the Access file is built as soon as the JCL is generated; once you are in the JCL, the Access File has already been created. Therefore, **if you change the name of your Access file in the JCL, you will get errors.**

11. You should receive a zero return code, meaning the PARMFILE was saved successful.

The new PARMFILE **CD.SECURE.NEWPASS.PARMFILE** has been rekeyed to the new pass phrase; the previous PARMFILE has not been altered.

Secure+ PARMFILE Cloning

Cloning can be done only on the Local Node.

NOTE: Cloning does not work with older versions of C:D prior to C:D z/OS 5.2 (with **APAR PI33615** applied).

The “cloned” Secure+ PARMFILE and Access file must be saved to new names (that is, the Cloning procedure does a **Save As** and not a **Save Active**).

In the cloned Secure+ PARMFILE, the original Local node from the “old” PARMFILE is saved as a Remote node. Therefore, a new Local Node name is required for the cloned PARMFILE.

The original PARMFILE remains unaltered, but once the new cloned PARMFILE is saved, it cannot be undone; you will have to delete it and redo this entire cloning process to change it.

To begin, open the Secure+ Admin Tool and open the PARMFILE that you are going to clone.

Enter **CL** next to the Local Node and hit **Enter**.

```

File  Edit  Help
-----
SC.ZOS.JMCGE52          Secure+ Admin Tool: Main Screen          Row 1 to 9 of 25
Option ==>                                   Scroll CSR

                                Table Line Commands are:

U Update node            H View History            D Delete node
I Insert node            V View node                CL Clone node

Node Filter : *

LC Node Name              Type  Secure+ Protocol Override Encryption  External Client
-----
CDWIN48                   R    *           N           *           *           N
CDWIN60                   R    *           N           *           *           N
SC.ALT.CD52T              R    *           N           *           *           N
SC.ZOS.CD51T              R    *           N           *           *           N
SC.ZOS.JMCD50T            R    *           N           *           *           N
SC.ZOS.JMCD51T            R    *           N           *           *           N
CL SC.ZOS.JMCGE52         L    Disabled  Y           Y           N           N
SC.ZOS.JMCGE52T          R    *           N           *           *           N
TESTADAPTER-1            R    *           *           *           *           *

```

You should see the **Clone Local Record Prompt** panel:

Local Record Clone

Option ---> _____

Secure+ Admin Tool: Clone Local Record Prompt

You are attempting to clone the Local Node Record. If you continue, the PARMFILE will be required to perform a "Save As" function to save as a new PARMFILE name.

You should consider a backup of the PARMFILE before you continue.

The cloning process involves changing the current local record to TYPE of R then allowing the new local node name to be entered for the new node.

Do you wish to continue with this Cloning Process?

OK
--Cancel

Select **OK**.

This is the first of several chances to cancel out of the cloning process. If **Cancel** is chosen for any of these, the new PARMFILE will not be saved, but you will be placed into the Secure+ Admin Tool with the "updated" PARMFILE that you can then exit without saving.

Add your new Local Node name in the **Node Name** field:

Secure+ Create/Update Panel

Option ==> _____

Node Name: NEW.LOCAL.NODEType: L(Local or Remote)

Security OptionsEA ParametersSSL/TLS Parameters

Secure+ Protocol:

Security Mode (Yes , No , Default to Local)

Enable SSL N

Enable FIPS N

Enable TLS 1.0 N

Enable SP800-131a Transition N

Enable TLS 1.1 N

Enable SP800-131a Strict N

Enable TLS 1.2 N

Enable NSA Suite B 128 bit N

Enable NSA Suite B 192 bit N

Auth Timeout: 120

Enable Override Y

Alias Names:

TCP Information:

IPaddr:

Port:

OK
--Cancel

And on the **SSL/TLS Parameters** panel, make any changes, if needed, for your **Certificate Label**, **Cipher Suites** and **Certificate Pathname** information. Then select **OK**.

```

Secure+ Create/Update Panel
Option ==> _____
Node Name:  NEW.LOCAL.NODE      Type:  L      (Local or Remote)
-----
| Security Options | EA Parameters | SSL/TLS Parameters |
| ---            | --          | ---                |
-----
Enable Client Auth      N      (Yes , No , Default to Local)
Enable Data Encrypt     Y
-----
Certificate Label       | certificatelabel |
Cipher Suites           | 002F0009000A    |
Certificate Pathname     | key database (and path) or keyring |
Certificate Common Name  |                  |
-----
                                OK      Cancel
                                --      ---

```

You will see the **Clone Local Record Prompt** panel again to allow you to exit the cloning process. If you wish to continue, select **OK**.

```

Local Record Clone
Option ---> _____

Secure+ Admin Tool: Clone Local Record Prompt

You have attempted to clone the Local Node Record and the
Admin tool is going to perform a "SAVE AS" function now
to preserve these changes before continuing. You will be
prompted to enter a new data set name for this new
PARMFILE.

After the "SAVE AS", the Admin tool will close and open
the new cloned PARMFILE.

OK      Cancel
--      ---

```

If you enter an invalid Certificate Label or Pathname, you will see a warning or error similar to the following:

```

Menu Utilities Compilers Help
-----
BROWSE  SYS19077.T175431.RA000.JMCGE2.R0A24245  Line 0000000000 Col 001 080
Command ==>                                     Scroll ==> CSR
***** Top of Data *****
Warning: NEW.LOCAL.NODE      Secure.SSL.Cert.Trusted validates with Error
                                129 ENOENT , ensure the specification for
                                SECURE.SSL.PATH.PREFIX in the Connect:Direct
                                init parms results in a valid path
***** Bottom of Data *****

061: Warnings have occurred during Audit.

```

Hit **F3** to exit from this panel.

On the **Cloning File Selection** panel, enter the new names for the cloned PARMFILE and Access File, then hit **Enter**.

```
Secure+ Admin Tool: Cloning File Selection

Option ---> _____

Enter file name for: CLONE OUTPUT SECURE PARM FILE
(Must be new file names)

Parm File
Name: SECURE.CLONE.PARMFILE

Access File
Name: SECURE.CLONE.ACCFILE

File System Type:
1 1. MVS

Cancel
---
```

Make sure you have the correct Product Load Library entered and either select **2** to edit the JCL before submitting or **3** to submit the JCL.

```
Secure+ Admin Tool: "Save As" information

Option ---> _____

What do you want to do with the generated JCL? (PF3/PF12 to cancel)
3 1. Browse 2. Edit 3. Submit Make Pass Phrase
-----

Job statement information. Verify before proceeding.
=====> //JOB CARD JOB (CDJOB1), 'CD-SECP', MSGCLASS=X, CLASS=A, NOTIFY=&SYSUID
=====> //*
=====> //*
=====> //*

Mgmt. Class _____ Volume Serial ARTS01
Stg. Class _____
Data Class _____

Product Load library information. Verify before proceeding.
=====> //STEPLIB DD DISP=SHR, DSN=CD.SDGALINK
=====> //*
=====> //*

Access file Dsname
=====> SECURE.CLONE.ACCFILE
```

NOTE: IMPORTANT!! If you choose to select **2** to edit the JCL, the Access file is built as soon as the JCL is generated; once you are in the JCL, the Access File has already been created. Therefore, if you change the name of your Access file in the JCL, you will get errors.

Once submitted, you will see the **Confirmation Prompt** panel:

Command Prompt

Secure+ Admin Tool: Confirmation Prompt

Option ---> _____

You are performing a Save As due to a Clone request. If you continue and save the parameter file it can not be reversed.
Are you sure you want to continue?

OK Cancel

-- ---

If you choose **Cancel**, the PARMFILE will not be saved (the Access file will be built however) and you will be put back into the Secure+ Admin Tool with the new cloned PARMFILE. You can then either exit without saving or make additional changes and do a **Save As**.

Encrypting the Pre-C:D z/OS 5.2 Secure+ PARMFILE with TDES

Beginning with Connect:Direct OS/390 4.5 and going through Connect:Direct for z/OS 5.1, the Secure+ PARMFILE was created using TDES encryption. However, prior to C:D 4.5, IDEAL encryption was used to encrypt the Secure+ PARMFILE. If an IDEAL PARMFILE is used beginning in C:D 4.5, a **SITA905W** warning message was received during startup, indicating that the PARMFILE is not encrypted with the TDES algorithm, as in the following:

```
SITA905W Secure+ Parameter file not encrypted using TDES
```

Starting in C:D z/OS 5.2, since ICSF is now used for encryption, a Secure+ PARMFILE can no longer be encrypted with IDEAL, since IDEAL is not supported by ICSF. That is, to run the conversion of a pre-C:D 5.2 PARMFILE to a C:D 5.2 PARMFILE or later, the PARMFILE must be encrypted with TDES.

To verify the encryption of your PARMFILE:

1. Browse your Secure+ Access file (or REPRO it to a flat file if you do not have a way to browse the Access file, since it is a VSAM file).
2. Whether browsing the Access or the flat file, switch to HEX mode. Go to offset x'9C' (decimal 156): if the fullword value is 00000080, then it is TDES; 000000D4 means it is IDEAL.

CAUTION: Before you begin this procedure, make copies of the Secure+ PARMFILE and the Access file (using **Save As** in the Secure+ Admin Tool) to enable you to back out of this change if necessary.

To encrypt a Secure+ PARMFILE with the TDES algorithm:

3. Go into the IUI and open the Secure+ Admin Tool (**ADMIN;S**, option **SA**)
4. To display the Secure+ Admin Tool File Selection panel, select **File** and type **2**.

File	Edit	Key Management	Help
<u>2</u>	1. New 2. Open *. Close *. Info... *. Rekey *. Save Active *. Save as... *. Unload 9. Exit	Secure+ Admin Tool: Main Screen Table Line Commands are: H View History D Delete node I Insert node Secure LC Node Name Type 123C Override Encryption Signature ExtAuth Autoupd ----- ***** BOTTOM OF DATA *****	

5. Type the name of your Secure+ PARMFILE, or if you are unsure, enter a partial prefix followed by an asterisk (*), select **Browse**, press **Enter**, then type **S** beside the file in the list and press **Enter**.

```

Secure+ Admin Tool: File Selection

Enter file name for: INPUT SECURE PARM FILE

File
Name:      CD.IDEAL.SECURE.PARMFILE      Browse
File System Type:
1 1. MVS  2. HFS                        Cancel

```

6. From the File menu, select option **7 (Save as)** and press **Enter**.

```

File  Edit  Key Management  Help
-----
7 1. New
2. Open
3. Close
4. Info...
*. Rekey
6. Save Active
7. Save as...
8. Unload
9. Exit

Secure+ Admin Tool: Main Screen
Row 1 to 10 of 13
Scroll CSR

Table Line Commands are:
H View History      D Delete node
I Insert node

Secure
LC Node Name      Type  123C  Override Encryption Signature ExtAuth Autoupd
-----
. CLIENT          R    NNYN   N          N          N          *          N
CDWIN48           R    NNYN   N          Y          N          *          N
CDWIN60           R    NNYN   N          Y          N          *          N
CDZOS52           R    NNYN   N          Y          N          *          N
NEEDLE_4200       R    NNYN   N          Y          *          *          N
OLLIE.CDTS3600    R    NNYN   N          *          N          N          N
SC.ALT.JMCGE51T   R    NYNN   N          *          N          *          N
SC.ZOS.JMCGE48T   R    NNNN   N          N          N          N          N
SC.ZOS.JMCGE50T   R    NNYN   N          N          N          N          N
SC.ZOS.JMCGE51T   L    NNNN   Y          N          N          N          N

```

There may be warnings; simply hit **F3** to exit the warning panel.

7. Select the name for the new Secure+ PARMFILE, then hit **Enter**.

```

Secure+ Admin Tool: File Selection

Enter file name for: OUTPUT SECURE PARM FILE

File
Name:      CD.TDES.SECURE.PARMFILE      Browse
File System Type:
1 1. MVS  2. HFS                        Cancel

```

8. On the **Save As information** panel, do the following:

- c. Select **1, 2** or **3** (doesn't matter which, but you need something entered), but do not hit **Enter** yet.
- d. Go to the top right and select the text string **Make Pass Phrase** and press **Enter**.

```
Secure+ Admin Tool: "Save As" information
(Or press PF3/PF12 to cancel)

What do you want to do with the generated JCL?
3 1. Browse 2. Edit 3. Submit Make Pass Phrase
Job statement information. Verify before proceeding.

=====> //JOB CD1F JOB (CDLEVL), 'CD-SECP', MSGCLASS=X, CLASS=A, NOTIFY=&SYSUID
=====> // *
=====> // *
=====> // *

Mgmt. Class _____ Volume Serial DASD02
Stg. Class _____
Data Class _____

Product Load library information. Verify before proceeding.

=====> //STEPLIB DD DISP=SHR, DSN=CD.SDGALINK
=====> // *
=====> // *

Access file Dsname (long names may need quotation)
=====> CD.SECURE.NEWPASS.ACCFILE
```

9. The panel will state that a passphrase already exists and ask you if you want to make a new one – select **OK** and hit **Enter**.

```
Command Prompt

Secure+ Admin Tool: Confirmation Prompt

Option: _____

A passphrase currently exists. Are you sure you want to make a new one?
(Current private keys will be re-keyed)

OK                               Cancel
```

10. Type a **Pass Phrase** – this must be a combination of numbers and upper and lower case letters – and press **Enter**.

```
Option ---> _____

Secure Admin Tool: Pass Phrase Generation

Please enter a Pass Phrase that
contains upper, lower, numeric, and
alphabetic character data, at least 32 bytes long.

-----
|
-----

<----- Ruler ----->
      1      2      3      4      5
12345678901234567890123456789012345678901234567
```


11. Once you hit **Enter**, you will again be at the “**Save As**” information panel; go to the bottom and make sure the Access file name is correct (change the name for the Access file if necessary). Also make sure your STEPLIB is correct.

```
Secure+ Admin Tool: "Save As" information
(Or press PF3/PF12 to cancel)

What do you want to do with the generated JCL?
3 1. Browse 2. Edit 3. Submit Make Pass Phrase
Job statement information. Verify before proceeding.

=====> //JOB CD1F JOB (CDLEVL), 'CD-SECP',MSGCLASS=X,CLASS=A,NOTIFY=&SYSUID
=====> //*
=====> //*
=====> //*

Mgmt. Class _____ Volume Serial DASD02
Stg. Class _____
Data Class _____

Product Load library information. Verify before proceeding.

=====> //STEPLIB DD DISP=SHR,DSN=CD.SDGALINK
=====> //*
=====> //*

Access file Dsname (long names may need quotation)
=====> CD.SECURE.NEWPASS.ACCFILE
```

12. To submit the Save as-generated JCL, select option **3-Submit** and press **Enter**. If you want to view or edit the JCL before submitting, choose **2-Edit** and press **Enter**.

NOTE: IMPORTANT!! If you choose to edit the JCL, the Access file is built as soon as the JCL is generated; once you are in the JCL, the Access File has already been created. Therefore, **if you change the name of your Access file in the JCL, you will get errors.**

13. You should receive a zero (0000) return code, meaning the PARMFILE was saved successful.

Strong Password Encryption (SPE)

Passwords are protected in the TCQ and AUTH files by encrypting them with Connect:Direct Secure+ proprietary "Polyalphabetic Substitution Cipher" which is a weak encryption. A stronger encryption algorithm, **TDESCBC112**, can be used if you add a **.PASSWORD** record to the Connect:Direct Secure+ parameter file. After you create this record and enable the Strong Password Encryption (SPE) feature, Connect:Direct Secure+ will need to be restarted. SPE protects Connect:Direct Secure+ passwords stored in the TCQ and AUTH files with the stronger algorithm.

NOTE: .PASSWORD

If SPE is already enabled and you need to disable it, go to **"Disabling Strong Password Encryption"**.

Implementing Strong Password Encryption

To implement Strong Password Encryption (SPE), a SPE record (**.PASSWORD**) must be added to the Secure+ PARMFILE in the same way any remote node record would be added.

In the C:D IUI, go to **ADMIN;S** and select option **SA** to go into the Secure+ Admin Tool.

You now need to either open a Secure+ PARMFILE that you already have created or build a new PARMFILE; if a new PARMFILE is needed, please refer to **"Manually Create Connect:Direct Secure+ PARMFILE"**.

1. In the Secure+ Admin Tool, you should initially see the following:

```
File  Edit  Help
-----
Secure+ Admin Tool: Main Screen
Option ===> _____ Scroll CSR

Table Line Commands are:

U Update node      H View History      D Delete node
I Insert node      V View node        CL Clone node

Node Filter : *_____

LC Node Name      Secure+      External Client
Type Protocol Override Encryption Auth Auth
-----
***** BOTTOM OF DATA *****
```

2. Either go to **File**, select option **2. Open** and enter the name of the PARMFILE that will be used, or type **FO** on the option line, press **Enter**, then enter the name of the Secure+ PARMFILE being used:

```
Secure+ Admin Tool: File Selection

Option ---> _____

Enter file name for: INPUT SECURE PARM FILE

File
Name:  CDZ.SECURE.PARMFILE      Browse
-----

File System Type:
1 1. MVS      Cancel
-----
```

3. You will see something like the following:

```

File  Edit  Help
-----
LOCAL_NODE          Secure+ Admin Tool: Main Screen          Row 1 to 9 of 25
Option ==>          Scroll CSR

Table Line Commands are:

U Update node        H View History        D Delete node
I Insert node         V View node           CL Clone node

Node Filter : *

LC Node Name        Type  Secure+      External Client
      Protocol Override Encryption  Auth  Auth
-----
CDWIN_REMOTE        R    TLSV11      *          *      N
LOCAL_NODE          L    Disabled    *          *      N
NEW_REMOTE          R    TLSV12      *          *      N
UNIX_REMOTE         R    TLSV12      *          *      N
ZOS.C               *
ZOS.C               007: The Secure+ parm file has been read successfully.
  
```

Press **Enter** to remove the **007** message.

4. The Secure+ PARMFILE should look something like this:

```

File  Edit  Help
-----
LOCAL_NODE          Secure+ Admin Tool: Main Screen          Row 1 to 9 of 25
Option ==>          Scroll CSR

Table Line Commands are:

U Update node        H View History        D Delete node
I Insert node         V View node           CL Clone node

Node Filter : *

LC Node Name        Type  Secure+      External Client
      Protocol Override Encryption  Auth  Auth
-----
CDWIN_REMOTE        R    TLSV11      *          *      N
LOCAL_NODE          L    Disabled    *          *      N
NEW_REMOTE          R    TLSV12      *          *      N
UNIX_REMOTE         R    TLSV12      *          *      N
ZOS.CD52            R    Disabled    *          *      N
ZOS.CD60            R    TLSV12      *          *      N
  
```

5. To add the **.PASSWORD** record, either go to **Edit** and select **1** for **Create/Update Record**:

```

File  Edit  Help
-----
1  1. Create/Update Record (EM)  Main Screen
*  *. Delete Record (ED)          Scroll CSR
3  3. Options (EO)

ds are:

U Update node        H View History        D Delete node
I Insert node         V View node           CL Clone node

Node Filter : *

LC Node Name        Type  Secure+      External Client
      Protocol Override Encryption  Auth  Auth
-----
CDWIN_REMOTE        R    TLSV11      *          *      N
LOCAL_NODE          L    Disabled    *          *      N
NEW_REMOTE          R    TLSV12      *          *      N
UNIX_REMOTE         R    TLSV12      *          *      N
ZOS.CD52            R    Disabled    *          *      N
ZOS.CD60            R    TLSV12      *          *      N
  
```

or go down to any of the records in the PARMFILE, place a **I** on the **LC** line next to it, then press **Enter**.

```

File  Edit  Help
-----
SC.ZOS.JMCGE52      Secure+ Admin Tool: Main Screen      Row 1 to 9 of 25
Option ==>>        Scroll CSR

                        Table Line Commands are:

U Update node        H View History        D Delete node
I Insert node        V View node          CL Clone node

Node Filter : *

LC Node Name        Secure+
                    Type Protocol Override Encryption External Client
                    Type Protocol Override Encryption Auth Auth
-----
  CDWIN_REMOTE      R   TLSV11          N          *          *          N
I LOCAL_NODE        L   Disabled        N          *          *          N
  NEW_REMOTE        R   TLSV12          N          *          *          N
  UNIX_REMOTE       R   TLSV12          N          *          *          N
  ZOS.CD52           R   Disabled        N          *          *          N
  ZOS.CD60           R   TLSV12          N          *          *          N

```

6. Either way will bring up the following:

```

                        Secure+ Create/Update Panel
Option ==>>

Node Name:          Type: R      (Local or Remote)
-----
| Security Options  | EA Parameters  | SSL/TLS Parameters  |
| ---              | --            | ---                  |
-----
Secure+ Protocol:   Security Mode (Yes , No , Default to Local)
Enable SSL          D Enable FIPS                      D
Enable TLS 1.0      D Enable SP800-131a Transition D
Enable TLS 1.1      D Enable SP800-131a Strict   D
Enable TLS 1.2      D Enable NSA Suite B 128 bit D
                   D Enable NSA Suite B 192 bit   D

Auth Timeout:       120 Enable Override          D
Alias Names:        TCP Information:
                   IPaddr:
                   Port:

                                OK      Cancel
                                --      ---

```

7. Type **.password** (case does not matter) in the **Node Name:** line and press **Enter**.

This changes the screen to enable only fields that are appropriate for the **.PASSWORD** record:

```

                        Secure+ Create/Update Panel - SPE Parameters      <Change Pending>
Option ==>>

Node
.PASSWORD          N Enable SPE
                   (SPE not currently in use)

                        < >
Password Public Key | *
Algorithm Names    | TDESCBC112
-----
                                OK      Cancel
                                --      ---

```

Note: The <Change Pending> in the upper right means the record has not been saved yet.

- Change **Enable SPE** to **Y** and press **Enter**. Then click **OK** to complete the SPE record.

```
Secure+ Create/Update Panel - SPE Parameters      <Change Pending>
Option ==>

Node
.PASSWORD                                     Y   Enable SPE
                                                (SPE not currently in use)

----- < > -----
Password Public Key | *                      |
Algorithm Names    | TDESCBC112             |
-----

                                OK            Cancel
                                --            ---
```

- Save the parameter file using either **Save Active** (if this is the active Secure+ PARMFILE for your instance of Connect:Direct) or **Save As** to create a new PARMFILE.
- Restart C:D Secure+, making sure that the previous Secure+ PARMFILE (if a new PARMFILE was created or a PARMFILE other than the active Secure+ PARMFILE) is specified on the **SECURE.DSN** parameter in your initialization parameters.
- To verify that C:D Secure+ initialization is complete along with the SPE feature, after restarting C:D Secure+, go back into the Secure+ Admin Tool and view the **.PASSWORD** record; you should see **SPE currently in use** displayed to confirm that SPE has indeed been implemented.

```
Secure+ Create/Update Panel - SPE Parameters
Option ==>

Node
.PASSWORD                                     Y   Enable SPE
                                                (SPE currently in use)

----- < > -----
Password Public Key | C6A2.FAFB.E917.6FB4     |
Algorithm Names    | TDESCBC112             |
-----

                                OK            Cancel
                                --            ---
```

Note: The **.PASSWORD** record will now have a value for **Password Public Key**, which is regenerated each time C:D is recycled.

Disabling Strong Password Encryption

If the Strong Password Encryption feature is backed out inappropriately by deleting the .PASSWORD record while encrypted passwords exist in the TCQ and AUTH files in the SPE format, you will one or more error messages such as SITA461I, SITA463E, SAFB023W, SAFF016W, SAFC016W or SAFE016W.

Note: If the SPE feature is backed out inappropriately, the only option is to COLD start the TCQ and rebuild the entire AUTH file. Therefore, perform this procedure as documented, do not vary from the procedure.

