



How to setup SSL with VSE

Java-based connector
VSE Navigator
TLS and HTTPD

Last formatted on: Friday, November 19, 2010

Joerg Schmidbauer
jschmidb@de.ibm.com

Dept. 3229
VSE Development
IBM Lab Böblingen
Schönaicherstr. 220

D-71032 Böblingen
Germany



Disclaimer

This publication is intended to help VSE system programmers setting up infrastructure for their operating environment. The information contained in this document has not been submitted to any formal IBM test and is distributed AS IS. The information about non-IBM ("vendor") products in this manual has been supplied by the vendor and IBM assumes no responsibility for its accuracy or completeness. The use of this information or the implementation of any of these techniques is a customer responsibility and depends on the customer's ability to evaluate and integrate them into the customer's operational environment. While each item may have been reviewed by IBM for accuracy in a specific situation, there is no guarantee that the same or similar results will be obtained elsewhere. Customers attempting to adapt these techniques to their own environments do so at their own risk. Any pointers in this publication to external Web sites are provided for convenience only and do not in any manner serve as an endorsement of these Web sites.

Any performance data contained in this document was determined in a controlled environment, and therefore, the results that may be obtained in other operating environments may vary significantly. Users of this document should verify the applicable data for their specific environment. Reference to PTF numbers that have not been released through the normal distribution process does not imply general availability. The purpose of including these reference numbers is to alert IBM customers to specific information relative to the implementation of the PTF when it becomes available to each customer according to the normal IBM PTF distribution process.

The following terms are trademarks of other companies:

Firefox and the Firefox Logos are trademarks of the Mozilla Foundation.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and/or other countries.

Microsoft, Windows, Windows XP, and the Windows logo are trademarks of Microsoft Corporation in the United States and/or other countries.

Contents

1	Introduction	4
2	SSL setup for Java-based connector	4
2.1	Generate the server key and certificates	5
2.1.1	Define the properties of your VSE system.....	5
2.1.2	Create the VSE key and certificates.....	7
2.2	Setting up VSE Connector Server for SSL.....	12
2.2.1	Setting up SKVCSCFG	12
2.2.2	Setting up SKVCSSSL for server authentication.....	12
2.3	Setting up VSE Navigator for SSL.....	14
2.4	Connecting to VSE.....	16
2.5	Restrictions with client authentication	18
2.6	Setting up for SSL client authentication.....	19
2.7	SSL client authentication with implicit logon	21
2.8	Using encryption with AES-256	24
3	SSL setup for TLS	26
3.1	Setup SSL native mode with HTTPD	26
3.2	Considerations on \$WEB user	27
3.3	Connect to the HTTPD using a Web Browser	28
3.4	Configure Ciphers in MS Internet Explorer	29
3.5	Configure Ciphers in Mozilla Firefox	29
3.6	Displaying SSL properties in Mozilla Firefox	30
4	Hardware- versus software-based encryption	31
4.1	Suppressing the CPU Assist feature	31
4.2	Forcing the CPU assist feature	32
4.3	Suppressing the use of crypto cards	32
4.4	Displaying available crypto hardware	33
5	Debugging SSL/TLS connections	34
5.1	Tracing on VSE.....	34
5.2	Tracing in Java	34
6	More information.....	34

Changes:

May 2009 – updated info on how to configure SSL cipher suites in Web browsers, see page 29.

Aug 2009 – some textual corrections

Dec 2009 – added new sections on SSL client authentication

Nov 2010 – added chapter 5 “Debugging SSL/TLS connections” and section 3.6

1 Introduction

This paper describes the setup of SSL in various scenarios with VSE acting as server. This involves the creation of RSA key pairs and digital certificates on the server and on the client side. For simplification, we do not purchase certificates from official Certificate Authorities (CAs), but create our own set of so called self signed certificates. Self-signed certificates are not signed by an official CA and therefore work only in a closed test environment.

The following software has been used in the test setup.

- z/VSE 4.1
- TCP/IP for VSE/ESA 1.5E as part of z/VSE 4.1
- VSE Connector Server as part of z/VSE 4.1 (job STARTVCS)
- Java 1.6.0_05 from Sun Microsystems
- Keyman/VSE, update from 08/2007
- *Dec 2009*: Keyman/VSE, update from 07/2009 with support for JKS keystores
- Mozilla Firefox version 2.0.0.6
- Microsoft Internet Explorer 6.0

2 SSL setup for Java-based connector

Currently, VSE supports SSL through various system components. Many of the different setup tasks are described in VSE manuals. In this paper we concentrate on the bottom line and aspects not covered by the official books. This includes the way of importing certificates into Web Browsers for use with CICS Web Support or TLS/D.

The Java-based connector supports SSL connections from any Java program running outside of VSE to the VSE Connector Server running on VSE. A detailed description of setting up SSL for use with the Java-based connector is contained in the *z/VSE V4R1 e-business Connectors User's Guide*, which is available online at

<http://www.ibm.com/servers/eserver/zseries/zvse/documentation/#conn>

Hereby, Java programs can be

- *Java applications*, like the VSE Navigator program, which is provided by IBM without warranty and can be downloaded for free from the VSE Homepage at

<http://www.ibm.com/servers/eserver/zseries/zvse/downloads/>

- *Java applets* or *servlets*, running in a Web Application Server environment, like IBM WebSphere. A detailed description of how to setup SSL from a Java servlet running in WebSphere to the VSE Connector Server is contained in Redbook *WebSphere V5 for Linux on zSeries Connectivity Handbook (SG24-7042)* that is available online at

<http://www.redbooks.ibm.com/abstracts/sg247042.html?Open>

The following sections describe the basic SSL setup for the Java-based connector with server and client authentication.

2.1 Generate the server key and certificates

The easiest way to generate all necessary keys and certificates for the VSE server side is by using the Keyman/VSE utility which is provided by IBM without warranty for free download from

<http://www.ibm.com/servers/eserver/zseries/zvse/downloads/>

Keyman/VSE is a Java application, which is typically installed on a Personal Computer. It has the following prerequisites.

- Java 1.4 or higher on the workstation side
- VSE Connector Client on the workstation side
- TCP/IP for VSE/ESA 1.5E on the VSE side
- VSE Connector Server up and running in non-SSL mode on the VSE side

Although Keyman/VSE provides many functions for manually creating keys and certificates, signing certificate requests, and so on, the easiest way for creating the necessary files on VSE is using the Wizard dialog for creating a self-signed keyring. For details about the individual Keyman/VSE functions refer to the HTML-based help of the Keyman tool