1. Start the Secure+ Admin Tool - you should see the following:

```
File  Edit  Help
-----
Secure+ Admin Tool: Main Screen
Option ==> _____ Scroll CSR

Table Line Commands are:

U Update node      H View History      D Delete node
I Insert node      V View node        CL Clone node

Node Filter : * _____

LC Node Name      Secure+      External Client
Type Protocol Override Encryption Auth Auth
-----
***** BOTTOM OF DATA *****
```

2. Either go to **File** and select option **2. Open** or simply type **FO** on the option line and press **Enter**, then enter the name of the Secure+ PARMFILE:

```
Secure+ Admin Tool: File Selection

Option ---> _____

Enter file name for: INPUT SECURE PARM FILE

File
Name:  CDZ.SECURE.PARMFILE      Browse
                                           ---
File System Type:
  1 1. MVS                      Cancel
                                           ---
```

3. You should see the **Secure+ Admin Tool: Main Screen** panel.

```
File  Edit  Help
-----
LOCAL NODE      Secure+ Admin Tool: Main Screen      Row 1 to 9 of 25
Option ==> _____ Scroll CSR

Table Line Commands are:

U Update node      H View History      D Delete node
I Insert node      V View node        CL Clone node

Node Filter : * _____

LC Node Name      Secure+      External Client
Type Protocol Override Encryption Auth Auth
-----
. PASSWORD        R Enabled      N      *      *      *
CDWIN_REMOTE      R TLSV11       N      *      *      N
LOCAL_NODE        L Disabled     N      *      *      N
NEW_REMOTE        R TLSV12       N      *      *      N
UNIX_REMOTE       R TLSV12       N      *      *      N
ZOS.CD52          R Disabled     N      *      *      N
ZOS.CD60          R TLSV12       N      *      *      N
```

- Type **U** next to the **.PASSWORD** record and press **Enter**.

```

File  Edit  Help
-----
LOCAL_NODE          Secure+ Admin Tool: Main Screen          Row 1 to 9 of 25
Option ==>          Scroll CSR

                        Table Line Commands are:

U Update node        H View History        D Delete node
I Insert node        V View node          CL Clone node

Node Filter : *

LC Node Name        Type  Secure+      External Client
                        Protocol Override Encryption Auth      Auth
-----
U .PASSWORD         R    Enabled      N          *          *
CDWIN_REMOTE        R    TLSV11       N          *          N
LOCAL_NODE          L    Disabled     N          *          N
NEW_REMOTE          R    TLSV12       N          *          N
UNIX_REMOTE         R    TLSV12       N          *          N
ZOS.CD52            R    Disabled     N          *          N
ZOS.CD60            R    TLSV12       N          *          N

```

- On the SPE Parameters panel, type **N** next to the **Enable SPE** field and press **Enter**, then click **OK**.

```

Secure+ Create/Update Panel - SPE Parameters      <Change Pending>
Option ==>

Node
.PASSWORD                                     N   Enable SPE
                                                (SPE currently in use)

----- < > -----
Password Public Key | C6A2.FAFB.E917.6FB4          |
Algorithm Names    | TDESCBC112                    |
-----

                        OK      Cancel
                        --      ---

```

- Save the parameter file using either **Save Active** (if this is the active Secure+ PARMFILE for your instance of Connect:Direct) or **Save As** to create a new PARMFILE.
- Restart C:D Secure+, making sure that the previous Secure+ PARMFILE (if a new PARMFILE was created or a PARMFILE other than the active Secure+ PARMFILE) is specified on the **SECURE.DSN** parameter in your initialization parameters.
- To verify that C:D Secure+ initialization is complete with the SPE feature disabled, after restarting C:D Secure+, go back into the Secure+ Admin Tool and view the **.PASSWORD** record; you should see **SPE not currently in use** displayed to confirm that SPE has indeed been disabled.

```

Secure+ Create/Update Panel - SPE Parameters
Option ==>

Node
.PASSWORD                                     N   Enable SPE
                                                (SPE not currently in use)

----- < > -----
Password Public Key | C7B2.40FB.D716.5FC4          |
Algorithm Names    | TDESCBC112                    |
-----

                        OK      Cancel
                        --      ---

```

Using FIPS Mode

Connect:Direct provides the capability to execute securely in FIPS 140-2 mode. By default, Connect:Direct will run in non-FIPS mode and must be configured to run in FIPS mode. For information about System SSL in FIPS mode, see manual *z/OS V2R4 Cryptographic Services System Secure Sockets Layer Programming SC14-7495-40*.

FIPS mode is supported in Connect:Direct for z/OS, Windows and Unix. While **SP800-131a Transition** can use SSL, TLSv1.0 and 1.1, it is recommended TLSv1.2 be used. All other FIPS modes only support TLSv1.2.

NOTE: TLSv1.3 is not supported in FIPS mode and TLSv1.3 will be forced to No or disabled.

While in FIPS mode, Connect:Direct Secure Plus will open a FIPS mode keystore (that is, the keyring or key database); however, initialization fails if the keystore is not in FIPS mode. If Connect:Direct Secure+ is not run in FIPS mode, FIPS mode keystore can still be used without any issues.

The settings in the Connect:Direct for z/OS Secure+ Admin Tool:

Secure+ Create/Update Panel

Option ==>

Node Name: LOCAL.NODE

Type: L

(Local or Remote)

Security Options

EA Parameters

SSL/TLS Parameters

Secure+ Protocol:

Enable SSL N

Enable TLS 1.0 N

Enable TLS 1.1 N

Enable TLS 1.2 N

Security Mode (Yes , No , Default to Local)

Enable FIPS Y

Enable SP800-131a Transition N

Enable SP800-131a Strict N

Enable NSA Suite B 128 bit N

Enable NSA Suite B 192 bit N

Auth Timeout: 120

Enable Override Y

Alias Names:

TCP Information:

IPaddr:

Port:

OK

Cancel

The settings in the Connect:Direct for Windows and Connect:Direct for Unix Secure+ Admin Tool:

Edit Record

Node Name: .Local

Type: Local

Security Options

TLS/SSL Options

External Authentication

Secure+ Protocol

☒ Disable Secure+

☐ Enable SSL 3.0

☐ Enable TLS 1.0

☐ Enable TLS 1.1

☐ Enable TLS 1.2

☐ Default to Local Node

Security Modes

☐ Disable

☒ FIPS 140-2

☐ SP800-131A Transition

☐ SP800-131A

☐ Suite B 128 bit

☐ Suite B 192 bit

☐ Default to Local Node

Node or Copy Statement Override

Enable Override: ☐ Yes ☒ No ☐ Default to Local Node

Authentication Timeout: 120 seconds.

Update History (Last 3 Updates):

OK

Cancel

NOTE: The Connect:Direct for z/OS setting **Enable FIPS** is the same as the Connect:Direct for Windows and Connect:Direct for Unix setting **FIPS 140-2**.

While in FIPS mode, only certain ciphers are supported. During the TLS handshake, any non-FIPS mode ciphers are ignored.

If either **SP800-131a Transition** and **SP800-131a Strict** is selected, one of the following ciphers must be used:

- TLS_RSA_WITH_AES_128_CBC_SHA (002F)
- TLS_RSA_WITH_AES_256_CBC_SHA (0035)
- TLS_RSA_WITH_3DES_EDE_CBC_SHA (000A)

If **NSA Suite B 128 bit** is selected, one of the following ciphers must be used:

- TLS_ECDHE_ECDSA_W_AES_128_CBC_SHA256 (C023)
- TLS_ECDHE_ECDSA_W_AES_128_GCM_SHA256 (C02B)

If **NSA Suite B 192 bit** is selected, one of the following ciphers must be used:

- TLS_ECDHE_ECDSA_W_AES_256_CBC_SHA384 (C024)
- TLS_ECDHE_ECDSA_W_AES_256_GCM_SHA384 (C02C)

If configuring FIPS in Connect:Direct for Windows or Connect:Direct for Unix, the keystore is automatically built in FIPS mode, so there is nothing further that is needed. However, configuring FIPS in Connect:Direct for z/OS will require additional parameters and the keyring or key database must be built in FIPS mode.

Security Modes

Four different FIPS security modes are supported in Connect:Direct Secure+:

SP800-131 A Transition Mode

This enables the SP800-131a requirements in a transition mode and the following restrictions apply.

- DES and RC2 cipher algorithms are disabled
- MD5 signature algorithms are disabled
- RSA and DSA certificates with key length less than 1024-bits are disabled
- Support for TLSV1.2 is enabled; SSLV3, TLSV1.0 and TLSV1.1 are allowed but should be disabled
- Non-compliant TLS cipher suites are disabled

SP800-131 A Strict Mode

This enables the SP800-131a requirements in a strict mode and the following restrictions apply:

- DES, RC2 and two-key Triple DES cipher algorithms are disabled
- MD5 and SHA1 signature algorithms are disabled
- RSA and DSA certificates with key length less than 2048-bits are disabled
- EC certificates with key length less than 224-bits are disabled
- Protocol must be TLSV1.2; all other protocols are disabled.

NSA Suite B 128 bit

This enables NSA Suite B 128-bit restrictions. Suite B cryptography specifies the cryptographic algorithms that can be used in a Suite B compliant TLS1.2 session, and the following restrictions apply:

- Certificates must be ECC using elliptic curve secp256r1 or secp384r1.
- Protocol must be TLSV1.2, all others are disabled
- Cipher algorithm must be AES-128
- Key exchange algorithm must be ECDH
- Digital signature algorithm must be ECDSA
- Hashing algorithm must be SHA256

NSA Suite B 192 bit

This enables NSA Suite B 192-bit restrictions. Suite B cryptography specifies the cryptographic algorithms that can be used in a Suite B compliant TLS1.2 session, and the following restrictions apply:

- Certificates must be ECC using elliptic curve secp384r1.
- Protocol must be TLSV1.2, all others are disabled
- Cipher algorithm must be AES-256
- Key exchange algorithm must be ECDH
- Digital signature algorithm must be ECDSA
- Hashing algorithm must be SHA384

Configuring FIPS Mode in C:D z/OS

SP800-131a and **NSA Suite B** require System SSL to run in FIPS mode. This requires the initialization parameter **FIPS=YES** to be specified. If either **SP800-131** or **NSA Suite B** are enabled in the Secure+ PARMFILE, then the **FIPS** parameter is forced to **YES**.

In Connect:Direct for z/OS, before FIPS can be used, the **FIPS=YES** initialization parameter must be coded at startup (the default is NO).

```

EXPDT = NONE          /* DO NOT PROPAGATE EXPIRATION DATE      */
EXTENDED.RECOVERY = NO /* XRF, EXTENDED RECOVERY FEATURE        */
FIPS = YES           /* Place System SSL in FIPS mode         */
GDGALLOC = GENERATION /* GDG DATASET ALLOCATION BY DSNAMES      */

```

If **FIPS=YES** is not specified and the keyring or key database is not FIPS mode, then coding **Enable FIPS** as **Y** in the Secure+ PARMFILE will be ignored and forced to **N**.

NOTE: System SSL must be in FIPS mode in order to do FIPS. Coding **FIPS =YES** in the initialization parameters is what tells Connect:Direct to indicate to System SSL to run in FIPS mode.

NOTE: **Enable FIPS** can only be coded on the Local node in the Secure+ PARMFILE; it cannot be overridden on the Remote nodes (it is “grayed” out).

The keyring or key database used must be built and defined as FIPS mode if FIPS=YES is specified in the initialization parameters. Also, if FIPS is defined in the Secure+ PARMFILE, FIPS=YES must be coded in the initialization parameters.

If FIPS=YES is coded in the initialization parameters, the keyring or key database is FIPS mode and FIPS is defined in the Secure+ PARMFILE, then at startup you should see something like the following:

```

SITA195I Secure+ SSL/TLS FIPS Mode initialization complete,
SITA195I DSN=HLQ.SECURE.PARMFILE

```

Setting Up Secure+ Connections

The following examples demonstrate the following:

1. **Setting Up a Secure+ SSL/TLS Connection on C:D z/OS**
2. **Setting Up and Using Secure+ SSL/TLS on C:D Windows**

NOTE: C:D Windows is used in the examples; however, C:D Windows and C:D Unix both use the same Secure+ Admin Tool and C:D Windows 4.7 and C:D Unix 4.2 both use **IKEYMAN** in the same way. Therefore, the instructions can be used to configure C:D Unix as well as C:D Windows.

Setting Up a Secure+ SSL/TLS Connection on C:D z/OS

When setting up a node for using SSL or TLS (includes protocols TLS, or TLSv1.0, TLSv1.1 and TLSv1.2), it is important to know if the local node is C:D z/OS 5.2 or later or pre-5.2, as the Secure+ settings and the PARMFILE changed significantly starting with C:D z/OS 5.2.

To use SSL or TLS, certificates must be exchanged between the trading partners. If you need help building certificates, please refer to the section **Building Certificates Using gskkyman**. If you need help importing or exporting the certificates to or from your key database, see the section **Using Unix System Service (USS) gskkyman (Key Database for C:D z/OS)**; if importing or exporting to or from a keyring, see the section **Using Secure+ with z/OS Security Applications** and use the instructions dealing with RACF, CA-ACF2, or CA-Top Secret (whichever applies).

Only one protocol can be used at a time; beginning with C:D z/OS 5.2, C:D Windows 4.7 and C:D Unix 4.2, multiple protocols can be coded and the highest available will be used (i.e. if SSL and TLS are both coded, and the remote coming across is using TLS, TLS will be used and SSL will be ignored).

This example will show a SSL/TLS connection from C:D z/OS 5.2 client (or sender), node SC.ZOS.JMCGE52 to C:D z/OS 5.2 server (or recipient), node SC.ALT.JMCGE52, to show how to set up each side of C:D z/OS 5.2.

In these instructions, the key database (gskkyman) will be used (rather than a keyring) and it is assumed that the key database has already been built and that the certificates have already been exchanged and loaded.

Update the Client (Sender) Node

Go into the UI and go to panel **ADMIN;S** and select option **SA**.

```
SC.ZOS.JMCGE52          Execute Secure Plus Commands          17:10
CMD ==> SA

CR - Execute Certificate Expiration Validation Command
RF - Execute Refresh Secure Plus Environment Command
SA - Execute Secure Plus Admin Tool
```

This will bring up the Secure+ Admin Tool.

```
File Edit Help
-----
Secure+ Admin Tool: Main Screen
Option ==> _____ Scroll CSR

Table Line Commands are:

U Update node          H View History          D Delete node
I Insert node          V View node          CL Clone node

Node Filter : * _____

LC Node Name          Secure+          External Client
Type Protocol Override Encryption Auth Auth
-----
***** BOTTOM OF DATA *****
```

You now need to either open a Secure+ PARMFILE that you already have created or build a new PARMFILE; if a new PARMFILE is needed, please refer to “**Manually Create Connect:Direct Secure+ PARMFILE**”.

Either go to **File** and select option **2. Open** and enter the name of the PARMFILE that will be used, or simply type **FO** on the option line and hit **Enter**, then enter the name of the Secure+ PARMFILE being used:

```
Secure+ Admin Tool: File Selection

Option ---> _____

Enter file name for: INPUT SECURE PARM FILE

File
Name: CD.SEND52.SECURE.PARMFILE Browse
---

File System Type:
1 1. MVS Cancel
---
```

You will see something like the following:

```

File  Edit  Help
-----
SC.ZOS.JMCGE52      Secure+ Admin Tool: Main Screen      Row 1 to 9 of 25
Option ==>          Scroll CSR

                        Table Line Commands are:

U Update node        H View History        D Delete node
I Insert node        V View node          CL Clone node

Node Filter : *

LC Node Name          Secure+                External Client
Type Protocol Override Encryption  Auth  Auth
-----
CDWIN47               R   *                N      *      N
CDWIN48               R   TLSV12           N      *      N
CDWIN60               R   TLSV12           N      *      N
SC.ALT.JMCGE52T       R   Disabled         N      *      N
SC.ZOS.CD51           R   Disabled         N      *      N
SC.ZO
SC.ZO  007: The Secure+ parm file has been read successfully.
SC.ZO
SC.ZOS.JMCGE52T       R   TLSV10           N      *      N

```

Hit **Enter** to remove the **007** message.

The Secure+ PARMFILE should look something like this:

```

File  Edit  Help
-----
SC.ZOS.JMCGE52      Secure+ Admin Tool: Main Screen      Row 1 to 9 of 25
Option ==>          Scroll CSR

                        Table Line Commands are:

U Update node        H View History        D Delete node
I Insert node        V View node          CL Clone node

Node Filter : *

LC Node Name          Secure+                External Client
Type Protocol Override Encryption  Auth  Auth
-----
CDWIN47               R   *                N      *      N
CDWIN48               R   TLSV12           N      *      N
CDWIN60               R   TLSV12           N      *      N
SC.ALT.JMCGE52T       R   Disabled         N      *      N
SC.ZOS.CD51           R   Disabled         N      *      N
SC.ZOS.CD52           R   TLSV12           N      *      N
SC.ZOS.JMCGE52        L   Disabled         Y      Y      N
SC.ZOS.JMCGE60        R   TLSV12           N      *      N
SC.ZOS.JMCGE52T       R   TLSV10           N      *      N

```

Your local node should already be setup with the correct certificate label and pathname (keyring or key database). If so, you can skip to **“Update the Remote Node on Client”**.

Add or Change Client (Site) Certificate on Local Node

Starting with the local node (SC.ZOS.JMCGE52), type **U** on the **LC** line to update the node and hit **Enter**.

```

File  Edit  Help
-----
SC.ZOS.JMCGE52      Secure+ Admin Tool: Main Screen      Row 1 to 9 of 25
Option ==>          Scroll CSR

Table Line Commands are:

U Update node      H View History      D Delete node
I Insert node      V View node      CL Clone node

Node Filter : *

LC Node Name      Secure+      External Client
Type Protocol Override Encryption Auth Auth
-----
CDWIN47           R   *           N           *           *           N
CDWIN48           R   TLSV12      N           *           *           N
CDWIN60           R   TLSV12      N           *           *           N
SC.ALT.JMCGE52T   R   Disabled    N           *           *           N
SC.ZOS.CD51       R   Disabled    N           *           *           N
SC.ZOS.CD52       R   TLSV12      N           *           *           N
U SC.ZOS.JMCGE52   L   Disabled    Y           Y           N           N
SC.ZOS.JMCGE60    R   TLSV12      N           *           *           N
SC.ZOS.JMCGE52T   R   TLSV10      N           *           *           N
  
```

Go to the **SSL/TLS Parameters** panel.

```

Secure+ Create/Update Panel

Option ==>

Node Name: SC.ZOS.JMCGE52      Type: L      (Local or Remote)
-----
| Security Options | EA Parameters | SSL/TLS Parameters |
| --- | -- | --- |
-----
Secure+ Protocol:      Security Mode (Yes , No , Default to Local)
Enable SSL             N      Enable FIPS             N
Enable TLS 1.0         N      Enable SP800-131a Transition N
Enable TLS 1.1         N      Enable SP800-131a Strict   N
Enable TLS 1.2         N      Enable NSA Suite B 128 bit N
                        Enable NSA Suite B 192 bit N

Auth Timeout:         120      Enable Override         Y

Alias Names:          TCP Information:
_____      IPAddr: _____
_____      Port: _____
_____

                        OK      Cancel
                        --      ---
  
```

If this is a new PARMFILE, and it was built using Quick Start (i.e. using your C:D NETMAP), the **Certificate Label** by default will contain the local node as the Label; if it was built manually without using the NETMAP, then **Certificate Label** will contain an asterisk (*).

Place your cursor on **Certificate Label** and hit **Enter**.

```

Secure+ Create/Update Panel
Option ==> _____
Node Name:   SC.ZOS.JMCGE52      Type:  L      (Local or Remote)
-----
| Security Options      | EA Parameters      | SSL/TLS Parameters  |
| ---                  | --                | ---                |
-----
Enable Client Auth      N      (Yes , No , Default to Local)
Enable Data Encrypt     Y
-----
Certificate Label      | SC.ZOS.JMCGE52    |
Cipher Suites          | 0001000200030004000500060009000A002F0035 |
Certificate Pathname    |                    |
Certificate Common Name  |                    |
-----
OK                      Cancel
--                      ---

```

Enter your **Certificate Label**.

Secure+ Admin Tool: "Certificate Label" information

Option ---> _____

Enter the label of the x.509 certificate for use within the box on the next page. The individual lines within the box will be concatenated to form a single string.

Z52.Certificate.Label

When you are finished entering the label, hit **Enter**.

NOTE: Beginning with C:D z/OS 6.0, the use of the **default certificate** is now available. If no label is entered on either the local or remote nodes, the default certificate (if one has been defined) will be used. Care should be taken to make sure the correct certificate is marked as the default in your keyring or key database.

You may notice that this panel may have **<Change Pending>** in the upper right corner; this means you have already made changes to this node and have not yet hit **OK** to save the changes.

If you wish to change the Cipher Suites, place your cursor on **Cipher Suites** and hit **Enter**.

```

Secure+ Create/Update Panel
Option ==>
Node Name: SC.ZOS.JMCGE52 Type: L (Local or Remote)
-----
| Security Options | EA Parameters | SSL/TLS Parameters |
| --- | -- | --- |
-----
Enable Client Auth N (Yes , No , Default to Local)
Enable Data Encrypt Y
-----
Certificate Label | Z52.Certificate.Label |
Cipher Suites | 0001000200030004000500060009000A002F0035 |
Certificate Pathname | |
Certificate Common Name | |
-----
OK Cancel
-- --

```

You can code up to 10 in any order; the ciphers will be used in the order they are numbered.

NOTE: You cannot code DEFAULT_TO_LOCAL_NODE on the Local Node.

```

Option --->
Update the order field below to enable and order cipher suites.

O All Available Cipher-Suites Enabled Cipher-Suites
== =====
1 TLS_RSA_WITH_AES_256_CBC_SHA TLS_RSA_WITH_AES_256_CBC_SHA
2 TLS_RSA_WITH_AES_128_CBC_SHA TLS_RSA_WITH_AES_128_CBC_SHA
3 TLS_RSA_WITH_3DES_EDE_CBC_SHA TLS_RSA_WITH_3DES_EDE_CBC_SHA
4 TLS_RSA_WITH_DES_CBC_SHA TLS_RSA_WITH_DES_CBC_SHA
-- TLS_RSA_WITH_NULL_MD5 *DEPRECATE
-- TLS_RSA_WITH_NULL_SHA *DEPRECATE
-- SA_EXPORT_WITH_RC4_40_MD5 *DEPRECATE
-- TLS_RSA_WITH_RC4_128_MD5 *DEPRECATE
-- TLS_RSA_WITH_RC4_128_SHA *DEPRECATE
-- EXPORT_WITH_RC2_CBC_40_MD5 *DEPRECATE
-- TLS_RSA_WITH_NULL_SHA256
-- TLS_RSA_WITH_AES_128_CBC_SHA256
-- TLS_RSA_WITH_AES_256_CBC_SHA256
-- TLS_RSA_WITH_AES_128_GCM_SHA256
-- TLS_RSA_WITH_AES_256_GCM_SHA384
-- TLS_ECDHE_ECDSA_WITH_NULL_SHA
More: +

```

NOTE: Due to z/OS APAR OA47405 (z/OS 1.13 and 2.1), ciphers 0001-0006 are no longer available.

The ciphers that are no longer available and are marked as ***DEPRECATE** are:

```

TLS_RSA_WITH_NULL_MD5
TLS_RSA_WITH_NULL_SHA
TLS_RSA_EXPORT_WITH_RC4_40_MD5
TLS_RSA_WITH_RC4_128_MD5
TLS_RSA_WITH_RC4_128_SHA
TLS_RSA_EXPORT_WITH_RC2_CBC_40_MD5

```

Hit **F3** to exit panel.

Now place your cursor on **Certificate Pathname** and hit **Enter**.

```

Secure+ Create/Update Panel
Option ==> _____
Node Name:  SC.ZOS.JMCGE52      Type:  L      (Local or Remote)
-----
| Security Options      | EA Parameters      | SSL/TLS Parameters  |
| ---                  | --                 | ---                 |
-----
Enable Client Auth      N      (Yes , No , Default to Local)
Enable Data Encrypt     Y
-----
Certificate Label       | Z52.Certificate.Label |
Cipher Suites          | 0035002F000A0009     |
Certificate Pathname    |                        |
Certificate Common Name |                        |
-----
OK                      Cancel
--                      ---

```

Here you enter either your keyring or your key database that is being used by this instance of Connect:Direct.

NOTE: Connect:Direct can support the use of a single keyring or a single key database, but not both.

If you want to enter a keyring, you must put only the name of the keyring without a path:

```

Secure+ Admin Tool: "Certificate Path" information
Option ---> _____
More: +
This is a scrollable panel. Please page down for
more information and modifiable fields.

In the first box below, enter the Unix path name of the .KDB file containing
the x.509 certificates for use by Connect:Direct.

In the second box below, enter the password used when creating the .KDB file.

-----
| KEYRING _____ |
| _____ |
| _____ |
| _____ |
| _____ |
| _____ |
| _____ |
| _____ |
| Certificate _____ |
| Path: _____ |
| _____ |
| _____ |

```

At this point, you are done; hit **Enter** or **F3** to exit this panel.

However, if you want to enter a key database, you will need to code the path as well as the key database name and include the **Certificate Pass Phrase** for that key database:

```
Secure+ Admin Tool: "Certificate Path" information
Option ---> _____ More: +
    This is a scrollable panel. Please page down for
        more information and modifiable fields.

In the first box below, enter the Unix path name of the .KDB file containing
the x.509 certificates for use by Connect:Direct.

In the second box below, enter the password used when creating the .KDB file.
```

| /u/userid/z52certs.kdb |
| _____ |
| _____ |
| _____ |
| _____ |
| _____ |
| _____ |
Certificate |
Path: |
| _____ |

However, before hitting **Enter**, use **F8** to scroll down and enter your **Certificate Pass Phrase**:

<u>Certificate</u> <u>Path:</u>	<div></div>
<u>Certificate</u> <u>Pass Phrase:</u>	<u>Z52PassPhrase</u> <div></div>

Hit **Enter** or **F3** to exit panel.

The **SSL/TLS Parameters** panel should now have either the keyring or the key database like the following:

```

Secure+ Create/Update Panel
Option ==>
Node Name: SC.ZOS.JMCGE52 Type: L (Local or Remote)
-----
| Security Options | EA Parameters | SSL/TLS Parameters |
| --- | -- | --- |
-----
Enable Client Auth N (Yes , No , Default to Local)
Enable Data Encrypt Y
-----
Certificate Label | Z52.Certificate.Label |
Cipher Suites | 0035002F000A0009 |
Certificate Pathname | /u/userid/Z52certs.kdb |
Certificate Common Name |
-----
OK Cancel
--

```

If Client Authentication is being used, you may also need to enter a **Certificate Common Name** used to add an extra layer of security when authenticating the client certificate. **Certificate Common Name** is optional, but if it coded, it MUST match what is coded on the trading partner's node definition in their PARMFILE.

To code the **Certificate Common Name**, place the cursor on **Certificate Common Name** and hit **Enter**.

Secure+ Create/Update Panel		<Change Pending>
Option ==> _____		
Node Name:	<u>SC.ZOS.JMCGE52</u>	Type: <u>L</u> (Local or Remote)

Security Options	EA Parameters	SSL/TLS Parameters
---	--	---

Enable Client Auth	<u>N</u>	(Yes , No , Default to Local)
Enable Data Encrypt	<u>Y</u>	

Certificate Label	Z52.Certificate.Label	
Cipher Suites	0035002F000A0009	
Certificate Pathname	/u/userid/Z52certs.kdb	
Certificate Common Name		

		OK Cancel
		-- ---

Enter the **Certificate Common Name** that you and your trading partner have agreed upon:

Secure+ Admin Tool: "Certificate Common Name" information	
Option ---> _____	
This is a scrollable panel. Please page down for more <u>information</u> and <u>modifiable</u> fields.	
Enter the Certificate Common Name validation string in the Field below and press the "End" key to save.	

CD52.Cert.Common.Name	

When finished, either hit **Enter** or **F3**. The **SSL/TLS Parameters** panel should now have the **Certificate Common Name** added. Place your cursor on **OK** and hit **Enter**.

Secure+ Create/Update Panel		<Change Pending>
Option ==> _____		
Node Name:	<u>SC.ZOS.JMCGE52</u>	Type: <u>L</u> (Local or Remote)

Security Options	EA Parameters	SSL/TLS Parameters
---	--	---

Enable Client Auth	<u>N</u>	(Yes , No , Default to Local)
Enable Data Encrypt	<u>Y</u>	

Certificate Label	Z52.Certificate.Label	
Cipher Suites	0035002F000A0009	
Certificate Pathname	/u/userid/Z52Certs.kdb	
Certificate Common Name	CD52.Cert.Common.Name	

		OK Cancel
		-- ---

Go back to the **Security Options** panel.

Secure+ Create/Update Panel			
Option ==>			
Node Name: <u>SC.ZOS.JMCGE52</u>		Type: <u>L</u>	(Local or Remote)

Security Options	EA Parameters	SSL/TLS Parameters	
---	--	---	

Secure+ Protocol:		Security Mode (Yes , No , Default to Local)	
Enable SSL	<u>N</u>	Enable FIPS	<u>N</u>
Enable TLS 1.0	<u>N</u>	Enable SP800-131a Transition	<u>N</u>
Enable TLS 1.1	<u>N</u>	Enable SP800-131a Strict	<u>N</u>
Enable TLS 1.2	<u>N</u>	Enable NSA Suite B 128 bit	<u>N</u>
		Enable NSA Suite B 192 bit	<u>N</u>
Auth Timeout:	<u>120</u>	Enable Override	<u>Y</u>
Alias Names:		TCP Information:	
_____		IPaddr: _____	
_____		Port: _____	

		OK	Cancel
		--	---

If you want to enable any Secure+ protocols on the Local node, code them here. Also make sure that **Enable Override** is set to **Y** if any of the remote nodes are going to be using any Secure+ protocol that is different from the Local node. Since the way the Local node is set (whether Secure+ disabled or one or more protocols are defined) is basically the “default” for the entire system, it is typically recommended that the Local node be set to have Secure+ disabled (that is, all protocols set to **N**) with **Enable Override** set to **Y**.

NOTE: To disable Secure+ on the entire Connect:Direct system, set all protocols to **N** and **Enable Override** to **N** on the Local Node, hit **OK**, then do a **Save Active** on this PARMFILE.

Also, unlike previous versions of Connect:Direct, multiple protocols can be enabled with the highest available being used. If SSL, TLS 1.0 (this is the same as TLS on previous versions of Connect:Direct), TLS 1.1 and TLS 1.2 are all enabled, then a remote node coming in with any of these protocols enabled will connect with the protocol defined by the remote. In this case, the highest protocol available to both sides will be used.

Place your cursor on **OK** and hit **Enter**.

You can now add changes to the remote node to enable SSL or TLS.

Update the Remote Node on Client

Update the remote node (SC.ALT.JMCGE52), type **U** on the **LC** line to update the node and hit **Enter**.

```

File  Edit  Help
-----
SC.ZOS.JMCGE52      Secure+ Admin Tool: Main Screen      Row 1 to 9 of 25
Option ==>>        Scroll CSR

                        Table Line Commands are:

U Update node        H View History        D Delete node
I Insert node        V View node          CL Clone node

Node Filter : *

LC Node Name        Secure+
                    Type Protocol Override Encryption External Client
                    Type Protocol Override Encryption Auth Auth
-----
--- CDWIN47          R * N * * N
--- CDWIN48          R TLSV12 N * * N
--- CDWIN60          R TLSV12 N * * N
U SC.ALT.JMCGE52    R Disabled N * * N
--- SC.ZOS.CD51      R Disabled N * * N
--- SC.ZOS.CD52      R TLSV12 N * * N
--- SC.ZOS.JMCGE52   L Disabled Y Y N N
--- SC.ZOS.JMCGE60   R TLSV12 N * * N
--- SC.ZOS.JMCGE52T  R TLSV10 N * * N

```

Code **Y** on whatever protocol you wish to use; multiple protocols can be coded, and the highest available will be used. For example, if you code all protocols on your remote, but your trading partner only has SSL and TLS 1.0 coded, then TLS 1.0 will be used. When the desired protocol(s) is set, go to the **SSL/TLS Parameters** panel.

```

Secure+ Create/Update Panel

Option ==>>

Node Name: SC.ALT.JMCGE52      Type: R      (Local or Remote)
-----
| Security Options | EA Parameters | SSL/TLS Parameters |
| --- | -- | --- |
-----
Secure+ Protocol:
Enable SSL N
Enable TLS 1.0 N
Enable TLS 1.1 N
Enable TLS 1.2 Y

Security Mode (Yes , No , Default to Local)
Enable FIPS D
Enable SP800-131a Transition D
Enable SP800-131a Strict D
Enable NSA Suite B 128 bit D
Enable NSA Suite B 192 bit D

Auth Timeout: 120      Enable Override N

Alias Names:
____
____
____

TCP Information:
IPAddr: _____
Port: _____

OK      Cancel
--      ---

```

NOTE: **Enable Override** on the remote node allows some Secure+ parameters in the process to override some parameters coded here. Unless needed, it is typically recommended to set this to **N** on the remote.

The **SSL/TLS Parameters** panel should look like the following:

Secure+ <u>Create/Update</u> Panel			<Change Pending>
Option ==>			
Node Name:	SC.ALT.JMCGE52	Type: R	(Local or Remote)

Security Options	EA Parameters	SSL/TLS Parameters	
---	--	---	

Enable Client Auth	N	(Yes , No , Default to Local)	
Enable Data Encrypt	Y		

Certificate Label	*		
Cipher Suites	FFFF		
Certificate Pathname	*		
Certificate Common Name			

		OK	Cancel
		--	---

NOTE: If changes were made on the previous panel, **<Change Pending>** will be in the upper right, as in this example. This simply means changes have been made since the last time **OK** was hit.

The current settings for the Certificate Information are normally fine – the * means it will default to the Local Node setting. However, if an Alternate Site (Local) Certificate is needed to be used for this remote node, enter the Label for that certificate on **Certificate Label**.

NOTE: Do **NOT** enter the label from the remote certificate here; **Certificate Label** is ONLY to be used for a local certificate and **MUST** have an accompanying private key.

Also, **beginning in C:D z/OS 6.0**, if no label is entered either here or the local node, the default certificate in the keyring (if defined) will be used.

The **FFFF** (or **FF** if pre-z/OS 2.1) on the **Cipher Suites** means it will default to the ciphers coded on the local node. If a different cipher setting is needed for this remote node, enter it here.

Certificate Pathname cannot be altered on the remote node; this can only be set on the local node.

You may also want to set either the **Enable Client Auth** or **Enable Data Encrypt** flags if needed.

If Client Authentication is being used, you may also need to enter a **Certificate Common Name** (if this is coded on the local node, it will not show up here). If a Common Name is needed for this remote node, code it here.

Once everything is set correctly, select **OK** and hit **Enter**.

You now need to save your PARMFILE; if this is the active PARMFILE, do a **Save Active**. Once saved, get out of the Secure+ Admin tool and go to panel **ADMIN;S** and select option **RF** to refresh the Secure+ environment (changing the local node and doing a **Save As** should automatically refresh the Secure+ environment, but it is always a good idea to make sure).

If you are creating a new PARMFILE, do a **Save As** and specify a different PARMFILE name. You then need to add the new PARMFILE to the **SECURE.DSN** parameter in your INITPARMs and recycle Connect:Direct. Assuming the correct changes have been made to the remote node, you should now have SSL or TLS.

Now you will need to go to the Server node (on your trading partner's system) and edit that PARMFILE.

Update the Server (Receiver) Node in Server PARMFILE

Go into the UI and go to panel **ADMIN;S** and select option **SA**.

```
SC.ZOS.JMCGE52          Execute Secure Plus Commands          17:30
CMD ==> SA

CR - Execute Certificate Expiration Validation Command
RF - Execute Refresh Secure Plus Environment Command
SA - Execute Secure Plus Admin Tool
```

This will bring up the Secure+ Admin Tool.

```
File Edit Help
-----
Secure+ Admin Tool: Main Screen
Option ==> _____ Scroll CSR

Table Line Commands are:

U Update node          H View History          D Delete node
I Insert node          V View node          CL Clone node

Node Filter : * _____

LC Node Name          Secure+          External Client
Type Protocol Override Encryption Auth Auth
-----
***** BOTTOM OF DATA *****
```

You now need to either open a Secure+ PARMFILE that you already have created or build a new PARMFILE; if a new PARMFILE is needed, please refer to **“Manually Create Connect:Direct Secure+ PARMFILE”**.

Either go to **File** and select option **2. Open** and enter the name of the PARMFILE that will be used, or simply type **FO** on the option line and hit **Enter**, then enter the name of the Secure+ PARMFILE being used:

```
Secure+ Admin Tool: File Selection

Option ---> _____

Enter file name for: INPUT SECURE PARM FILE

File
Name: CD.RECV52.SECURE.PARMFILE Browse
---

File System Type:
1 1. MVS Cancel
---
```

You will see something like the following:

File Edit Help

SC.ALT.JMCGE52

Secure+ Admin Tool: Main Screen

Row 1 to 9 of 15

Option ==> _____

Scroll CSR

Table Line Commands are:

U Update node

H View History

D Delete node

I Insert node

V View node

CL Clone node

Node Filter : * _____

LC	Node Name	Type	Secure+ Protocol	Override	Encryption	External Auth	Client Auth

___	CDWIN48	R	*	N	*	*	N
___	CDWIN60	R	*	N	*	*	N
___	SC.ALT.JMCGE52	L	Disabled	Y	Y	N	N
___	SC.ALT.JMCGE52T	R	Disabled	Y	*	*	N
___	SC.ZOS.CD51	R	*	N	*	*	N
___	SC.ZOS.CD52	R	*	N	*	*	N
___	SC.ZOS.JMCGE52	R	Disabled	Y	*	*	N
___	SC.ZOS.JMCGE60	R	TLSV12	Y	*	*	N

If you initially see a **007** message (“The Secure+ parm file has been read successfully”), hit **Enter** to remove it.

The local node should already be setup with the correct certificate label and pathname (keyring or key database). If so, you can skip to “**Update the Remote Node on Server (C:D 5.2)**”.

Add or Change Server Certificate on Local Node

Starting with the local node (SC.ALT.JMCGE52), type **U** on the **LC** line to update the node and hit **Enter**.

```

File  Edit  Help
-----
SC.ALT.JMCGE52          Secure+ Admin Tool: Main Screen          Row 1 to 9 of 15
Option ==> _____ Scroll CSR

                        Table Line Commands are:

U Update node           H View History           D Delete node
I Insert node           V View node           CL Clone node

Node Filter : * _____

LC Node Name           Type  Secure+ Protocol  Override  Encryption  External Client Auth
-----
CDWIN48                R    *              N          *          *          N
CDWIN60                R    *              N          *          *          N
U SC.ALT.JMCGE52        L    Disabled     Y          Y          N          N
SC.ALT.JMCGE52T        R    Disabled     Y          *          *          N
SC.ZOS.CD51            R    *              N          *          *          N
SC.ZOS.CD52            R    *              N          *          *          N
SC.ZOS.JMCGE52         R    Disabled     Y          *          *          N
SC.ZOS.JMCGE60         R    TLSV12        Y          *          *          N

```


Go to the **SSL/TLS Parameters** panel.

```

Secure+ Create/Update Panel
Option ==>

Node Name:  SC.ALT.JMCGE52      Type:  L      (Local or Remote)
-----
| Security Options | EA Parameters | SSL/TLS Parameters |
| ---            | --          | ---                |
-----
Secure+ Protocol:      Security Mode (Yes , No , Default to Local)
  Enable SSL           N      Enable FIPS           N
  Enable TLS 1.0       N      Enable SP800-131a Transition N
  Enable TLS 1.1       N      Enable SP800-131a Strict   N
  Enable TLS 1.2       N      Enable NSA Suite B 128 bit N
                           Enable NSA Suite B 192 bit N

Auth Timeout:         120      Enable Override       Y

Alias Names:          _____
                    _____
                    _____

                    TCP Information:
                    IPaddr: _____
                    Port:   _____

                                   OK      Cancel
                                   --      ---

```

If this is a new PARMFILE, notice that the **Certificate Label**, by default, has your local node as the Label.

Place your cursor on **Certificate Label** and hit **Enter**.

```

Secure+ Create/Update Panel
Option ==> _____
Node Name:  SC.ALT.JMCGE52      Type:  L      (Local or Remote)
-----
| Security Options      | EA Parameters      | SSL/TLS Parameters  |
| ---                  | --                | ---                |
-----
Enable Client Auth      N      (Yes , No , Default to Local)
Enable Data Encrypt     Y
-----
Certificate Label       | SC.ALT.JMCGE52    |
Cipher Suites          | 0001000200030004000500060009000A002F0035 |
Certificate Pathname    |                    |
Certificate Common Name  |                    |
-----
OK                      Cancel
--                      ---

```

Enter your **Certificate Label**.

NOTE: Beginning in C/D z/OS 6.0, if this field is blank, the default certificate in your keyring will be used.

```
Secure+ Admin Tool: "Certificate Label" information
Option ---> _____

Enter the label of the x.509 certificate for use within the box on the next
page. The individual lines within the box will be concatenated to form
a single string.

-----
| Z52.Alt.Certificate.Label |
| _____ |
| _____ |
| _____ |
| _____ |
-----
```

When you are finished entering the label, hit **Enter**.

If you wish to change the Cipher Suites, place your cursor on **Cipher Suites** and hit **Enter**.

```

Secure+ Create/Update Panel                                <Change Pending>
Option ==> _____
Node Name:  SC.ALT.JMCGE52      Type:  L      (Local or Remote)
-----
| Security Options | EA Parameters | SSL/TLS Parameters |
| ---            | --          | ---                |
-----
Enable Client Auth      N      (Yes , No , Default to Local)
Enable Data Encrypt     Y
-----
Certificate Label       | Z52.ALT.Certificate.Label |
Cipher Suites           | 0001000200030004000500060009000A002F0035 |
Certificate Pathname    |                               |
Certificate Common Name |                               |
-----
OK      Cancel
--      ---

```

You can code up to 10 in any order; the ciphers will be used in the order they are numbered.

NOTE: You cannot code DEFAULT_TO_LOCAL_NODE on the Local Node.

```

Option ---> _____
Update the order field below to enable and order cipher suites.

O  All Available Cipher-Suites      Enabled Cipher-Suites
==  =====
1  TLS_RSA_WITH_AES_256_CBC_SHA      TLS_RSA_WITH_AES_256_CBC_SHA
2  TLS_RSA_WITH_AES_128_CBC_SHA      TLS_RSA_WITH_AES_128_CBC_SHA
3  TLS_RSA_WITH_3DES_EDE_CBC_SHA      TLS_RSA_WITH_3DES_EDE_CBC_SHA
4  TLS_RSA_WITH_DES_CBC_SHA          TLS_RSA_WITH_DES_CBC_SHA
   TLS_RSA_WITH_NULL_MD5              *DEPRECATE
   TLS_RSA_WITH_NULL_SHA              *DEPRECATE
   SA_EXPORT_WITH_RC4_40_MD5          *DEPRECATE
   TLS_RSA_WITH_RC4_128_MD5          *DEPRECATE
   TLS_RSA_WITH_RC4_128_SHA          *DEPRECATE
   EXPORT_WITH_RC2_CBC_40_MD5        *DEPRECATE
   TLS_RSA_WITH_NULL_SHA256
   TLS_RSA_WITH_AES_128_CBC_SHA256
   TLS_RSA_WITH_AES_256_CBC_SHA256
   TLS_RSA_WITH_AES_128_GCM_SHA256
   TLS_RSA_WITH_AES_256_GCM_SHA384
   TLS_ECDHE_ECDSA_WITH_NULL_SHA

```

NOTE: Due to z/OS APAR OA47405 (1.13 and 2.1), ciphers 0001-0006 are no longer available.

The ciphers that are no longer available and are marked as ***DEPRECATE** are:

```

TLS_RSA_WITH_NULL_MD5
TLS_RSA_WITH_NULL_SHA
TLS_RSA_EXPORT_WITH_RC4_40_MD5
TLS_RSA_WITH_RC4_128_MD5
TLS_RSA_WITH_RC4_128_SHA
TLS_RSA_EXPORT_WITH_RC2_CBC_40_MD5

```

Hit **F3** to exit panel.

However, before hitting **Enter**, use **F8** to scroll down and enter your **Certificate Pass Phrase**:

[illegible]

Hit **Enter** or **F3** to exit panel.

The **SSL/TLS Parameters** panel should now have either the keyring or the key database like the following:

```
Secure+ Create/Update Panel                                <Change Pending>
```

Option ==> _____

Node Name:	<u>SC.ALT.JMCGE52</u>	Type:	<u>L</u>	(Local or Remote)

Security Options	EA Parameters	SSL/TLS Parameters		
---	--	---		

Enable Client Auth	<u>N</u>	(Yes , No , Default to Local)
Enable Data Encrypt	<u>Y</u>	

Certificate Label	Z52.Certificate.Label	
Cipher Suites	0035002F000A0009	
Certificate Pathname	/u/userid/Z52altcerts.kdb	
Certificate Common Name		

OK	Cancel
--	----

If Client Authentication is being used, you may also need to enter a **Certificate Common Name** used to add an extra layer of security when authenticating the client certificate. **Certificate Common Name** is optional, but if it coded, it MUST match what is coded on the trading partner's node definition in their PARMFILE.

To code the **Certificate Common Name**, place the cursor on **Certificate Common Name** and hit **Enter**.

```

Secure+ Create/Update Panel
Option ==>
Node Name: SC.ALT.JMCGE52 Type: L (Local or Remote)
-----
| Security Options | EA Parameters | SSL/TLS Parameters |
| --- | -- | --- |
-----
Enable Client Auth N (Yes , No , Default to Local)
Enable Data Encrypt Y
-----
Certificate Label | Z52.Certificate.Label |
Cipher Suites | 0035002F000A0009 |
Certificate Pathname | /u/userid/Z52altcerts.kdb |
Certificate Common Name |
-----
OK Cancel
--

```

Enter the Common Name that you and your trading partner have agreed upon:

```
Secure+ Admin Tool: "Certificate Common Name" information
Option ---> _____

This is a scrollable panel. Please page down for
more information and modifiable fields.

Enter the Certificate Common Name validation string in the
Field below and press the "End" key to save.
```

CD52.Cert.Common.Name

When finished, either hit **Enter** or **F3**. The **SSL/TLS Parameters** panel should now have the **Certificate Common Name** added. Place your cursor on **OK** and hit **Enter**.

```

Secure+ Create/Update Panel
Option ==>
Node Name: SC.ALT.JMCGE52 Type: L (Local or Remote)
-----
| Security Options | EA Parameters | SSL/TLS Parameters |
| --- | -- | --- |
-----
Enable Client Auth N (Yes , No , Default to Local)
Enable Data Encrypt Y
-----
Certificate Label | Z52.Certificate.Label |
Cipher Suites | 0035002F000A0009 |
Certificate Pathname | /u/userid/Z52altcerts.kdb |
Certificate Common Name | CD52.Cert.Common.Name |
-----
OK Cancel
-- --

```

Go back to the **Security Options** panel.

```

Secure+ Create/Update Panel
Option ==>

Node Name: SC.ALT.JMCGE52      Type: L      (Local or Remote)
-----
| Security Options | EA Parameters | SSL/TLS Parameters |
| ---            | --          | ---                |
-----
Secure+ Protocol:      Security Mode (Yes , No , Default to Local)
  Enable SSL           N      Enable FIPS           N
  Enable TLS 1.0       N      Enable SP800-131a Transition N
  Enable TLS 1.1       N      Enable SP800-131a Strict   N
  Enable TLS 1.2       N      Enable NSA Suite B 128 bit N
                           Enable NSA Suite B 192 bit N

Auth Timeout:         120      Enable Override       Y

Alias Names:          TCP Information:
_____             IPaddr: _____
_____             Port:      _____
_____

                                OK      Cancel
                                --      ---

```

Place your cursor on **OK** and hit **Enter**. You can now add changes to the remote node to enable SSL or TLS.

Update the Remote Node on Server

Update the remote node (SC.ZOS.JMCGE52), type **U** on the **LC** line to update the node and hit **Enter**.

```

File  Edit  Help
-----
SC.ALT.JMCGE52      Secure+ Admin Tool: Main Screen      Row 1 to 9 of 15
Option ==>> _____ Scroll CSR

Table Line Commands are:

U Update node      H View History      D Delete node
I Insert node      V View node       CL Clone node

Node Filter : * _____

LC Node Name      Secure+      External Client
Type Protocol Override Encryption Auth Auth
-----
CDWIN48           R *          N *          * N
CDWIN60           R *          N *          * N
SC.ALT.JMCGE52    L Disabled  Y Y          N N
SC.ALT.JMCGE52T   R Disabled  Y *          * N
SC.ZOS.CD51       R *          N *          * N
SC.ZOS.CD52       R *          N *          * N
U SC.ZOS.JMCGE52  R Disabled  Y *          * N
SC.ZOS.JMCGE60    R TLSV12     Y *          * N
  
```

Code **Y** on whatever protocol you wish to use; multiple protocols can be coded, and the highest available will be used. For example, if you code all protocols on your remote, but your trading partner only has SSL and TLS 1.0 coded, then TLS 1.0 will be used. When the desired protocol(s) is set, go to the **SSL/TLS Parameters** panel.

```

Secure+ Create/Update Panel

Option ==>>

Node Name: SC.ZOS.JMCGE52      Type: R      (Local or Remote)
-----
| Security Options | EA Parameters | SSL/TLS Parameters |
| --- | --- | --- |
-----
Secure+ Protocol:
Enable SSL      N
Enable TLS 1.0  Y
Enable TLS 1.1  Y
Enable TLS 1.2  Y

Security Mode (Yes , No , Default to Local)
Enable FIPS      D
Enable SP800-131a Transition N
Enable SP800-131a Strict   N
Enable NSA Suite B 128 bit N
Enable NSA Suite B 192 bit N

Auth Timeout: 120      Enable Override      N

Alias Names:
____
____
____

TCP Information:
IPaddr: _____
Port: _____

OK      Cancel
--      ---
  
```

NOTE: **Enable Override** on the remote node allows some Secure+ parameters in the process to override some parameters coded here. Unless needed, it is typically recommended to set this to **N** on the remote.

The **SSL/TLS Parameters** panel should look like the following:

Secure+ <u>Create/Update</u> Panel			<Change Pending>
Option ==>			
Node Name:	SC.ZOS.JMCGE52	Type: R	(Local or Remote)

Security Options	EA Parameters	SSL/TLS Parameters	
---	--	---	

Enable Client Auth	N	(Yes , No , Default to Local)	
Enable Data Encrypt	Y		
Certificate Label	*		
Cipher Suites	FFFF		
Certificate Pathname	*		
Certificate Common Name			
		OK	Cancel
		--	---

NOTE: If changes were made on the previous panel, **<Change Pending>** will be in the upper right, as in this example. This simply means changes have been made since the last time **OK** was hit.

The current settings for the Certificate Information are normally fine – the * means it will default to the Local Node setting. However, if an Alternate Site (Local) Certificate is needed to be used for this remote node, enter the Label for that certificate on **Certificate Label**.

NOTE: Do **NOT** enter the label from the remote certificate here; **Certificate Label** is ONLY to be used for a local certificate and **MUST** have an accompanying private key.

Also, **beginning in C:D z/OS 6.0**, if no label is entered either here or the local node, the default certificate in the keyring (if defined) will be used.

The **FFFF** (or **FF** if on pre-z/OS 2.1) on the **Cipher Suites** means it will default to the ciphers coded on the local node. If a different cipher setting is needed for this remote node, enter it here.

Certificate Pathname cannot be altered on the remote node; this can only be set on the local node.

You may also want to set either the **Enable Client Auth** or **Enable Data Encrypt** flags if needed.

If Client Authentication is being used, you may also need to enter a **Certificate Common Name** (if this is coded on the local node, it will not show up here). If a Common Name is needed for this remote node, code it here.

Once everything is set correctly, select **OK** and hit **Enter**.

You now need to save your PARMFILE; if this is the active PARMFILE, do a **Save Active**. Once saved, get out of the Secure+ Admin tool and go to panel **ADMIN;S** and select option **RF** to refresh the Secure+ environment (changing the local node and doing a **Save As** should automatically refresh the Secure+ environment, but it is always a good idea to make sure).

NOTE: If you are editing multiple nodes, it is recommended changing only 3-4 at a time before doing a **Save**.

If you are creating a new PARMFILE, do a **Save As** and specify a different PARMFILE name. You then need to add the new PARMFILE to the **SECURE.DSN** parameter in your INITPARMs and recycle Connect:Direct.

Assuming the correct changes have been made to the remote node, your SSL or TLS connection should work.

Using Secure+ on C:D Windows / C:D Unix

Secure+ is configured and setup differently on C:D Windows and C:D Unix than it is on C:D z/OS, and the interface for the Secure+ Admin Tool is considerably different.

NOTE: The Secure+ Admin Tool used for C:D Windows is the same as that used for C:D Unix; while the two platforms may be different, the setup for Secure+ is basically the same. However, if you need to use the Command Line Interface, please refer to section **Using IKEYMAN From the Command Line Interface (CLI)**.

It is assumed in these instructions that C:D Windows 4.7 or C:D Unix 4.2 (or higher) with Secure+ has already been installed. If not, whichever you are wanting to use will need to be downloaded and installed with Secure+ prior to continuing with these instructions.

In these instructions, the example using C:D Windows 4.7 will be used.

Before the SSL/TLS connection can be established, you will need to export the certificate from the z/OS node and copy it over to Windows. Please refer to section **Copying z/OS Certificates from gskkyman** if you need assistance.

You will also have to copy the C:D Windows certificate to the C:D z/OS. You can either copy it directly from C:D Windows to HFS, using a process like the following:

```
HFSCOPYZ PROCESS PNODE=CDWin_node SNODE=CDZ_node
STEP01 COPY FROM (PNODE FILE='C:\Certs\CDWin47_cert.txt'
                  DISP=SHR)
TO (SNODE FILE='/u/userid/cdwin47.crt'
   DATATYPE=TEXT PERMISS=777 DISP=RPL)
```

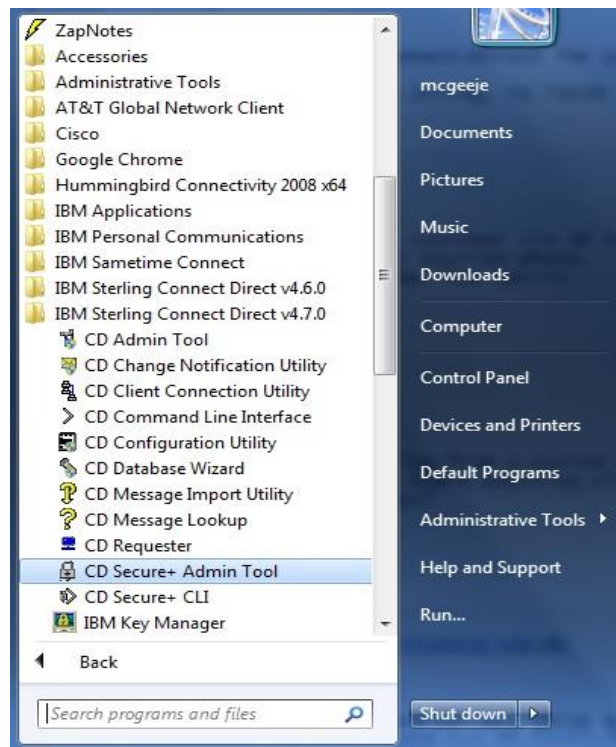
Or, if for some reason this will not work (depending on the systems involved it may not), copy the certificate file from C:D Windows to C:D z/OS, then copy that mainframe file to HFS, like the following:

```
ZOSCOPY1 PROCESS PNODE=CDWin_node SNODE=CDZ_node
STEP01 COPY FROM (PNODE FILE='C:\Certs\CDWin47_cert.txt')
TO (SNODE FILE=hlq.CDWIN47.CERT
   DCB=(RECFM=FB,LRECL=80,BLKSIZE=27920)
   DISP=NEW)

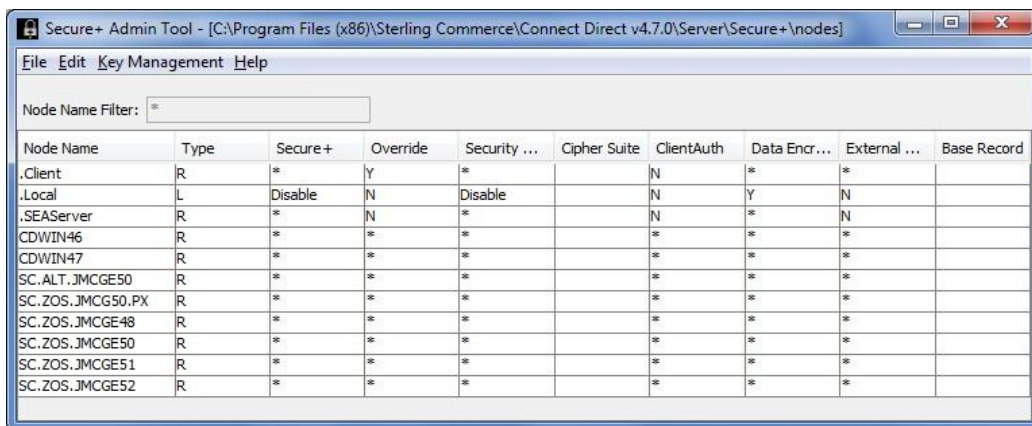
HFSCOPY2 PROCESS PNODE=CDZ_node SNODE=CDZ_node
STEP01 COPY FROM (PNODE FILE=hlq.CDWIN47.CERT DISP=SHR) -
TO (SNODE FILE='/u/userid/cdwin47.CRT' -
   DATATYPE=TEXT PERMISS=777 DISP=RPL)
```

NOTE: As with C:D z/OS 5.2, beginning with C:D Windows 4.7 and C:D Unix 4.2, STS is no longer supported as a protocol. Also, the way that certificates are stored and managed in C:D Windows 4.7 and C:D Unix 4.2 has changed, more resembling a key database as used by the mainframe in gskkyman. That is, C:D Windows and C:D Unix now use IKEYMAN to utilize a KEYSTORE (i.e. key database) for storing and maintaining certificates.

To start, go to Start and open All Programs >> IBM Sterling Connect Direct v4.7.0 >> CD Secure+ Admin Tool:



When this eventually opens, you will see something like the following:



At this point, you will need to set up the Secure+ SSL/TLS connection for your **.local** and remote nodes in C:\D Windows 4.7.

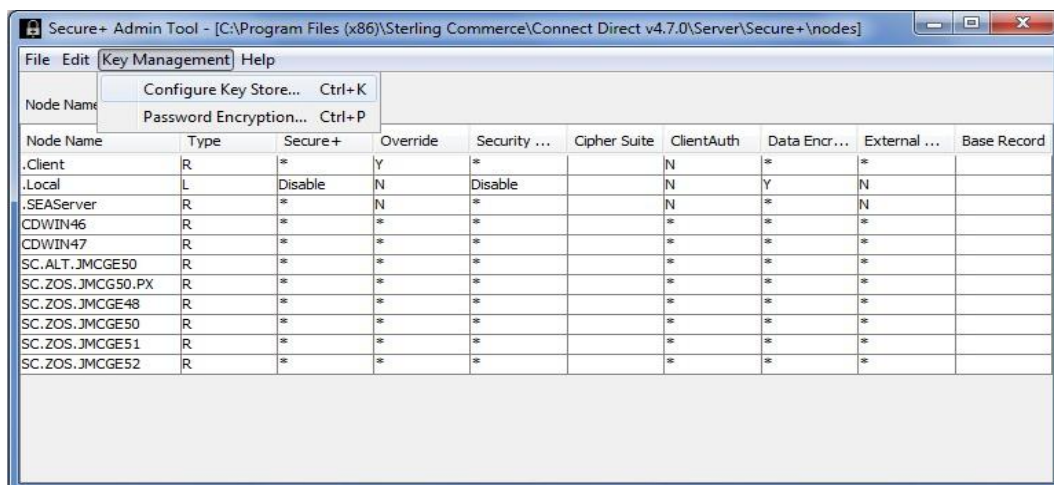
Setting Up a Secure+ SSL/TLS Connection for C:D Windows / C:D Unix

If you have not already set up the **CMS KeyStore**, you will need to do that before continuing. You can use the **CMS KeyStore** already shipped (cdkeystore.kdb) or create a new one. If you choose to use the **CMS KeyStore** already shipped, you can skip to section **Import Certificate into CMS KeyStore**.

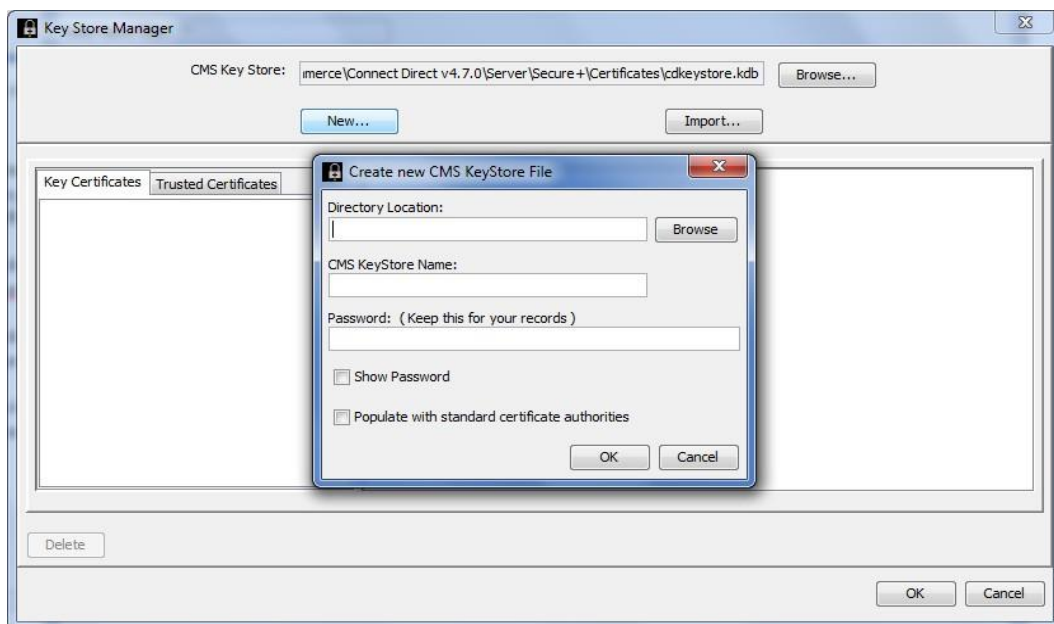
NOTE: When C:D Windows was originally installed and IKEYMAN was installed, a password had to be created; it is this password that will now be needed for the **CMS KeyStore** that was already shipped (cdkeystore.kdb).

Create New CMS KeyStore

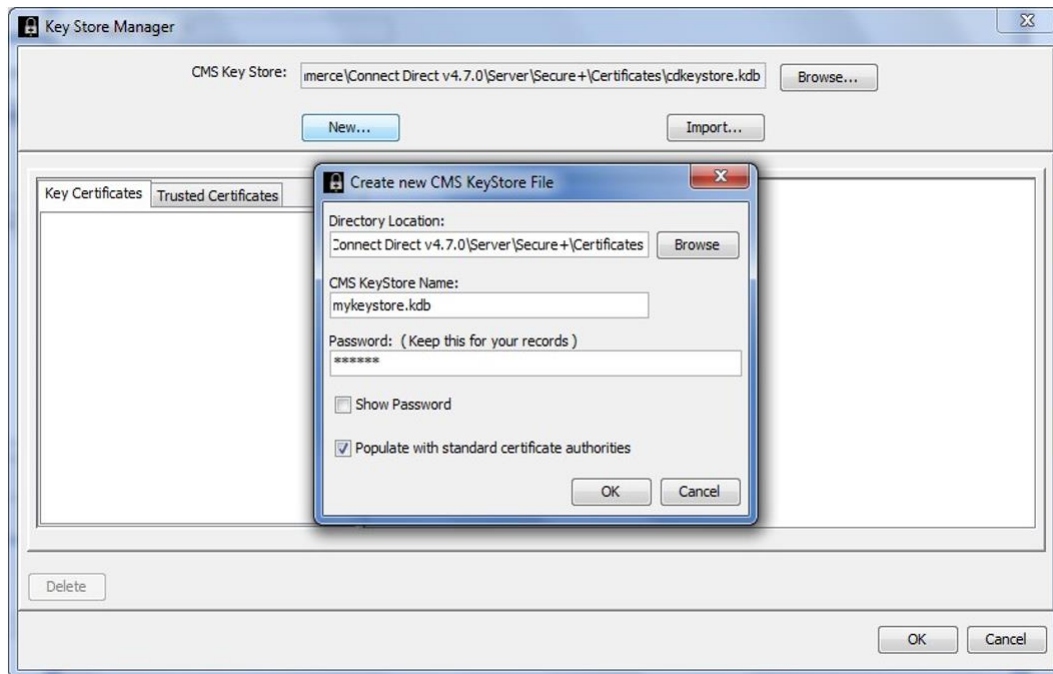
Go to **Key Management** in the pull-down menu and select **Configure Key Store**.



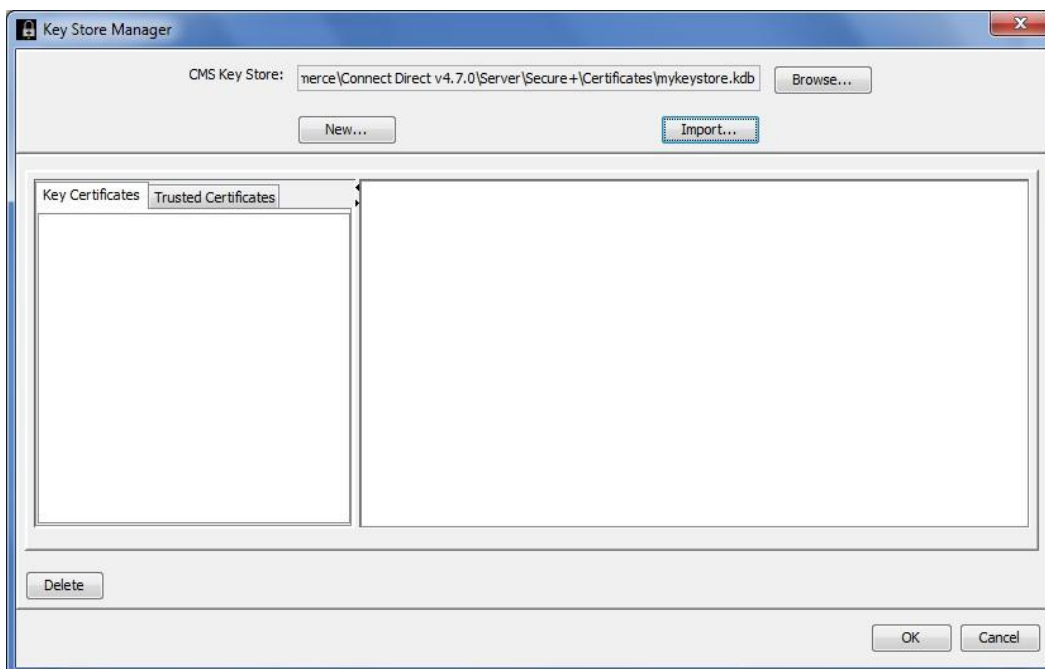
On the **Key Store Manager** window, select **New**. The **Create new CMS KeyStore File** dialog box appears.



Enter the Directory location (you can also **Browse** to the location desired), the KeyStore file name, and the password for the new KeyStore file. It is recommended you select **Populate with standard certificate authorities**, which imports all standard public CA Root certificates into the new KeyStore.



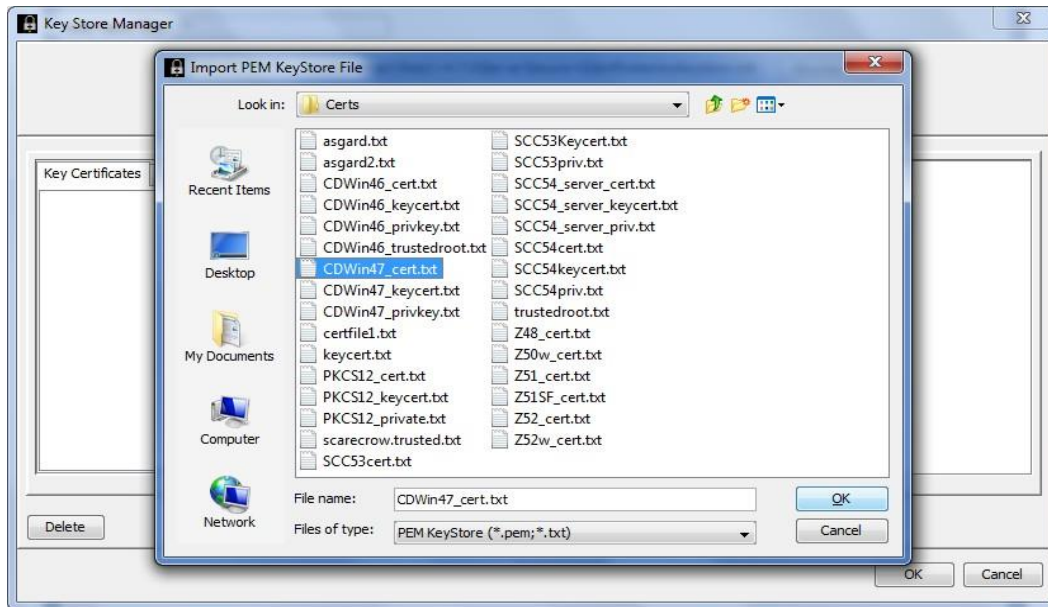
Select **OK** to create the new **CMS KeyStore** file. Key Store Manager will display contents of the new KeyStore. Your new KeyStore name should be in the **CMS Key Store** field.



Import Certificate into CMS KeyStore

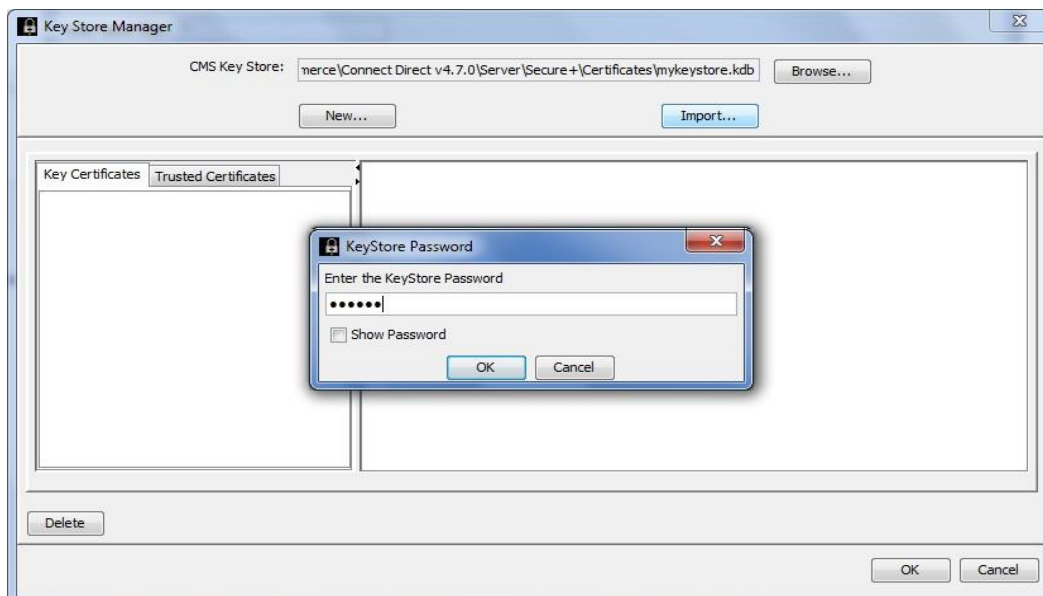
Start **C:D Secure+ Admin**, go to **Key Management** in the pull-down menu and select **Configure Key Store**. Either accept the current keystore (select **OK**) or browse to where your keystore is located, then select **OK**.

In **Key Store Manager**, select **Import**. On the **Import PEM KeyStore File** window, navigate to the folder where your certificate file is located and select the certificate file you want to use and select **OK**.



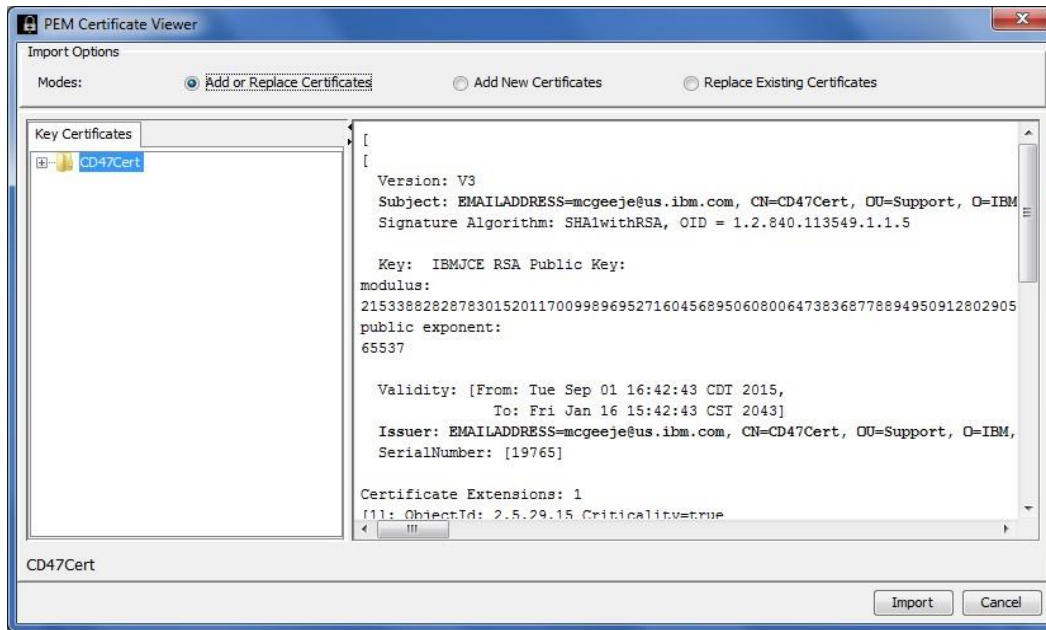
If a key certificate file is being imported, the **KeyStore Password** window appears. Type your password (for the .PEM keycert, not the keystore) and select **OK**.

NOTE: If you are migrating from an older C:D Windows or are using certificates that were not built using **IKEYMAN**, you will need to import the keycert (that is, private and public keys). If you used **IKEYMAN** to build the certificate, you only need to import the public certificate, since the private key is already in the keystore.



NOTE: **IKEYMAN** does not support the .PEM format for keycert (the format used prior to C:D Windows 4.7 and C:D Unix 4.2) as far as importing and exporting. The Secure+ Admin tool will need to be used to import the .PEM file, then it can be used in **IKEYMAN**.

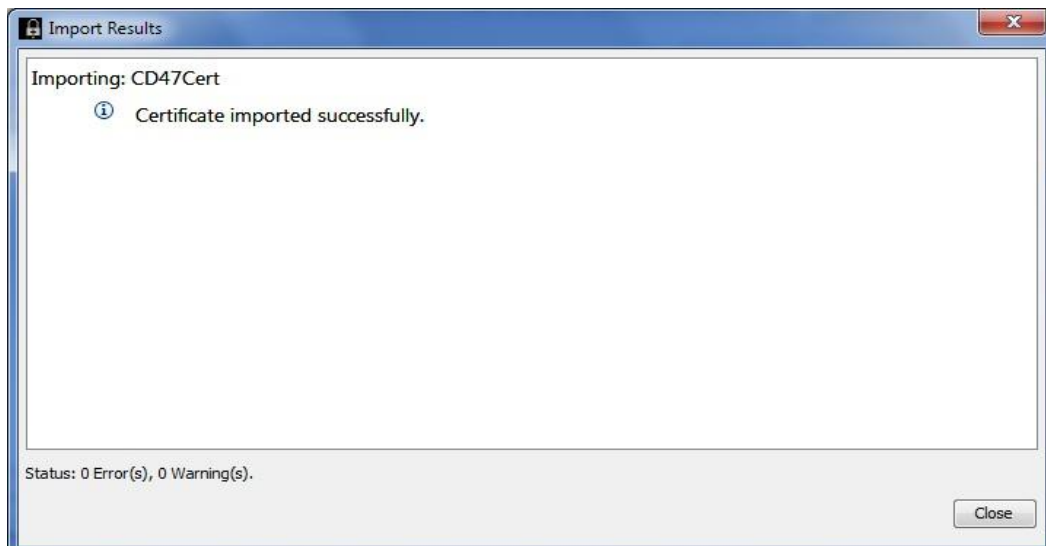
The **PEM Certificate Viewer** displays to allow a review of the certificate file. Verify the certificate is valid and select the **Import** button.



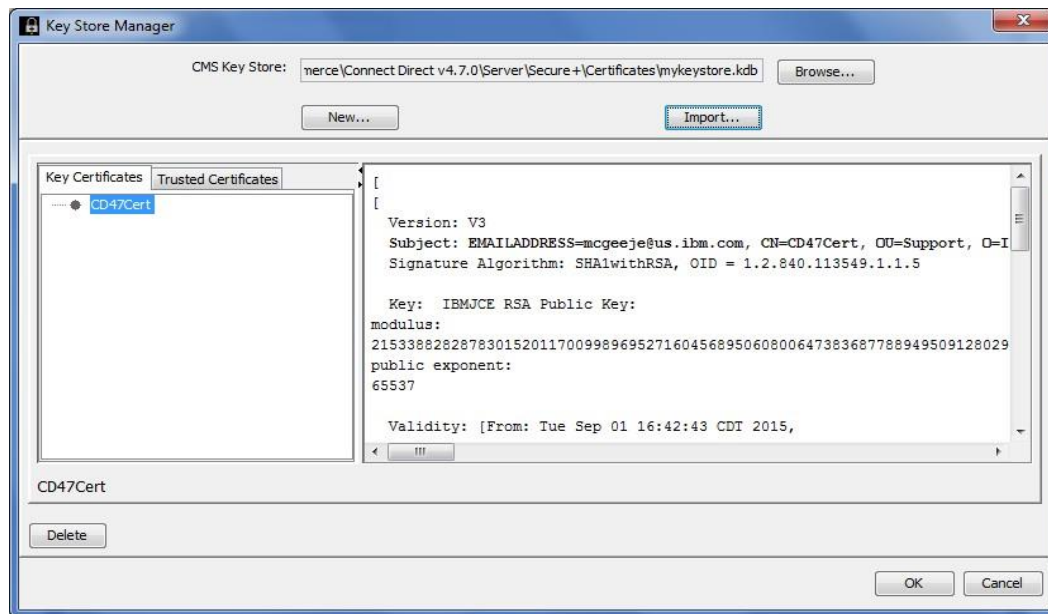
The certificate is imported and given a Label based on the certificate Common Name, (CN=). Note the serial number to identify the correct certificate after import.

NOTE: A common name is used for Label and identification therefore multiple certificates can have the same common name and therefore, can be overwritten depending on the setting of the Default Mode. Additionally, the Default Mode of Import is **Add or Replace Certificates**.

Import Results window displays with status of imported certificate. Select **Close**.



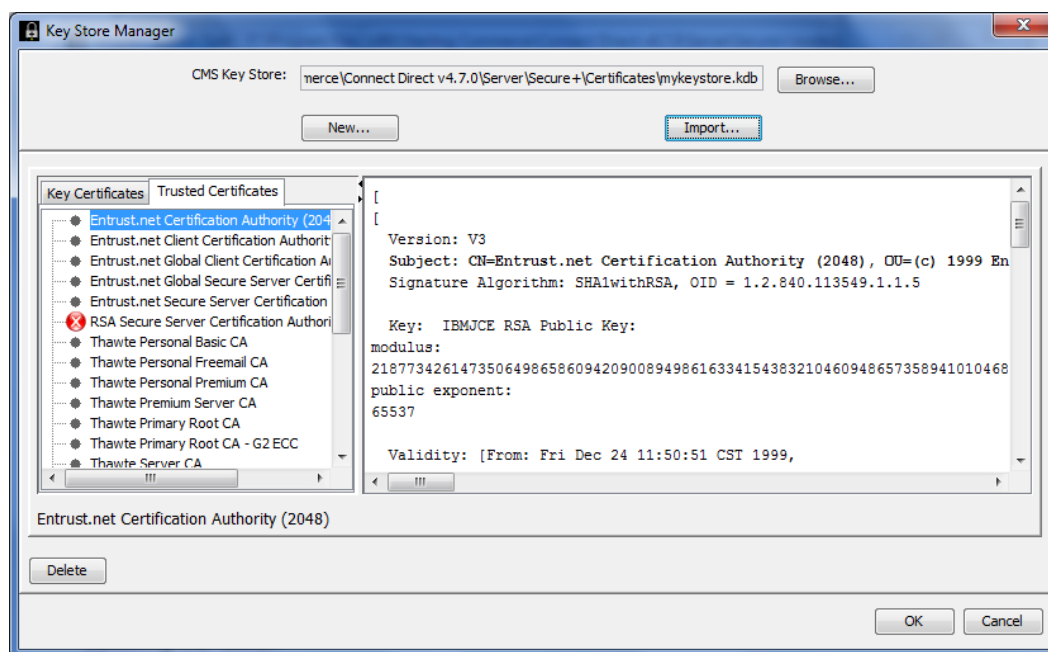
In the **Key Store Manager**, you should see something like the following on the **Key Certificates** tab:



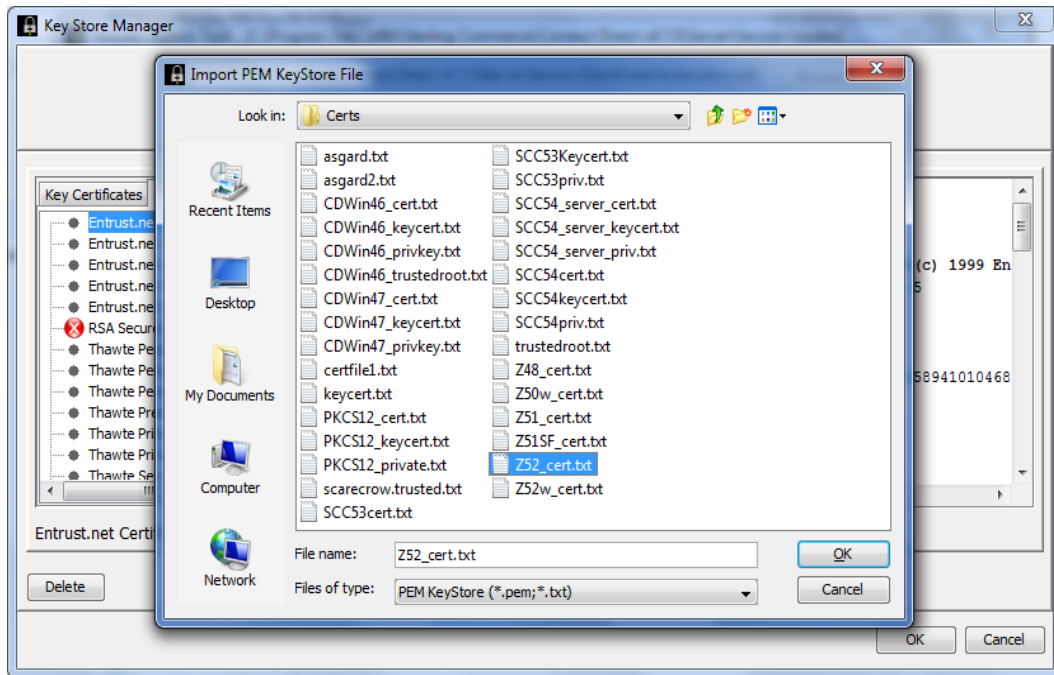
NOTE: If you do not see your keycert on the **Key Certificates** tab, then it has not been properly imported.

Next, you will need to import the remote certificate from your trading partner.

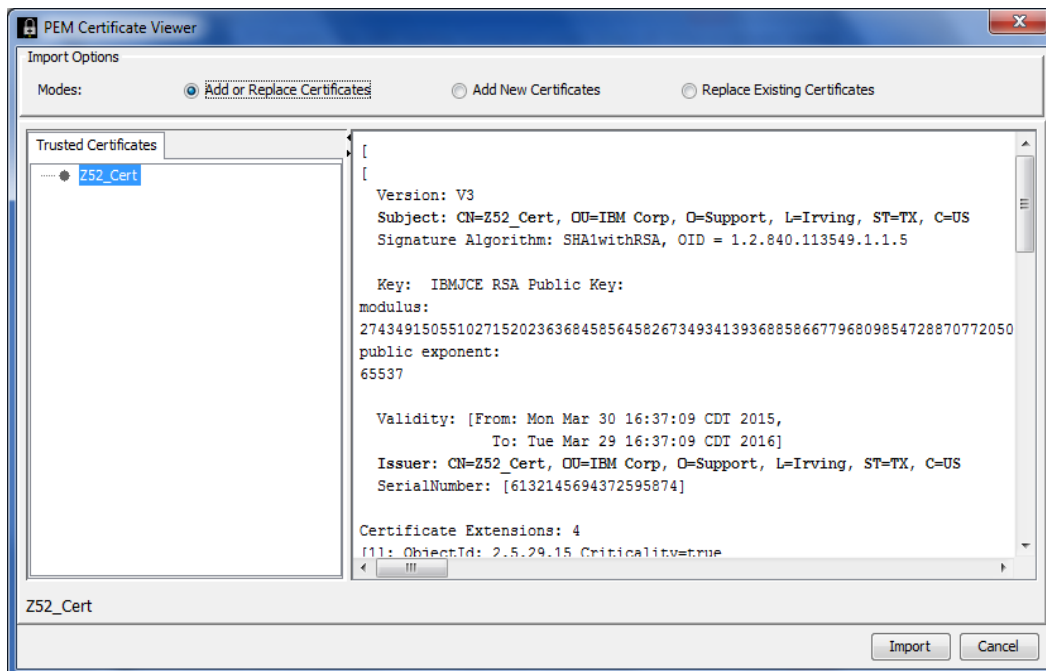
Select the **Trusted Certificates** tab.



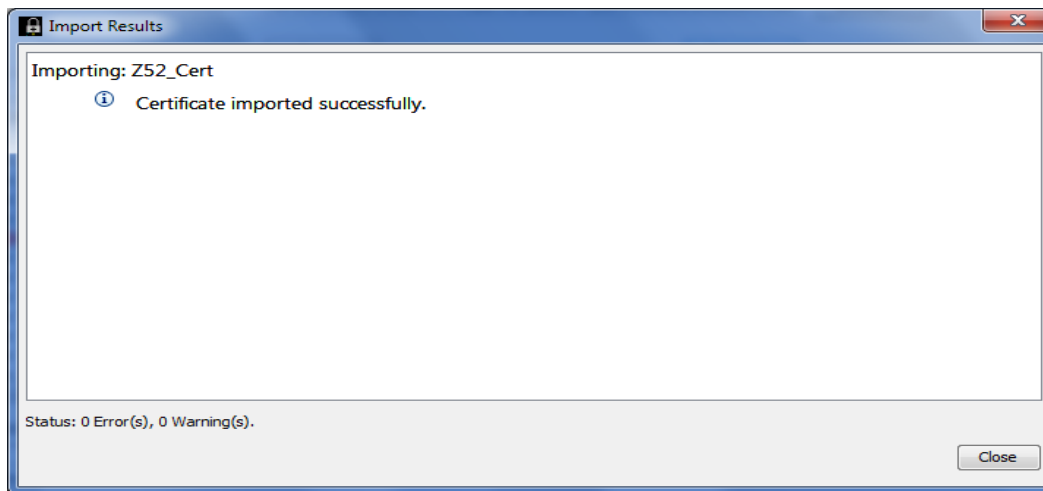
Select **Import** and select the remote certificate you need to import.



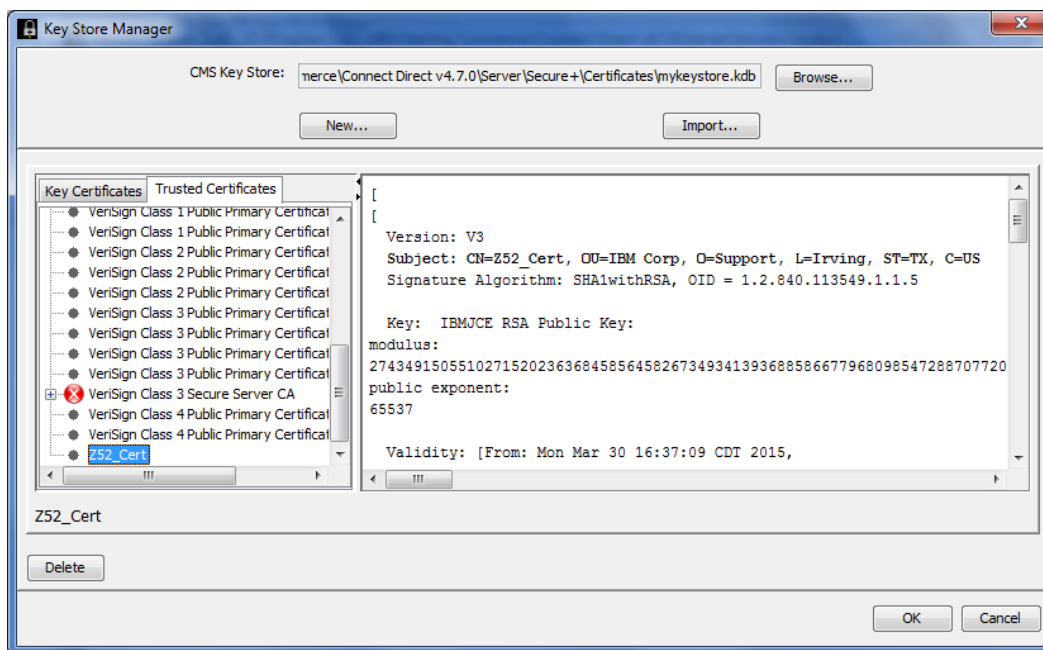
The **PEM Certificate Viewer** displays to allow a review of the certificate file. Verify the certificate is valid and select the **Import** button.



Import Results window displays with status of imported certificate. Select **Close**.



In the **Key Store Manager**, you should see your certificate on the **Trusted Certificates** tab (you may have to scroll down to find it):



NOTE: If you do not see your certificate on the **Trusted Certificates** tab, then it has not been properly imported.

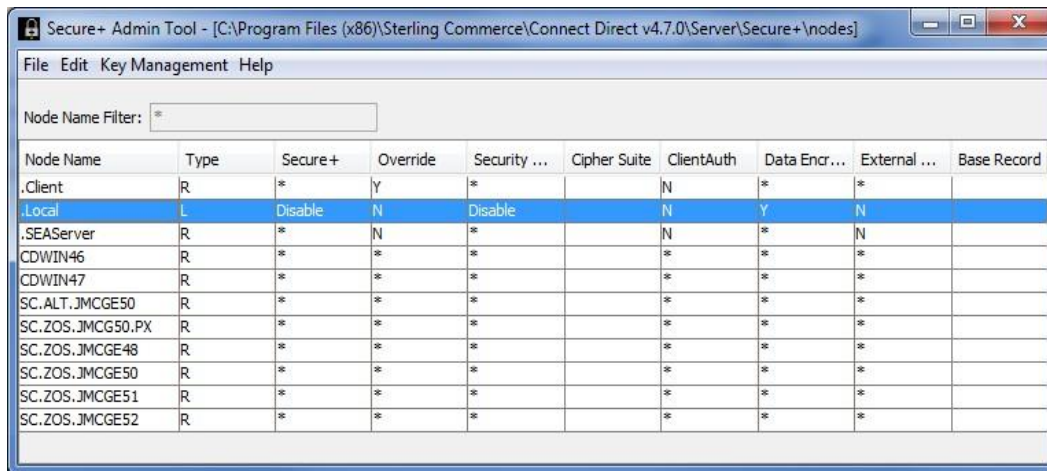
Select **OK** to exit the **Key Store Manager** (it may take a few second for this to complete).

You should now be in the Secure+ Admin Tool.

Update the .Local Node for C:D Windows / C:D Unix

NOTE: You must add or import the key certificate into your key store prior to configuring your node. If this has not been done, please go back and complete section **Import Certificate into CMS KeyStore**.

Select **.Local** by double-clicking on it.



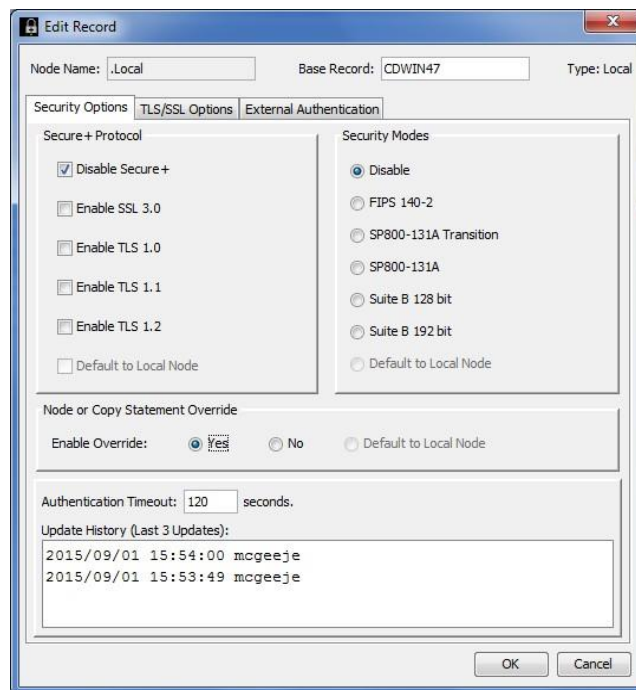
Secure+ Admin Tool - [C:\Program Files (x86)\Sterling Commerce\Connect Direct v4.7.0\Server\Secure+\nodes]

File Edit Key Management Help

Node Name Filter: *

Node Name	Type	Secure+	Override	Security ...	Cipher Suite	ClientAuth	Data Encr ...	External ...	Base Record
.Client	R	*	Y	*		N	*	*	
.Local	L	Disable	N	Disable		N	Y	N	
.SEAServer	R	*	N	*		N	*	N	
CDWIN46	R	*	*	*		*	*	*	
CDWIN47	R	*	*	*		*	*	*	
SC.ALT.JMCGE50	R	*	*	*		*	*	*	
SC.ZOS.JMCG50.PX	R	*	*	*		*	*	*	
SC.ZOS.JMCGE48	R	*	*	*		*	*	*	
SC.ZOS.JMCGE50	R	*	*	*		*	*	*	
SC.ZOS.JMCGE51	R	*	*	*		*	*	*	
SC.ZOS.JMCGE52	R	*	*	*		*	*	*	

The first panel you should see is the **Security Options** panel (if not, please select the **Security Options** tab):



Edit Record

Node Name: .Local Base Record: CDWIN47 Type: Local

Security Options | TLS/SSL Options | External Authentication

Secure+ Protocol

- ☒ Disable Secure+
- ☐ Enable SSL 3.0
- ☐ Enable TLS 1.0
- ☐ Enable TLS 1.1
- ☐ Enable TLS 1.2
- ☐ Default to Local Node

Security Modes

- ☒ Disable
- ☐ FIPS 140-2
- ☐ SP800-131A Transition
- ☐ SP800-131A
- ☐ Suite B 128 bit
- ☐ Suite B 192 bit
- ☐ Default to Local Node

Node or Copy Statement Override

Enable Override: ☒ Yes ☐ No ☐ Default to Local Node

Authentication Timeout: 120 seconds.

Update History (Last 3 Updates):

```
2015/09/01 15:54:00 mgeeeje
2015/09/01 15:53:49 mgeeeje
```

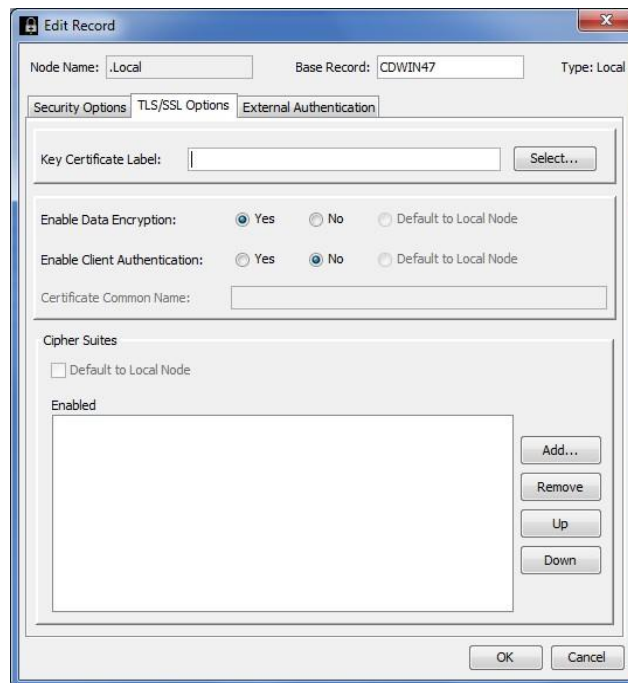
OK Cancel

You can choose to code a **Security Protocol** or **Security Mode** but make sure that **Enable Override** is set to **Yes**. The other settings can be left as is.

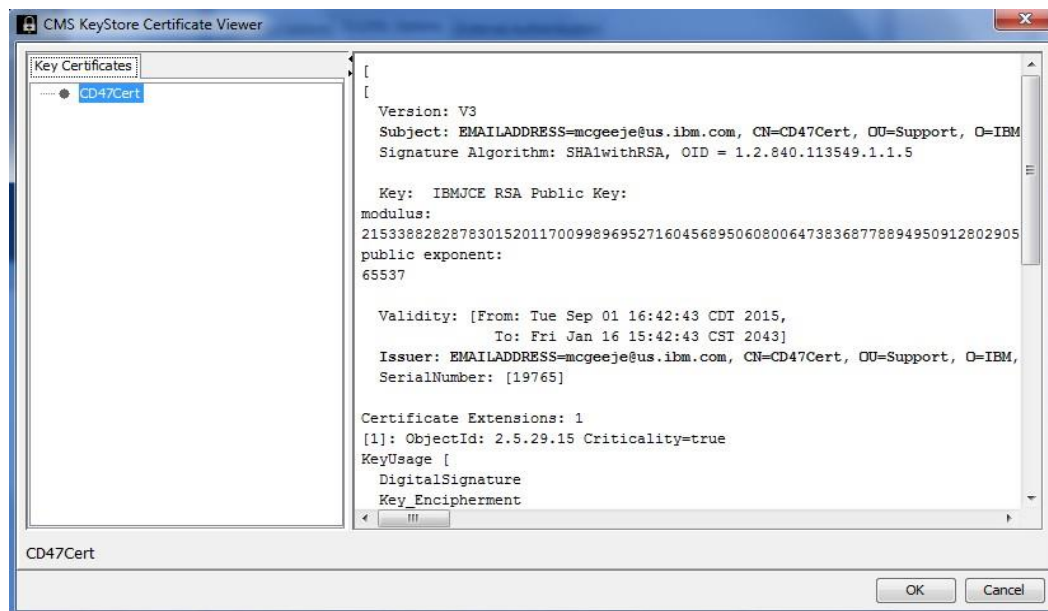
NOTE: The settings on the **.Local** are the defaults for your system. It is recommended, unless otherwise warranted, to set **.Local** to **Disable Secure+** and **Enable Override** to **Yes**. This means your system default will be non-Secure+ (which allows non-Secure+ nodes to function properly), but will also allow each remote node to be set differently according to how they are needed and override the **.Local** settings.

If a Secure+ protocol is set on the **.Local**, then any non-Secure+ remote node will need to be added to the Secure+ parameter file and have Secure+ disabled.

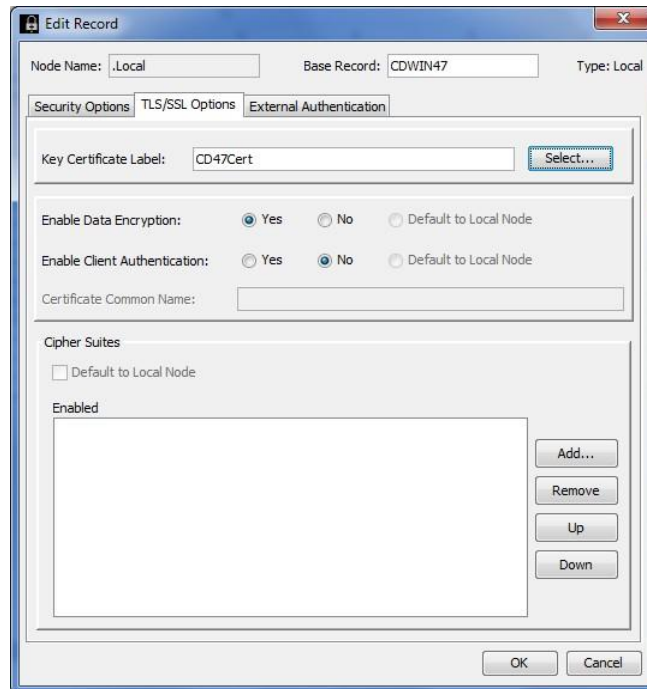
Go to the **TLS/SSL Options** tab. The **TLS/SSL Options** dialog box is displayed.



Select an existing Key Certificate from the key store. To select a **Key Certificate** from the keystore, select **Select** next to **Key Certificate Label**. The **CMS KeyStore Certificate Viewer** appears.



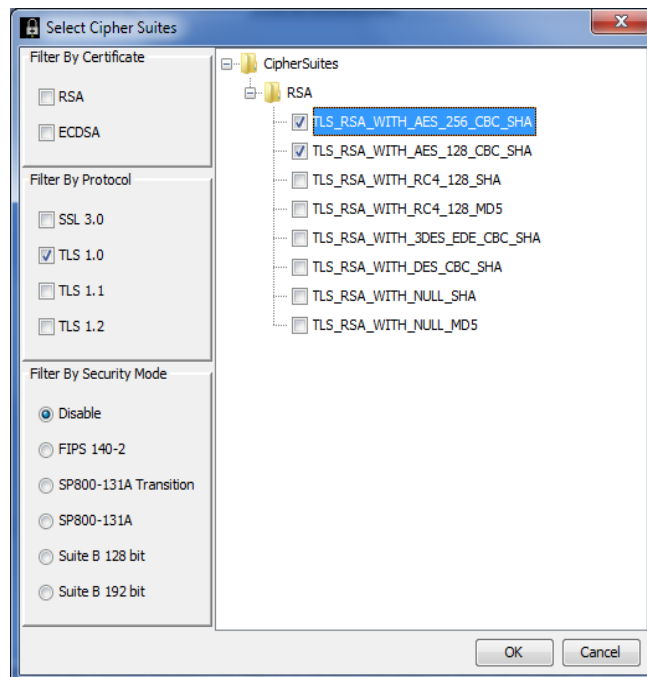
In the Key Certificates area, select the key certificate you want to use and select **OK** box. If there is only one, simply select **OK**.



You should now see the Label in the **Key Certificate Label** field.

Now you need to select the Cipher Suite(s) needed for the Secure+ connection.

Under **Cipher Suites**, select on **Add**:



Select **OK**.

You are not back at the **.Local Edit Record**. Notice that the selected ciphers are now **Enabled** in the **Cipher Suites** parameter field:

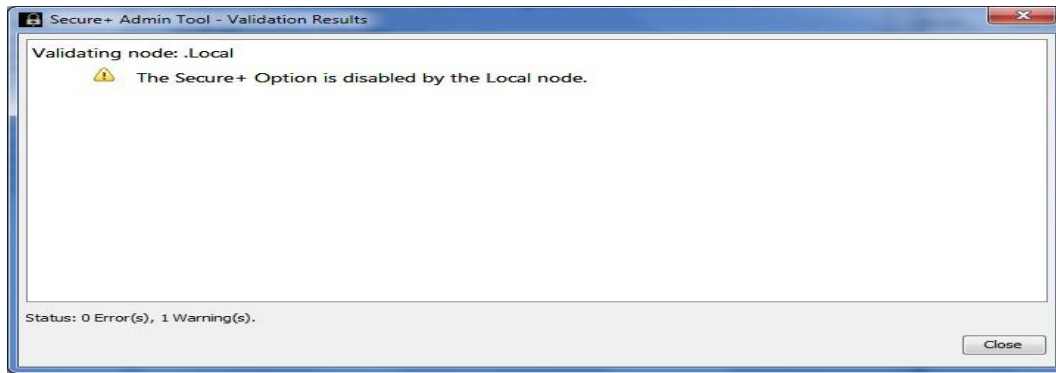
The screenshot shows the 'Edit Record' dialog box with the 'Security Options' tab selected. The 'Node Name' is '.Local', 'Base Record' is 'CDWIN47', and 'Type' is 'Local'. Under 'Security Options', 'Key Certificate Label' is 'CD47Cert'. 'Enable Data Encryption' is set to 'Yes'. 'Enable Client Authentication' is set to 'No'. 'Certificate Common Name' is empty. The 'Cipher Suites' section shows 'Default to Local Node' unchecked, and a list of 'Enabled' ciphers: 'TLS_RSA_WITH_AES_256_CBC_SHA' and 'TLS_RSA_WITH_AES_128_CBC_SHA'. Buttons for 'Add...', 'Remove', 'Up', and 'Down' are on the right. 'OK' and 'Cancel' buttons are at the bottom.

External Authentication (SEAS) defaults to **No**; it is recommended, unless necessary, to leave it as **No**. If this is needed, select on the **External Authentication** tab, select **Yes** in the **Enable External Authentication** box, then type the **Certificate Validation Definition** character string defined in Sterling External Authentication Server.

The screenshot shows the 'Edit Record' dialog box with the 'External Authentication' tab selected. 'Enable External Authentication' is set to 'No'. 'Certificate Validation Definition' is empty. The 'Server Properties' section has 'Host Name' and 'Port' fields, both empty. 'OK' and 'Cancel' buttons are at the bottom.

Select **OK** to close the **Edit Record** dialog box and update the parameters file.

NOTE: When you select **OK**, you may see a **Validation Results** box like the following:



This is typically a warning or an informational message; an error will prevent the **.Local** from being saved properly. Select **Close**.

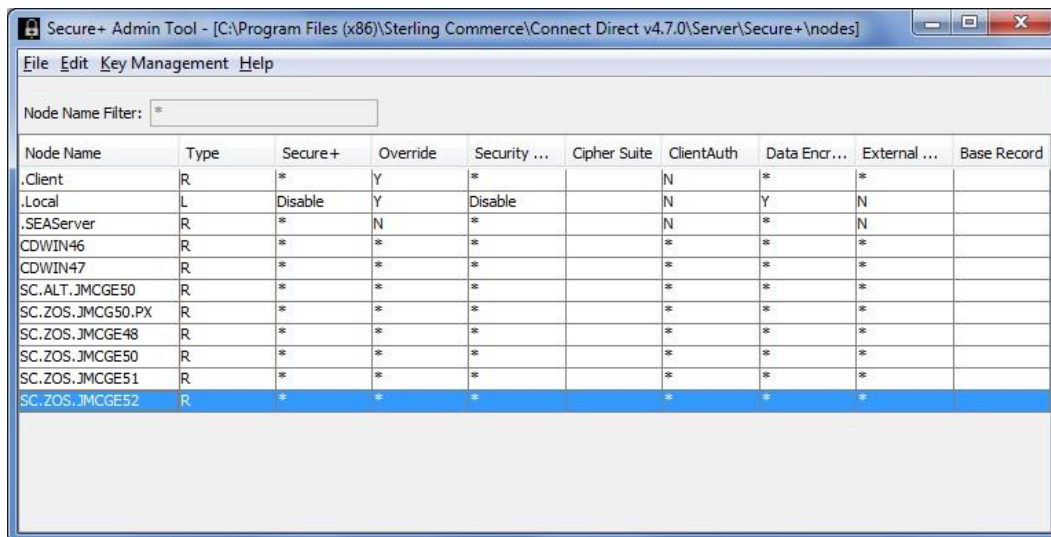
You should now be back in the Secure+ Admin Tool.

You now need to update the Remote Node entry.

Update the Remote Node for C:D Windows / C:D Unix

NOTE: You must add or import the key certificate into your key store prior to configuring your node. If this has not been done, please go back and complete section **Import Certificate into CMS KeyStore**.

Select the Remote Node you want to update and double-click on it:



In the Edit Record dialog box, select the **Security Options** tab.

The screenshot shows the 'Edit Record' dialog box with the 'Security Options' tab selected. The 'Node Name' is 'SC.ZOS.JMCGE52', 'Base Record' is 'SC.ZOS.JMCGE52', and 'Type' is 'Remote'. The 'Security Options' tab has three sub-tabs: 'Security Options', 'TLS/SSL Options', and 'External Authentication'. The 'Security Options' sub-tab is active, showing 'Secure+ Protocol' and 'Security Modes' sections. In 'Secure+ Protocol', 'Disable Secure+' is unchecked, 'Enable SSL 3.0' is unchecked, 'Enable TLS 1.0' is unchecked, 'Enable TLS 1.1' is unchecked, 'Enable TLS 1.2' is unchecked, and 'Default to Local Node' is checked. In 'Security Modes', 'Disable' is selected. Below these, 'Node or Copy Statement Override' has 'Enable Override' set to 'Default to Local Node'. The 'Authentication Timeout' is 120 seconds. The 'Update History (Last 3 Updates)' shows '2015/09/02 10:37:57 mcgeeje'. 'OK' and 'Cancel' buttons are at the bottom.

The default setting is **Default to Local Node**. If the **.Local** is set to a particular protocol that will be used by this remote node, this can stay as is; however, assuming **.Local** is set to **Disable Secure+** or a protocol different than what is needed, you will need to set your desired protocol.

In this example, setting **Enable TLS 1.0** is selected (equivalent to TLS is prior versions).

This screenshot is identical to the previous one, but with 'Enable TLS 1.0' checked in the 'Secure+ Protocol' section. The 'Security Modes' section remains unchanged with 'Disable' selected. All other settings, including the 'Update History' and buttons, are the same.

Select the **TLS/SSL Options** tab. The **TLS/SSL Options** dialog box is displayed.

The screenshot shows the 'Edit Record' dialog box with the 'TLS/SSL Options' tab selected. The 'Node Name' is 'SC.ZOS.JMGE52', 'Base Record' is 'SC.ZOS.JMGE52', and 'Type' is 'Remote'. The 'Security Options' section includes a 'Key Certificate Label' field with a 'Select...' button. Below this are two sections: 'Enable Data Encryption' and 'Enable Client Authentication', each with radio buttons for 'Yes', 'No', and 'Default to Local Node'. The 'Certificate Common Name' field is empty. The 'Cipher Suites' section has a checked 'Default to Local Node' checkbox and an 'Enabled' list box with 'Add...', 'Remove', 'Up', and 'Down' buttons. At the bottom are 'OK' and 'Cancel' buttons.

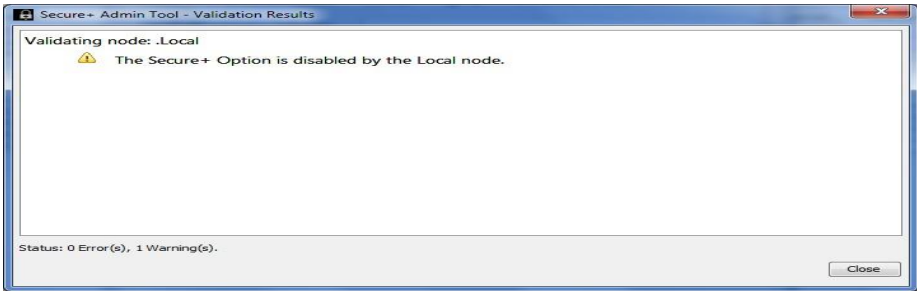
If this Remote Node needs an alternate site certificate (i.e. you need to use a different certificate with this trading partner), click **Select** next to **Key Certificate Label** to select an existing Key Certificate. The **CMS KeyStore Certificate Viewer** appears. In the **Key Certificates** area, select the key certificate you want to use and select **OK** box.

External Authentication (SEAS) defaults to **No**; it is recommended, unless necessary, to leave it as **No**. If this is needed, select on the **External Authentication** tab, select **Yes** in the **Enable External Authentication** box, then type the **Certificate Validation Definition** character string defined in Sterling External Authentication Server (this will have to be obtained from SEAS).

The screenshot shows the 'Edit Record' dialog box with the 'External Authentication' tab selected. The 'Node Name' is 'SC.ZOS.JMGE52', 'Base Record' is 'SC.ZOS.JMGE52', and 'Type' is 'Remote'. The 'Security Options' section includes 'Enable External Authentication' with radio buttons for 'Yes', 'No', and 'Default to Local Node'. Below this is the 'Certificate Validation Definition' field. The 'Server Properties' section has 'Host Name' and 'Port' fields. At the bottom are 'OK' and 'Cancel' buttons.

Select **OK** to close the **Edit Record** dialog box and update the parameters file.

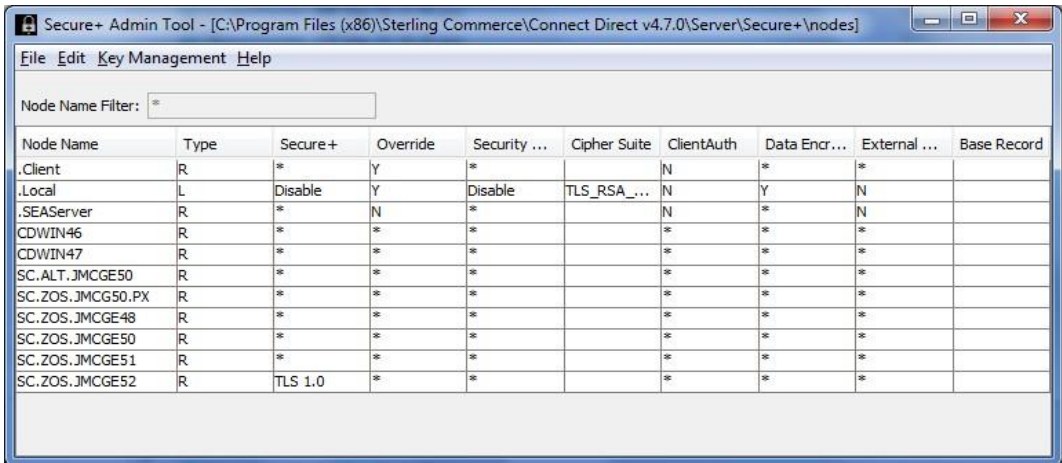
NOTE: When you select **OK**, you may see a **Validation Results** box like the following:



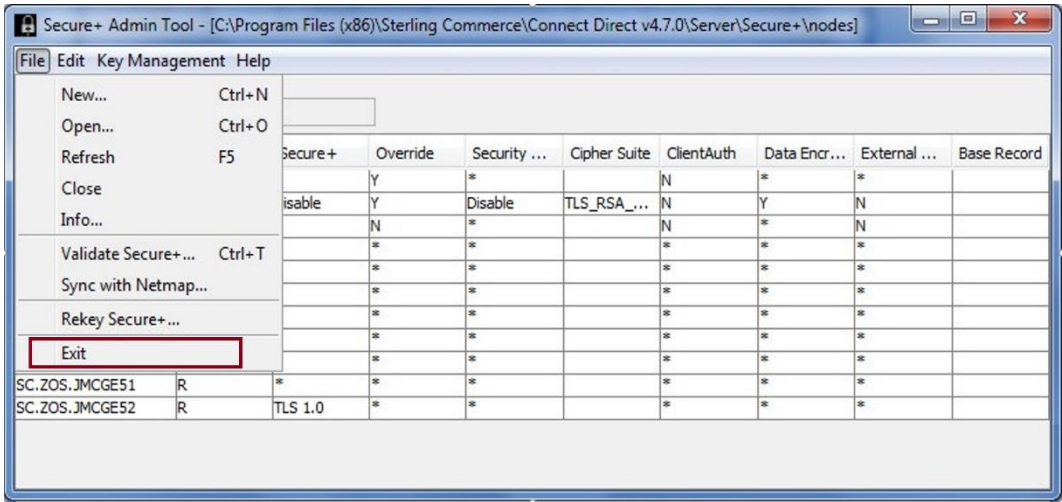
This is typically a warning or an informational message; an error will prevent the .Local from being saved properly. Select **Close**.

You should now be back in the Secure+ Admin Tool.

Your Remote Node now should show the Secure+ protocol that you selected.



When finished, go to **File** (pull-down menu) and select **Exit** to save the Secure+ PARMFILE and exit the Secure+ Admin Tool.



At this point, once the Secure+ PARMFILE is set up on the remote node and the C:D Windows certificate is properly exchanged to that remote node, this Secure+ connection should work.

Using Secure+ with z/OS Security Applications

If using a keyring and not a key database using gskkyman, Connect:Direct Secure+ supports using a keyring in three different security applications: RACF, CA-ACF2, and CA-Top Secret.

The following section is divided into three sections:

1. Secure+ using RACF
2. Secure+ using CA-ACF2
3. Secure+ using CA-Top Secret

Secure+ Using RACF

The following commands described below are examples used to create RACF definitions to be used with Connect:Direct z/OS Secure+. The commands should always be associated to the user under which Connect:Direct z/OS will run.

Create RACF Keyring

While the Certification Authority generates the certificate for the Certificate Request (CSR) created and sent above, a RACF keyring should be created to be used by Connect:Direct z/OS Secure+. Remember that every RACF command should be associated to the user under which Connect:Direct z/OS Secure+ will run.

```
===> RACDCERT ID(STCNDM) ADDRING(SECUREIBMRING3)
```

Other related commands:

```
***** RACDCERT ID(STCNDM) LISTRING(SECUREIBMRING3)
***** RACDCERT ID(STCNDM) DELRING(SECUREIBMRING3)
***** RACDCERT ID(STCNDM) LISTRING(*)
```

NOTE: Remember this keyring as it will be needed as **Certificate PATHNAME** on local node entry in the Secure+ PARMFILE.

Generate Certificate Request (CSR)

A Certificate Request (CSR) will need to be generated under RACF; two commands will be used:

1. Command used to generate a Certificate Request (CSR).

```
===> RACDCERT ID(STCNDM) GENCERT SUBJECTSDN(CN('IBMPIL3.CSR.IBM') O('IBM')
OU('TELECOM') C('US')) SIZE(2048) WITHLABEL('ZOS CSR CERTIFICATE')
```

CN – COMMON NAME – This value can optionally be declared on a node in the Secure+ PARMFILE **SSL/TLS Parameters** panel on field **Certificate Common Name**.

2. Command used to export the previously generated Certificate Request (CSR) so that it can be sent to a Certification Authority (CA).

```
====> RACDCERT ID(STCNDM) GENCERT GENREQ(LABEL('ZOS CSR CERTIFICATE'))  
DSN('DCA.RACF.CERT3')
```

Other related commands:

```
***** RACDCERT ID(STCNDM) LIST(LABEL('OS390 CERTIFICATE'))
```

```
***** RACDCERT ID(STCNDM) REMOVE(LABEL('UTELK103') RING(SECUREIBMRING3))
```

```
***** RACDCERT ID(STCNDM) DELETE(LABEL('OS390 CERTIFICATE'))
```

The file created on z/OS should be downloaded as ASCII to Windows or Unix so that it can be sent to a Certification Authority (CA) by e-mail or HTTP, for example.

Generate Self-Signed Certificate

Generate a digital certificate containing a private key for the RACF address space.

```
====> RACDCERT ID(STCNDM) GENCERT SUBJECTSDN(CN('IBMPIL3.CERT.IBM') O('IBM')  
C('US')) SIZE(2048) WITHLABEL('ZOS CERTIFICATE') SIGNWITH(CERTAUTH  
LABEL('RACFCA')) KEYUSAGE(HANDSHAKE DATAENCRYPT DOCSIGN))
```

CN – COMMON NAME – This value can optionally be declared on a node in the Secure+ PARMFILE **SSL/TLS Parameters** panel on field **Certificate Common Name**.

Load Certificate into Keyring

The Certification Authority (CA) will send back two files. The first will contain one or more **Trusted (root) Certificate(s)** of the Certification Authority (CA) itself. The second file will contain the certificate generated for the Certificate Request described above. This is the **Connect:Direct z/OS Secure+ Certificate**.

RACF already has some Certification Authorities (CA) defined. If the Certification Authority (CA) is one of those, it is not necessary to receive its trusted (root) certificate.

For instance, the trusted (root) certificates below are already received under RACF as of z/OS R2.1:

VeriSign Class 1 Public Primary Certification Authority
VeriSign Class 2 Public Primary Certification Authority
VeriSign Class 3 Public Primary Certification Authority
Thawte Server Certificate Authority
Thawte Premium Server Certificate Authority
Thawte Personal Basic Certificate Authority
Thawte Personal Freemail Certificate Authority
Thawte Personal Premium Certificate Authority
Equifax Secure Certificate Authority

Equifax Secure eBusiness Certificate Authority-1
Equifax Secure eBusiness Certificate Authority-2
Equifax Secure Global eBusiness Certificate Authority-1
VeriSign Class 1 Public Primary Certificate Authority - G2
VeriSign Class 2 Public Primary Certificate Authority - G2
VeriSign Class 3 Public Primary Certificate Authority - G2
VeriSign Class 4 Public Primary Certificate Authority - G2
VeriSign Class 1 Public Primary Certificate Authority - G3
VeriSign Class 2 Public Primary Certificate Authority - G3
VeriSign Class 3 Public Primary Certificate Authority - G3
VeriSign Class 4 Public Primary Certificate Authority - G3
VeriSign Class 3 Public Primary Certificate Authority - G5

Related commands:

***** *RACDCERT LIST CERTAUTH*

1. **Trusted (root) Certificate:** If the Certification Authority (CA) is not one listed above it will be necessary to add its trusted (root) certificate onto RACF.

The file received from the Certification Authority (CA) that contains its trusted (root) certificate should be written onto z/OS with the following DCB:

DCB= (RECFM=VB , LRECL=84 , DSORG=PS)

Then the following command should be issued:

```
====> RACDCERT CERTAUTH ADD('DCA.RACF.CERT5') LABEL('LABEL00000001') TRUST
```

Then the trusted (root) certificate received on the previous commands should be assigned to the keyring defined on step 3; that's the keyring that will be used by Connect:Direct z/OS Secure+.

```
====> RACDCERT ID(STCNDM) CONNECT(CERTAUTH LABEL('LABEL00000001')  
RING(SECUREIBMRING3) USAGE(CERTAUTH))
```

2. **Connect:Direct z/OS Secure+ certificate:** The file received from the Certification Authority (CA) that contains the Connect:Direct z/OS Secure+ certificate should be written onto z/OS with the following DCB:

DCB= (RECFM=VB , LRECL=84 , DSORG=PS)

Then the following command should be issued:

```
====> RACDCERT ID(STCNDM) ADD('DCA.RACF.CERT') WITHLABEL('ZOS CERTIFICATE')
```

Other related commands:

***** *RACDCERT ID(STCNDM) REMOVE(LABEL('UTELK103') RING(SECUREIBMRING3))*

3. The last step will be to associate Connect:Direct z/OS Secure+ certificate to the keyring defined previously; that is the keyring that will be used by Connect:Direct z/OS Secure+:

```
===> RACDCERT ID(STCNDM) CONNECT(ID(STCNDM) LABEL('ZOS CERTIFICATE')  
RING(SECUREIBMRING3) DEFAULT USAGE(PERSONAL))
```

The keyring (SECUREHSBCRING3) in the example above will be declared in the Secure+ PARMFILE Local node definition on the field **Certificate Pathname** on the **SSL/TLS Parameters** panel of the Local node.

The Certificate (ZOS PILOTO3 CERTIFICATE) in the example above will be declared on Secure+ PARMFILE **SSL/TLS Parameters** panel on the field **Certificate Label**.

For further information and/or clarification of RACF commands regarding certificates, please refer to manual **“z/OS Security Server RACF Security Administrator's Guide”**.

Secure+ Using CA-ACF2

The following are only examples for a point of reference. The actual commands shown have not been run in the respective environments. Please check the syntax carefully before using them as a guide.

NOTE: CA-ACF2 has the concept of record id or record key. The record key is logonid.suffix. **logonid** is a one to eight character user id, **CERTAUTH** (Certificate Authority) or **SITECERT** (Site Certificate), followed by a dot (.), followed by a one to eight character suffix. The suffix can be defined upon creation of a CA-ACF2 item or allow CA ACF2 to define it. The record key can be used as an abbreviation to access the CA-ACF2 item. The record key will not be used in the examples that follow. The **logonid**, **CERTAUTH** or **SITECERT** must always be entered if the record key is not used.

* **CA-ACF2** is a registered trademark of Computer Associates Technologies, Copyright © 2017 CA. All rights reserved. To the best of our knowledge the information presented below is accurate, but the user should consult CA-ACF2 documentation for information about CA-ACF2 and thoroughly test the examples before using them.

CA-ACF2 Identifiers

The following identifiers are used in the commands below but they can be any characters and case the customer desires. They are entered in lower case to distinguish them from actual command words. They are quoted in this note but the command will show whether they should be single quoted or not.

1. **cdid** is the Connect:Direct DTF userid.
2. **keyringname** is the Keyring name used by Connect:Direct. Keyring names are case sensitive and can be as long as 237 characters.
3. **cdservcert** is the Connect:Direct End-User Server/Client Certificate Label. Label can be any name desired but using the Subject Common Name will provide a better understanding of the contents of the certificate. It is case sensitive and can be up to 237 characters.
4. **rootcert** is a trusted root (or CA) certificate Label. It is case sensitive and can be up to 237 characters.
5. **commonname** is Certificate Common Name. The Subject Common Name of the End-User Server Certificate would be given to the Remote Connect:Direct node if they wanted to do Client Authentication. It is required.
6. **organization unit** is Organizational Unit. It is an optional entry. (e.g. Support)
7. **organization** is Organization. It is required. (e.g. IBM Corp)
8. **location** is City/Locality. It is an optional entry. (e.g. Irving)
9. **state** is State or Province. It is an optional entry. (e.g. TX)
10. **country** is a 2-character Country or Region identifier. It is required. (e.g. US)
11. **begindate** is Certificate start or beginning date in the format mm/dd/yyyy.
12. **expiredate** is Certificate end or expiration date in the format mm/dd/yyyy.

List Connect:Direct Keyring and Certificate

Display keyring information

```
SET PROFILE(USER) DIV(KEYRING)
LIST cdid RINGNAME(keyringname)
```

NOTE: This **keyringname** will be needed as the **Certificate PATHNAME** on the local node entry in the Secure+ PARMFILE.

Display the certificate information:

```
SET PROFILE(USER) DIV(CERTDATA)
LIST cdid LABEL(cdsvrcert)
LIST CERTAUTH LABEL(rootcert)
```

Create Keyring

```
SET PROFILE(USER) DIV(KEYRING)
INSERT cdid RINGNAME(keyringname)
```

Generate a Root Certificate Authority (CA) Certificate

```
SET PROFILE(USER) DIV(CERTDATA)

GENCERT CERTAUTH -
    SUBJSDN(CN='commonname' -
            O='organization' -
            L='location' -
            SP='state' -
            C='country' -
            OU='organization unit') -
    LABEL(rootcert) -
    SIZE(1024) -
    KEYUSAGE(CERTSIGN) -
    TRUST -
    EXPIRE(expiredate)
```

NOTE: Self-signed certificate

Export a Root Certificate Authority (CA) Certificate

```
SET PROFILE(USER) DIV(CERTDATA)

EXPORT CERTAUTH -
    LABEL(rootcert) -
    DSNAME(' &hlq.base64.root') -
    FORMAT(CERTB64)
```

Export Certificate Format

CERTB64 - Specifies a DER encoded X.509 certificate that has been encoded using Base64. This is a text file, so it can be ship in an e-mail. If transferred using FTP or Connect:Direct, TEXT or ASCII mode must be used.

CERTDER - Specifies a DER encoded X.509 certificate. It is a binary file, so if transferred using FTP or Connect:Direct, BINARY mode must be used.

PKCS12B64 - Specifies a DER encoded PKCS#12 package that has been encoded using Base64. A PKCS12 PASSWORD must also be supplied. Export the certificate and the private key (which must exist and must not be an ICSF key). The package produced by specifying one of the PKCS #12 keywords is encrypted using the password specified according to the PKCS #12 standard. Processing will attempt to package any certificate-authority certificate necessary to complete the basing chain to the exported certificate. This is a text file, so it can be ship in an e-mail. If transferred using FTP or Connect:Direct, TEXT or ASCII mode must be used.

PKCS12DER - Specifies a DER encoded PKCS#12 package. A PKCS12 PASSWORD must also be supplied. Export the certificate and the private key (which must exist and must not be an ICSF key). The package produced by specifying one of the PKCS #12 keywords is encrypted using the password specified according to the PKCS #12 standard. Processing will attempt to package any CA certificate necessary to complete the basing chain to the exported certificate. It is binary, so if transferred using FTP or Connect:Direct, BINARY mode must be used.

NOTE: **PKCS12B64** is the default if PASSWORD is specified; otherwise, **CERTB64** is the default.

NOTE: The File Format of an Export is determined by the support on the receiving system. When receiving system is a platform that uses the C:D Certificate Wizard or is z/OS V1R2 or earlier system, the selected format must be **CERTB64** or **PKCS12DER** (when exporting certificate with private key). **CERTB64** provides a file format that can be directly copied into a Certificate Wizard based trusted root file.

CA-ACF2 Keyring Commands

Generate a Signed End-User Server/Client Certificate

```
SET PROFILE(USER) DIV(CERTDATA)
```

```
GENCERT cdid -
  SUBJSDN(CN='commonname' -
    O='organization' -
    L='location' -
    SP='state' -
    C='country' -
    OU='organization unit') -
  LABEL(cdsvrcert) -
  SIZE(1024) -
  KEYUSAGE(HANDSHAKE DATAENCRYPT DOCSIGN) -
  SIGNWITH(CERTAUTH LABEL(rootcert)) -
  TRUST -
  EXPIRE(expiredate)
```

NOTE: Signed by Root Certificate created earlier. Organization is acting as their own Certificate Authority (CA).

Export a Signed Certificate

```
SET PROFILE(USER) DIV(CERTDATA)
```

```
EXPORT cdid -
  LABEL(cdsvrcert) -
  DSNAME('&hlq.base64.servcert') -
  FORMAT(CERTB64)
```

Connect a Connect:Direct's End-User Server/Client Certificate to a Keyring

SET PROFILE (USER) DIV (KEYRING)

CONNECT CERTDATA (cdid)	-
LABEL (cdservcert)	-
KEYRING (cdid)	-
RINGNAME (keyringname)	-
USAGE (PERSONAL)	-
DEFAULT	

Connect a Root Certificate Authority (CA) Certificate to a Keyring

SET PROFILE (USER) DIV (KEYRING)

CONNECT CERTDATA (CERTAUTH)	-
LABEL (rootcert)	-
KEYRING (cdid)	-
RINGNAME (keyringname)	-
USAGE (CERTAUTH)	

Turn on Trust Status to a Root Certificate Authority (CA) Certificate

SET PROFILE (USER) DIV (CERTDATA)

CHANGE CERTAUTH LABEL (rootcert) TRUST

Provide Connect:Direct Access to its Certificates

Connect:Direct logonid (userid) will need READ access and resource rule to IRR.DIGTCERT.LIST in order to access certificates.

Provide Connect:Direct Access to its Keyring

Connect:Direct logonid (userid) will need READ access and resource rule to IRR.DIGTCERT.LISTRING in order to access its own keyring.

Connect:Direct Needs to Share a SITECERT Certificate

Connect:Direct logonid (userid) will need CONTROL access or WRITE Access and DELETE Resource rule to IRR.DIGTCERT.GENCERT in order to access certificates in SITECERT.

Reference

Computer Associates International Inc. Manuals eTrust™ CA ACF2 Security Cookbook for z/OS and OS/390

NOTE: Computer Associates Logon ID and Password is required to view their manuals. Computer Associates Site ID is required to create a Logon ID and Password.

Secure+ Using CA-Top Secret

NOTE: The following are only examples for a point of reference. The actual commands shown have not been run in the respective environments. Please check the syntax carefully before using them as a guide.

* **CA-Top Secret** is a registered trademark of Computer Associates Technologies, Copyright © 2017 CA. All rights reserved. To the best of our knowledge the information presented below is accurate, but the user should consult CA-Top Secret documentation for information about CA-Top Secret and thoroughly test the examples before using them.

CA-Top Secret Identifiers

The following identifiers are used in the commands below, but they can be any characters and case the customer desires. They are entered in lower case to distinguish them from actual command words. They are quoted in this note, but the command will show whether they should be single quoted or not.

1. **cdid** is the Connect:Direct DTF userid or ACID.
2. **keyringname** is the Keyring name. It is limited to 8 characters and must be specified.
3. **keyringlabel** is the Keyring name used by Connect:Direct. Keyring Label names are case sensitive and can be as long as 237 characters.
4. **digicertrootname** is the digital certificate name for the root (CA) certificate. The DIGICERT name is limited to 8 characters, and must be specified.
5. **digicertname** is the digital certificate name for the Connect:Direct End-User Server/Client Certificate. The DIGICERT name is limited to 8 characters, and must be specified.
6. **cdservcert** is the Connect:Direct End-User Server/Client Certificate Label. Label can be any name desired but using the Subject Common Name will provide a better understanding of the contents of the certificate. It is case sensitive and can be up to 237 characters
7. **rootcert** is a trusted root (or CA) certificate Label. It is case sensitive and can be up to 237 characters.
8. **commonname** is Certificate Common Name. The Subject Common Name of the End-User Server Certificate would be given to the Remote Connect:Direct Node if they wanted to do Client Authentication. It is required.
9. **organization unit** is Organizational Unit. It is an optional entry. (e.g. Support)
10. **Organization** is Organization. It is required. (e.g. IBM Corp)
11. **location** is City/Locality. It is an optional entry. (e.g. Irving)
12. **state** is State or Province. It is an optional entry. (e.g. TX)
13. **country** is a 2-character Country or Region identifier. It is required. (e.g. US)
14. **begindate** is Certificate start or beginning date in the format mm/dd/yy.
15. **expiredate** is Certificate end or expiration date in the format mm/dd/yy.

List Connect:Direct Keyring and Certificate

Create Keyring

```
TSS ADD (cdid) KEYRING(keyringname) -  
    LABLRING('keyringlabel')
```

NOTE: The **KEYRING** name is limited to 8 characters and must be specified. The **LABLRING** is an alternate name, case sensitive and can be up to 237 characters. Connect:Direct uses the name in the **LABLRING** to call

the Keyring. The **LABLRING** name must be in **Certificate Pathname** field on the **SSL/TLS Parameters** screen in the Local record of the Connect:Direct Secure+ PARMFILE when using CA-Top Secret.

List all keyrings for an acid (Userid)

```
TSS LIST(acid) KEYRING(ALL)
```

List specific keyring for an ACID

```
TSS LIST(acid) KEYRING(8-byte keyring name)
```

List the Connect:Direct digital certificate by certificate name (DIGICERT) or label (LABLCERT)

```
TSS LIST(cdid) {LABLCERT('cdservcert')} |  
                {DIGICERT(digicertname)}
```

List a root (CA) digital certificate by certificate name (DIGICERT) or label (LABLCERT)

```
TSS LIST(CERTAUTH) {LABLCERT('rootcert')} |  
                   {DIGICERT(digicertrootname)}
```

Generate a Root Certificate Authority (CA) Certificate

```
TSS GENCERT(CERTAUTH) DIGICERT(digicertrootname) -  
    SUBJECTN('CN="commonname" -  
              O="organization" -  
              C="country" -  
              OU="organization unit"') -  
    NADATE(expiredate) -  
    KEYUSAGE('CERTSIGN') -  
    LABLCERT('rootcert')
```

NOTE: As with the keyring name, in CA-Top Secret there are two names; **DIGICERT** name is limited to 8 characters and must be specified, and a **LABLCERT** name that is mixed case up to 237 characters.

Export a Root Certificate Authority (CA) Certificate

```
TSS EXPORT(CERTAUTH) {DIGICERT(NNNNN)} |  
                    {LABLCERT(LABEL NAME)} -  
                    DCDSN(&hlq.output.dataset.name) -  
                    FORMAT(CERTB64)
```

Export Certificate Format

CERTB64 - Specifies a DER encoded X.509 certificate that has been encoded using Base64. This is a text file so it can be ship in an e-mail. If it being transferred using FTP or Connect:Direct, TEXT or ASCII mode must be used.

CERTDER - Specifies a DER encoded X.509 certificate. It is a binary file, so if it being transferred using FTP or Connect:Direct, BINARY mode must be used.

PKCS12B64 - Specifies a DER encoded PKCS#12 package that has been encoded using Base64. A PKCS12 PASSWORD must also be supplied. Export the certificate and the private key (which must exist and must not be an ICSF key). The package produced by specifying one of the PKCS #12 keywords is encrypted using the password specified according to the PKCS #12 standard. Processing will attempt to package any certificate-

authority certificate necessary to complete the basing chain to the exported certificate. This is a text file so it can be ship in an e-mail. If it being transferred using FTP or Connect:Direct, TEXT or ASCII mode must be used.

PKCS12DER - Specifies a DER encoded PKCS#12 package. A PKCS12 PASSWORD must also be supplied. Export the certificate and the private key (which must exist and must not be an ICSF key). The package produced by specifying one of the PKCS #12 keywords is encrypted using the password specified according to the PKCS #12 standard. Processing will attempt to package any certificate-authority certificate necessary to complete the basing chain to the exported certificate. It is a binary file, so if it being transferred using FTP or Connect:Direct, BINARY mode must be used.

NOTE: **PKCS12B64** is the default if PASSWORD is specified; otherwise, **CERTB64** is the default.

NOTE: The File Format of an Export is determined by the support on the receiving system.

Import a Certificate with or without Private Key

```
TSS IMPORT(uid) {DIGICERT(name)} |
                {LABLCERT(label name)} -
                PASSWORD(xxxxxxxx) -
                DCDSN(&hlq.input.dataset.name) -
                FORMAT(PKCS12DER)

TSS IMPORT(uid) {DIGICERT(name)} |
                {LABLCERT(label name)} -
                DCDSN(&hlq.input.dataset.name) -
                FORMAT(CERTB64)
```

(uid is either the Connect:Direct **ACID** or **CERTAUTH**)

Import Certificate Format

CERTB64 - Specifies a DER encoded X.509 certificate that has been encoded using Base64. This is a text file so it can be ship in an e-mail. If being transferred using FTP or C:D, TEXT or ASCII mode must be used.

CERTDER - Specifies a DER encoded X.509 certificate. It is a binary file, so if being transferred using FTP or C:D, BINARY mode must be used.

PKCS12B64 - Specifies a DER encoded PKCS#12 package that has been encoded using Base64. A PKCS12 PASSWORD must also be supplied. Export the certificate and the private key (which must exist and must not be an ICSF key). The package produced by specifying one of the PKCS #12 keywords is encrypted using the password specified according to the PKCS #12 standard. Processing will attempt to package any certificate-authority certificate necessary to complete the basing chain to the exported certificate. This is a text file so it can be ship in an e-mail. If being transferred using FTP or Connect:Direct, TEXT or ASCII mode must be used.

PKCS12DER - Specifies a DER encoded PKCS#12 package. A PKCS12 PASSWORD must also be supplied. Export the certificate and the private key (which must exist and must not be an ICSF key). The package produced by specifying one of the PKCS #12 keywords is encrypted using the password specified according to the PKCS #12 standard. Processing will attempt to package any certificate-authority certificate necessary to complete the

basing chain to the exported certificate. It is binary, so if transferred using FTP or Connect:Direct, BINARY must be used.

NOTE: **PKCS12B64** is the default if **PASSWORD** is specified; otherwise, **CERTB64** is the default.

NOTE: The File Format of an Import is determined by the support on the sending system.

Generate a Signed Certificate

```
TSS GENCERT(cdid) DIGICERT(digicertname) -
    SUBJECTN('CN="commonname" -
              O="organization" -
              C="country" -
              OU="organization unit"') -
    NADATE(expiredate) -
    KEYSIZE(1024) -
    TRUST -
    KEYUSAGE('HANDSHAKE') -
    LABLCERT('cdservcert') -
    SIGNWITH(CERTAUTH,digicertrootname)
```

NOTE: As with the keyring name, in CA-Top Secret there are two names; **DIGICERT** name is limited to 8 characters and must be specified, and a **LABLCERT** name that is mixed case up to 237 characters. The **LABLCERT** name is used in the Local record of the Connect:Direct Secure+ PARMFILE **Certificate Label** field.

The name of the root CA certificate referenced on **SIGNWITH** in this command is the 8-character **DIGICERT** name, not the **LABLCERT** name.

Connect a Connect:Direct's End-User Server/Client Certificate to a Keyring

```
TSS ADD(cdid) KEYRING(keyringname) -
    RINGDATA(cdid,digicertname) -
    TRUST -
    DEFAULT -
    USAGE(PERSONAL)
```

Connect Root Certificate Authority (CA) Certificate to a Keyring

```
TSS ADD(cdid) KEYRING(keyringname) -
    RINGDATA(CERTAUTH,digicertrootname) -
    TRUST -
    USAGE(CERTAUTH)
```

NOTE: **RINGDATA** specifies the certificate owner and certificate **DIGICERT** name.

Connect:Direct Needs to Share a SITE Certificate

CERTSITE is an ACID in which your installation can maintain site-generated certificates; it is similar to **SITE** certificates in RACF. The **CERTSITE** ACID would hold certificates that can be share amongst user ACIDs. If an ACID (userid) wants to add a **CERTSITE** certificate, they must have **CONTROL** authority to use it.

Troubleshooting

Most Secure+ issues are either issues with the configuration of the Secure+ PARMFILE or issues dealing with the certificates. Most issues with SSL/TLS, after the initial setup of the Secure+ PARMFILE, will deal with certificates. Some issues are rather simple to resolve, but some will require further diagnosis.

Connect:Direct does NOT authenticate certificates; this is done by System SSL (z/OS) or IBM GSKit (Windows or Unix). Any Secure+ error message that is contained in the CSPA202E reason is an error that has been returned from System SSL or IBM GSKit and will often require tracing to be done to diagnose. On C:D z/OS, this involves taking a System SSL Trace, which will often need to be sent to SSL Support to be properly diagnosed.

If there is a question as to the validity of a certificate, a “loopback” (TCP version of PNODE-SNODE) can be done to test the local certificate.

There are other times where issues dealing with the Secure+ PARMFILE and/or certificate will require the Secure+ PARMFILE and Access file to be sent into Support for further diagnosis.

Therefore, there are five sections included here:

- 1. Common Secure+ Errors and Possible Solutions**
 - a. Saving the Secure+ PARMFILE
 - b. Initializing Connect:Direct with Secure+
 - c. SSL Authentication
 - d. FIPS Errors
- 2. Capturing a z/OS System SSL Trace**
- 3. Creating a Secure+ Loopback Node**
- 4. Sending in C:D z/OS Secure+ PARMFILE and Access File to Support**
- 5. Manually Create Secure+ VSAM Files (PARMFILE and Access File)**

Common Secure+ Errors and Possible Solutions

Saving the Secure+ PARMFILE

Receive “Invalid Option” when attempting to get into the Secure+ Admin Tool

- Running C:D z/OS 6.0, functional authority **SECURE+ ADMIN** (in the AUTH file) is not set to **Y** or **S#WNCR** (in BYTE08 of the Stage-2 Security Exit) is not set, meaning the userid does not have access to the Secure+ Admin Tool (from either the IUI or Control Center).

Certificate path cannot be null Secure.SSL.Cert.Trusted must be valued in the Local Node Secure.SSL.Cert.Trusted cannot default in the Local Node

- The keyring / key database has not been entered for the Certificate Pathname on the Local node.
- Certificate Pathname on the Local node contains an asterisk (*) instead of the correct keyring / key database.

Certificate label cannot default to itself in the Local Node Secure.SSL.Cert.Trusted cannot default in the Local Node

- An asterisk (*) is entered in the Certificate Label on the Local node instead of a valid certificate label.

Certificate Pass Phrase is not specified.

- If the **Certificate Pathname** is a key database, then the Pass Phrase is required; if this is a keyring, then this condition is ok (Pass Phrase cannot be used on a keyring).

RCDECYP - CANNOT DECRYPT OLD PARMFILE RECORD DECRYPT ERROR

- The keys that exist in your Access file are a longer length that the PARMFILE is expecting. While this typically will occur when running the conversion program (DGASCONV) converting a pre-C:D z/OS 5.2 to a C:D z/OS 5.2 PARMFILE, it can occur without the conversion taking place. The PARMFILE will need to be “re-keyed” (that is, make a new pass phrase for the PARMFILE) before the PARMFILE can successfully be converted.

For instructions on how to re-key the PARMFILE, refer to section **Making a New Pass Phrase for your Connect:Direct Secure+ PARMFILE** on page 93.

CSPA081E Unable to initialize workspace

CSPN107E TOOLKIT ERROR: CSPA081E: 0012/1302

RANDOM - CANNOT GENERATE RANDOM NUMBER, RC = 0000000C REASON = 00000000

- ICSF (Integrated Cryptographic Service Facility) is not active (ICSF is required for C:D z/OS 5.2 or later)
- The Connect:Direct userid for Secure+ and the TSO userid of the Secure+ administrator must have read access defined in the RACF CSFSERV facilities class (required for C:D z/OS 5.2 or later).
- For C:D z/OS 5.2, make sure APAR/PTF PI40982/UI27708 is applied.

CSPA605E Secure+ SSL Environment Refresh failed. rc=-00000097, rs=GSK_SC_NO_DATABASE_SPECIFIED

- The keyring or key database specified on the local node in the Secure+ PARMFILE is not valid.
- FIPS has been selected as an option in the Secure+ PARMFILE, but the keyring or key database has not been defined as FIPS.

Cipher suite cannot default to itself

- Cipher DEFAULT_TO_LOCAL_NODE (value **FF** or **FFFF**) has been selected on the Local node.

Initializing Connect:Direct with Secure+

SITA166I Secure+ SSL initialization failed. rc=000000416, rs=Permission denied.

SITA166I Secure+ SSL initialization failed. rc=000000416, rs=GSK_ERR_PERMISSION_DENIED.

- Userid used by Connect:Direct does not have permission to access the keyring
- Connect:Direct does not have the proper authority to the BPX.SERVER facility
- Connect:Direct userid does not have read access defined in the RACF CSFSERV facilities class
- A password is entered on **Certificate Pass Phrase** in the **Certificate Pathname** on the Local node entry in the PARMFILE, but a keyring is being used (Pass Phrase is invalid with a keyring).

SITA166I Secure+ SSL initialization failed. rc=000000202, rs=GSK_KEYRING_OPEN_ERROR .

- The OMVS dataset is migrated off, causing Connect:Direct to not find the keyring file (un-migrating the dataset and re-cycling Connect:Direct should correct this).
- Secure+ PARMFILE specified a nonexistent key database name.
- The Pass Phrase coded for the key database in the Local node in the Secure+ PARMFILE is incorrect.
- Initialization parameter SECURE.SSL.PATH.PREFIX is coded with a path, but a path is also entered for the Certificate Pathname in the local node in the Secure+ PARMFILE or the path is incorrect.

SITA166I Secure+ SSL initialization failed. rc=000000202, rs=Error detected while opening the certificate database.

- Passphrase is coded for keyring in the PARMFILE (passphrase is only valid for a key database) or is incorrect.
- Keyring / Key Database incorrect in the Local node
- Keyring / Key Database is on another system and cannot be accessed

SITA166I Database name has no value in SEC+

- The keyring or key database specified in the **Certificate Pathname** is not valid.
- SECURE.DSN is coded in the initialization parameters, but the Local node has OVERRIDE=N and all protocols set to N (should see SITA190I earlier in the C:D z/OS joblog).

CSPN100E SECURE+ PARM RECORD HASH VALIDATION FAILURE.

- C:D z/OS 5.2 or later is being started using a PARMFIL that has not been converted to 5.2 (or bringing up pre-C:D z/OS 5.2 with a C:D z/OS 5.2 or later PARMFIL).
- The PARMFIL and the Access File have become mismatched, you are bringing up the PARMFIL with the wrong Access File or 'Save Active' was done on a PARMFIL which is not the active PARMFIL.
- Initialization parameter SECURE.DSN is pointing to the wrong PARMFIL.

SITA190I Secure+ Initialization failed, Secure No Override No.

- The Local node entry has all Secure+ protocols set to N and Override is set to N, meaning that Secure+ is disabled on your C:D.
- This is a warning message; if Secure+ was not meant to be disabled, on the Local node entry set one or more Secure+ protocols to Y and/or change Override to Y.

SITA196E FIPS Mode Requested but SECURE.DSN is not specified

- The initialization parameters have **FIPS=YES** coded, but Secure+ has not been enabled by coding **SECURE.DSN** in the initialization parameters.

SSL Authentication

CSPA091E Secure+ session attempted with non-Secure+ node

- One side of the transmission is set as a Secure+ node and the other side is not.
- The Local node has Secure+ set to **No** and has **OVERRIDE=NO** coded (that is, Remote cannot override).

CSPA011E Illegal attempt to override Secure+ parameters

- The process has SECURE= parameter coded, but the remote entry for this node has OVERRIDE=NO.
- If doing a PULL, the PNODE (pulling side) has Data Encryption set to NO and the SNODE has Data Encryption set to YES. The PNODE and SNODE should both have Data Encryption set the same.

CSPA202E SSL handshake failure, reason=SSL protocol or certificate type is not supported

- Typically occurs when C:D z/OS 5.2 or later is attempting to use TLS 1.1 or 1.2, but the remote node does not support TLS 1.1 or 1.2.

CSPA001E Node definition for remote Secure+ node missing

- The PNODE is attempting to establish a Secure+ session with a remote node that is defined in the NETMAP but not defined in the Secure+ PARMFILE.
- The remote node is attempting to establish a Secure+ connection with your local node, but the remote node is not defined in the Secure+ PARMFILE.

CSPA202E SSL handshake failure, reason=GSK_ERR_BAD_V3_CIPHER

- Ciphers set on both sides may not match.
- The Cipher selected may not be installed in the operating system (z/OS SSL)
- If APAR OA47405 is applied on z/OS, ciphers 01-06 have been removed. Unselect these ciphers in your Secure+ PARMFILE.

CSPA202E SSL handshake failure, reason=GSK_ERROR_BAD_CERTIFICATE or GSK_ERROR_BAD_CERT

CSPA202E SSL handshake failure, reason=Certificate is not valid

CSPA202E SSL handshake failure, reason=Certificate validation error

CSPA202E SSL handshake failure, reason=GSK_ERR_CERT_VALIDATION

- If a new Certificate Authority was added to the key data base (.kdb) or a new certificate to Keyring, you either need to recycle the DTF to re-establish the valid Certificate Authorities or refresh the Secure+ environment in the C:D z/OS IUI (ADMIN;S - RF command).
- Certificate expiration date has been reached.
- The certificate is not validated on any local trusted CA certificate. This error is common if you use self-signed certificates because the remote Connect:Direct system does not have the CA certificate. Verify that each trading partner can validate the other's certificates and resubmit the process.
- RACLIST REFRESH has not been issued to refresh the changed definition in RACF system.
- Chained certificates do not have all the intermediate and the root certificates that make up the chain in their trusted root store or keyring/key database.
- Certificate was either built incorrectly (for example, the wrong format) or was somehow corrupted either when created or when exchanged with your trading partner.
- An alternate site certificate is being used, but the label is not entered on the remote node entry in the Secure+ PARMFILE. The default goes to the label in the local node, which returns the wrong certificate.
- The CA Root has been reissued for the certificate chain and the customer has not removed the old CA Root or the old CA Root is in the keyring ahead of the new one. Resolution is to remove the old one or reorder the certificates so that the new one is ahead of the old one in the keyring.
- The Local Keyring does not have an entry for the certificate authority from RSA Data Security. The Remote's certificate was not signed by the Verisign certificates in the Local's keyring. The Local side must add to their Keyring the Trusted Root certificate and possibly the intermediate Trusted Root certificate that was used to sign the Remote's certificate.
- Multiple G5 certificates are within the keyring, and the wrong certificate is trying to be validated.

CSPA010E Strong authentication connection failure

- The **Auth Timeout** value may be set too low in the node definitions for one or both nodes.
- There could be a problem with the certificate chain from the remote node. Reloading the certificate chain may resolve the problem.
- Possible network issues. Increase TCP.CONNECT.TIMEOUT timer, but also make sure HIPER maintenance is applied (if increasing TCP.CONNECT.TIMEOUT helps, it is probably a network issue).
- The remote node is behind an SSP (Secure Proxy) and the remote node is not responding.

CSPA202E SSL handshake failure, reason=Key label is not found

CSPA202E SSL handshake failure, reason=GSK_KEY_LABEL_NOT_FOUND

CSPA202E SSL handshake failure, reason=Key entry does not contain a private key

CSPA202E SSL handshake failure, reason=GSK_ERR_NO_PRIVATE_KEY

- This is typically caused by any one of the following five things:
 - ❖ Label coded on the **Certificate Label** on the local node in the PARMFILE is incorrect.
 - ❖ Keyring / Key Database name coded on the **Certificate Pathname** on the local node is incorrect.
 - ❖ An incorrect label was entered on the remote entry in the PARMFILE (a common mistake is putting the label of the remote certificate in the **Certificate Label** field on the remote entry in the PARMFILE; the **Certificate Label** field **ONLY** applies to your local certificate).
 - ❖ Either the local site certificate has been copied to the keyring / key database without the private key or an alternate site certificate is being used without the private key in the keyring.
 - ❖ A new certificate was added to the keyring but Connect:Direct was not refreshed nor recycled.
- Almost always, the problem here is the label (whether incorrect on the local or invalid on the remote) or the Pathname (keyring or key database name).

CSPA202E SSL handshake failure, reason=ICSF callable service returned an error

- Ciphers that use Elliptic Curve Algorithms (for example: TLS_ECDHE_ECDSA_WITH_RC4_128_SHA) were being attempted but not all prerequisites are fulfilled to support and use Elliptic Curve Algorithms.
- Elliptical ciphers are being attempted but access has not been granted to support the use of Elliptic Curve Algorithms. The userid associated with the DTF and the userid of the C:D Secure+ Administrator need to have read access to the RACF CSFSERV class resource profiles.

Note: For further information and/or help regarding how to setup and handle Elliptic Curve Algorithms on a z/OS system, either consult z/OS Cryptographic Services Integrated Cryptographic Service Facility (ICSF) documentation or contact ICSF Support to determine what is needed to provide the correct access authority.

- If ciphers using Elliptic Curve Algorithms are not needed, use a non-elliptical cipher (that is, a cipher without ECDHE / ECDSA in the name).

CSPA202E SSL handshake failure, reason=Certification authority unknown
CSPA202E SSL handshake failure, reason=Certification authority is unknown

- The authenticating node does not have the entire certificate chain (most often an intermediate certificate is missing).
- The authenticating node has not installed the entire certificate chain correctly.
- The Client (PNODE) does not have the correct CA certificates installed that will verify the Key Certificate being used by the Server (SNODE).

CSPA024E STS parameters used for SSL/TLS connection

- The process has SECURE= parameter with options SIG=Y|N and/or ENC=algorithm (or ENCRYPT.DATA=algorithm) - these are STS parameters

(NOTE: ENCRYPT.DATA=Y|N is an SSL/TLS option, but if an algorithm (such as DESCBC56, TDESCBC112, or IDEACBC128) is specified in the process, it becomes an STS option)

CSPA202E SSL or TLS handshake failure, reason=GSK_ERROR_SOCKET_CLOSED
CSPA202E SSL or TLS handshake failure, reason=Socket closed by remote partner

- The cipher(s) being used do not match on each side
- Incomplete certificate chain on the remote side
- Possible network issue

CSPA202E SSL handshake failure, reason=GSK_ERROR_OTHER_FATAL_ALERT / GSK_OK, Peer sent a fatal alert / Function completed with no error CSF callable service returned an error

- Certificate being used with Signature Algorithm MD2 or MD5, which is no longer supported.

A workaround for this on Windows or Unix is to go to CD_Install\ibm_jre\jre\lib\security (Windows) or C:\D_Install\jre\ibm-java-i386-70\jre\lib\security (Unix) and edit the **java.security** file.

Find the following line:

```
jdk.certpath.disabledAlgorithms=MD2, MD5, RSA keySize < 1024, \  
DSA keySize < 1024
```

Remove MD2 and/or MD5 (*jdk.certpath.disabledAlgorithms=RSA keySize < 1024, DSA keySize < 1024*) and save this file.

NOTE: If any maintenance is applied after making this change, MD2 and MD5 will again be disabled.

- Multiple G5 certificates are within the keyring with the same common name, and the wrong G5 certificate (MD2) is trying to be validated. Remove the MD2 G5 certificate from the keyring / key database.
- Remote is an older release of Connect:Direct that does not support SHA2 certificates.
- Certificate and/or intermediate(s) are not imported into the client keyring / key database / keystore.

**CSPA202E SSL handshake failure, reason=Signature algorithm not in signature algorithm pairs list
TLS/SSL Socket Init Error, 0x000001d3 (Signature algorithm not in signature algorithm pairs list).
ENDING TLS/SSL HANDSHAKE, RC=467**

- The certificate used is signed by a signature algorithm that is not supported by TLSv1.2 protocol. Acquire a new certificate that is using the new signature algorithm pair.
- The certificate used has been signed by one of the new signature algorithms for TLSv1.2 which is not in the in the signature algorithm pairs list.
- The TLSv1.2 protocol made the signature algorithm and the hash algorithm that are used for digital signatures an independent attribute. Previously the negotiated cipher suite determined these algorithms. System SSL has the infrastructure to support multiple signature algorithms; this is not the case with previous protocols, such as TLSv1.1 and TLSv1.0. Update your signature algorithm pairs list in the environment variable GSK_TLS_SIG_ALG_PAIRS with the new signature algorithm pairs.

For additional information on Signature Algorithm pairs please reference the z/OS Cryptographic Services System SSL Programming guide.

Unable to import certificate. Status 0x03353024 - Issuer certificate not found.

- The certificates are being imported “out of order” - for example, the intermediate certificate is being imported before the root certificate is imported.

FIPS Errors

SITA166I Secure+ SSL initialization failed. rc=000000445, rs=Key database is not a FIPS mode database

- The keyring or key database is not defined as FIPS mode and FIPS=YES is specified in the initialization parameters; this error typically occurs at startup.

CSPA605E Secure+ SSL Environment Refresh failed. rc=-00000097, rs=GSK_SC_NO_DATABASE_SPECIFIED

- The keyring or key database is not defined as FIPS mode and FIPS=YES is specified in the initialization parameters; this error typically occurs when Secure+ is refreshed.

CSPA202E TLS/SSL handshake failure, reason=Handle is not valid

- The keyring or key database is FIPS mode and FIPS is specified in the Secure+ PARMFILE, but FIPS=YES is not specified in the initialization parameters (Connect:Direct for z/OS).

Capturing a z/OS System SSL Trace

There are two different ways to take a z/OS SSL System Trace as needed by Connect:Direct:

Method 1: Using the **GSKSRVR** utility

Method 2: Does not require **GSKSRVR** (requires Connect:Direct to be recycled)

Customers can decide which method is best for them; SSL Support typically prefers **Method 1**, but if the customer does not have **GSKSRVR** available on their system, they must use **Method 2**.

Method 1: Using the GSKSRVR Utility

If you are diagnosing a problem with the keyring read by SSL, the trace **MUST** be started just **prior** to the channel initiator; otherwise, the required trace information will not be collected.

If you are diagnosing a problem with the certificate exchange during the channel handshake, the trace **MUST** be started just **prior** to the secure channel being started; otherwise, the required trace information will not be collected.

This method requires you have **GSKSRVR** available on your system.

NOTE: Before beginning this trace, make sure that the correct trace dataset is specified in the correct system PROCLIB. For example: SYS2.USER.PROCLIB(GSKWTR).

- Capture trace:

NOTE: This involves taking two traces, the second being to capture at CHANNEL START. The second trace should only be run if specifically requested by support to do so.

This will involve using the **GSKSRVR** CTRACE process to gather the trace and requires that the **GSKSRVR** started task be running:

1. S GSKSRVR
2. Start an external writer: /TRACE CT,WTRSTART=GSKWTR
3. Start the SSL trace: /TRACE CT,ON,COMP=GSKSRVR
4. /R xx,JOBNAME=(stcname),OPTIONS=(LEVEL=255),WTR=GSKWTR,END
 where stcname is the jobname of your Connect:Direct DTF
5. Either refresh Secure+ environment (ADMIN;S, option RF) or recycle Connect:Direct
6. Recreate the problem.
7. Stop traces and writers:
 /TRACE CT,OFF,COMP=GSKSRVR
 /TRACE CT,WTRSTOP=GSKWTR,FLUSH
8. /P GSKSRVR (if the next trace at CHANNEL START is not requested)

For further details about **GSKSRVR** and running **z/OS System SSL Trace**, refer to the **z/OS Cryptographic Services System Secure Sockets Layer Programming**, manual number SC24-7495-xx.

At this point, either copy the dataset from the above trace to another file or write this next trace to a different dataset.

This next trace will capture the error at channel start (**GSKSRVR** should still be started before stcname job):

1. S GSKSRVR (if it is not already started from the previous trace)
2. Start an external writer: /TRACE CT,WTRSTART=GSKWTR
3. Start the SSL trace: /TRACE CT,ON,COMP=GSKSRVR
4. /R xx,JOBNAME=(stcname),OPTIONS=(LEVEL=255),WTR=GSKWTR,END
where stcname is the jobname of your Connect:Direct DTF
5. Either refresh Secure+ environment (ADMIN;S, option RF) or recycle Connect:Direct
6. Stop traces and writers:
/TRACE CT,OFF,COMP=GSKSRVR
/TRACE CT,WTRSTOP=GSKWTR,FLUSH
7. /P GSKSRVR

- Format trace:

To format the trace in IPCS, ensure the load library **hlq.SIEALNKE** is in the STEPLIB concatenation.

Issue the IPCS command:

CTRACE COMP(GSKSRVR) LOCAL FULL

For further details, refer to the [z/OS Cryptographic Services System Secure Sockets Layer Programming](#), manual number SC24-7495-xx.

Method 2: Does not require GSKSRVR

This method requires access to USS (Unix System Services) and requires Connect:Direct to be recycled:

1. Create a PDS formatted with DCB=(RECFM=VB,LRECL=137,BLKSIZE=27998,DSORG=PO). This file will be used to change the USS Environment Parameters.
2. Add member to new PDS using the following parameter entries as a guide:
TZ=CST6CDT
LC_ALL=EN_US.IBM-037
GSK_TRACE_FILE=/u/somedirectory/gsk44a.txt
GSK_SSL_HW_DETECT_MESSAGE=1
GSK_SSL_ICSF_ERROR_MESSAGE=1
GSK_SSL_BSAFE_ERROR_MESSAGE=1
GSK_TRACE=255

NOTE: **somediretory** is a USS directory that the Connect:Direct address space has access to write.

These parameters mean:

TZ - Time Zone

LC_ALL= Language

GSK_TRACE_FILE is file that will contain GSK Trace data.

3. Add the //ENVIRON DD DISP=SHR,DSN=PDS created in **Step 1** (member created in **Step 2**) to the Connect:Direct Job. Recycle Connect:Direct.
 4. Using OE (OpenEdition) create a new file in UNIX directory accessible by Connect:Direct. The file needs to be empty.
- HINT:** Open file in Edit mode and add characters to one line. Close the file. Re-open the file in Edit mode and delete the line of characters added previously. Browse the file to verify that it is there (there should be no contents).
5. Make sure the Unix file created in step 4 is the name of the file in the GSK_TRACE_FILE in the PDS member in the **//ENVIRON DD DSN**
 6. Change permissions on the new file so that the Userid running Connect:Direct can read and write to the Unix file: **chmod 777 file.name**
 7. Start Connect:Direct with SSL. Run Test scenario. All GSK trace info will go to the trace file indicated in the **//ENVIRON DD**
 8. The trace data is in raw form. To format into text readable format use the following Unix gsk command and pipe output to another file.
For example: *gsktrace /u/somedirectory/gsk52a.txt > /u/somedirectory/gsk52a.for*
 9. Go back to OE (OpenEdit) and browse the contents.
 10. If sending the trace into Connect:Direct Support for review, send the formatted file.

Reference

Manual **z/OS Cryptographic Services System Secure Sockets Layer Programming**, *Obtaining Diagnostic Information*.

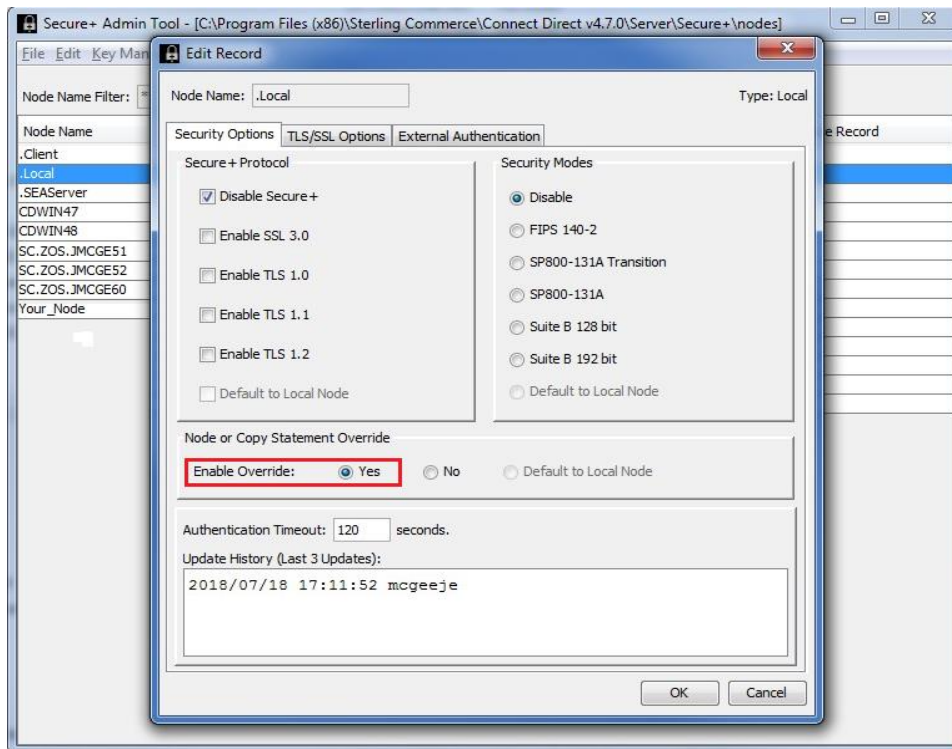
NOTE: Add CEEOUT DD SYSOUT=* to Connect:Direct started task for additional USS debug information.

Creating a Secure+ LOOPBACK Node

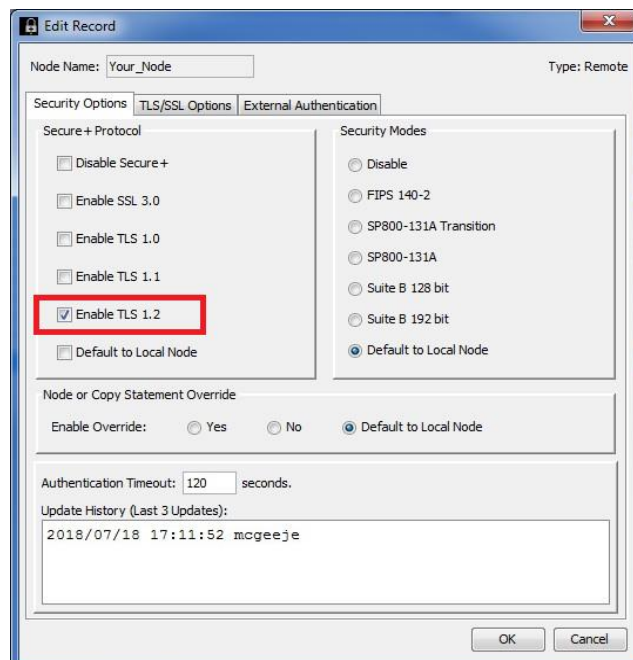
At times, there may be a question as to the validity of a certificate and it is either unclear whether the client or the server certificates are at issue or the remote node is not available for testing. In that case, a “loopback” (a node copying a file to itself) can be done on the local node to test and verify the local certificate.

C:D Windows/Unix Node

1. Make sure the .Local node has **Enable Override** set to **Yes**



2. Edit the remote entry for your local node and set to whatever Secure+ protocol that is needed.



3. Submit a process copying a file to yourself, similar to the following:

```
/*BEGIN_REQUESTER_COMMENTS
  $PNODE$="Your_Node" $PNODE_OS$="Windows"
  $SNODE$="Your_Node" $SNODE_OS$="Windows"
  $OPTIONS$="WDOS"
  END_REQUESTER_COMMENTS*/

COPYFILE PROCESS
  SNODE=Your_Node

STEP01 COPY
  FROM (
    FILE="C:\Processes\Test_file.txt"
  )
  TO (
    FILE="C:\Processes\Test_file_loop.txt"
    DISP=RPL
  )

PEND
```

4. Once the loopback test is complete, you should be able to look in the stats to determine if Secure+ was used. If Secure+ was used and the copy was successful, the certificate has been tested as good.
5. If the changes made to the remote node entry need to be removed, they can now be removed.

C:D z/OS Node

1. Add a LOOPBACK node entry to the NETMAP like the following:

```
$$INSERT
  ADJACENT.NODE=( (LOOPBACK,port,ip_addr,TCP) -
    PARSESS=(4,1) -
    ENVIRONMENT=ZOS -
    DESCRIPTION='NETMAP ENTRY FOR LOOPBACK (PNODE=SNODE) TESTING' -
  )
```

Make sure the IP address and port number are the ones being used by the local C:D. You should be able to obtain this information from the C:D JES MESSAGE LOG.

NOTE: Adding this LOOPBACK node allows you to test without changing the SECURE+ settings on the local node entry.

NOTE: If you do not want the comment (description), delete the line that begins **DESCRIPTION=**.

2. Be sure the Secure+ local node definition is defined as the following:

If the Local Node is defined with Secure+ disabled, Override=Yes must be coded.

If the Local Node is set with a Secure+ protocol defined, if Override=Yes is not coded, the protocol must match the protocol defined in the LOOPBACK remote definition.

3. Set up a Secure+ LOOPBACK remote node definition like the following:

Secure+ Create/Update Panel			
Option ==>			
Node Name: <u>LOOPBACK</u>		Type: <u>R</u>	(Local or Remote)

Security Options	EA Parameters	SSL/TLS Parameters	
---	--	---	

Secure+ Protocol:		Security Mode (Yes , No , Default to Local)	
Enable SSL	<u>N</u>	Enable FIPS	<u>N</u>
Enable TLS 1.0	<u>N</u>	Enable SP800-131a Transition	<u>N</u>
Enable TLS 1.1	<u>N</u>	Enable SP800-131a Strict	<u>N</u>
Enable TLS 1.2	<u>Y</u>	Enable NSA Suite B 128 bit	<u>N</u>
		Enable NSA Suite B 192 bit	<u>N</u>
Auth Timeout:	<u>120</u>	Enable Override	<u>Y</u>
Alias Names:	TCP Information:		
_____	IPAddr: _____		
_____	Port: _____		

		OK	Cancel
		--	---

Secure+ Create/Update Panel		<Change Pending>
Option ==>		
Node Name: <u>LOOPBACK</u>		Type: <u>R</u> (Local or Remote)

Security Options	EA Parameters	SSL/TLS Parameters
---	--	---

Enable Client Auth	<u>N</u>	(Yes , No , Default to Local)
Enable Data Encrypt	<u>Y</u>	
Certificate Label	*	
Cipher Suites	FFFF	
Certificate Pathname	*	
Certificate Common Name		

		OK Cancel
		-- ---

Make sure and save the PARMFILE to pick up the addition of LOOPBACK.

4. Submit a process using node LOOPBACK like the following:

```
LBTEST1  PROCESS  SNODE=LOOPBACK NOTIFY=&SYSUID
STEP1    COPY    FROM (PNODE DSN=SOURCE.TEST.FILE -
                        DISP=SHR) -
                        TO (SNODE DSN=DEST.TEST.FILE -
                        DISP=RPL)
```

5. Check the statistics to make sure Secure+ was used.

If the process ran successfully, look at the statistics (CT stat record) and make sure that Secure+ was used.

NOTE: Once you have completed the loopback test, the LOOPBACK node that was added can be removed from the PARMFILE and NETMAP.

Sending in C:D z/OS Secure+ PARMFILE and Access File to Support

Some Secure+ issues may require the C:D z/OS Secure+ PARMFILE and Access File to be sent in to Connect:Direct Support for analysis.

NOTE: This can also be used to move your Secure+ PARMFILE and Access File to another system. Create the new VSAM files on the new system, then copy the flat files to that system and REPRO back into the VSAM files.

Do a REPRO on your Secure+ PARMFILE and ACCESS file to flat files by using the following sample JCL – simply change the file names and VOL=SER, as well as the job card (if needed):

```
//REPRO1IR JOB (CDLEV1), 'IDCAMS JOB',  
//          REGION=0M, CLASS=A, MSGCLASS=X, NOTIFY=&SYSUID  
//REPRO1    EXEC PGM=IDCAMS  
//SYSPRINT  DD SYSOUT=*  
//SYSUT1    DD DSN=CD.SECURE.PARMFILE,  
//          DISP=SHR  
//SYSUT2    DD DSN=CD.SECURE.FLAT.PARMFILE,  
//          DISP=(NEW,CATLG, ), UNIT=SYSDA, VOL=SER=123456,  
//          LRECL=3072, BLKSIZE=32768, RECFM=VB, DSORG=PS,  
//          SPACE=(CYL,(10,2),RLSE)  
//SYSIN     DD *  
          REPRO INFILE(SYSUT1) OUTFILE(SYSUT2)  
/*  
//REPRO2    EXEC PGM=IDCAMS  
//SYSPRINT  DD SYSOUT=*  
//SYSUT1    DD DSN=CD.SECURE.ACCFILE,  
//          DISP=SHR  
//SYSUT2    DD DSN=CD.SECURE.FLAT.ACCFILE,  
//          DISP=(NEW,CATLG, ), UNIT=SYSDA, VOL=SER=123456,  
//          LRECL=3072, BLKSIZE=32768, RECFM=VB, DSORG=PS,  
//          SPACE=(CYL,(10,2),RLSE)  
//SYSIN     DD *  
          REPRO INFILE(SYSUT1) OUTFILE(SYSUT2)  
/*  
//
```

Once you have the flat files, terse these and send them as binary files.

Also, Support needs to know the exact name of the PARMFILE and Access File to rebuild and test them.

To reload the flat file into the VSAM files, run a job like the following for each of the PARMFILE and Access file:

```
//REPRO2IR JOB (CDLEV1), 'IDCAMS JOB',  
//          REGION=0M, CLASS=A, MSGCLASS=X, NOTIFY=&SYSUID  
//REPRO     EXEC PGM=IDCAMS  
//SYSPRINT  DD SYSOUT=*  
//SYSUT1    DD DISP=SHR, DSN=CD.SECURE.FLAT.PARMFILE  
//SYSUT2    DD DISP=SHR, DSN=CD.SECURE.PARMFILE  
//SYSIN     DD *  
          REPRO INFILE(SYSUT1) OUTFILE(SYSUT2)  
/*  
//
```

Manually Create Secure+ VSAM Files (PARMFILE and Access File)

The VSAM job to manually create the Secure+ PARMFILE and Access file, if needed:

```

/*****
/*      SECURE+ PARMFILE & ACCFILE CLUSTER DEFINITIONS      */
/*****
DELETE (CD.SECURE.PARMFILE) CLUSTER
DELETE (CD.SECURE.ACCFILE) CLUSTER
DELETE (CD.SECURE.PARMFILE.TEMP) CLUSTER
DELETE (CD.SECURE.ACCFILE.TEMP) CLUSTER
IF MAXCC=8 THEN SET MAXCC=0
/*****
/* DEFINE THE PARMFILE                                     */
/*****
DEFINE CLUSTER
  (NAME (CD.SECURE.PARMFILE)
  RECORDS (12 4)
  VOLUMES (xxxxxx)
  INDEXED
  FREESPACE (0 0)
  NOIMBED
  KEYS (18 2)
  RECORDSIZE (1292 4086)
  NOREPLICATE
  SHAREOPTIONS (3 3))
DATA
  (CONTROLINTERVALSIZE (4096)
  NAME (CD.SECURE.PARMFILE.DATA))
INDEX
  (CONTROLINTERVALSIZE (512)
  NAME (CD.SECURE.PARMFILE.INDEX))
/*****
/* DEFINE THE ACCESS FILE                                   */
/*****
DEFINE CLUSTER
  (NAME (CD.SECURE.ACCFILE)
  RECORDS (12 4)
  VOLUMES (xxxxxx)
  INDEXED
  FREESPACE (0 0)
  NOIMBED
  KEYS (18 2)
  RECORDSIZE (1292 4086)
  NOREPLICATE
  SHAREOPTIONS (3 3))
DATA
  (CONTROLINTERVALSIZE (4096)
  NAME (CD.SECURE.ACCFILE.DATA))
INDEX
  (CONTROLINTERVALSIZE (512)
  NAME (CD.SECURE.ACCFILE.INDEX))
```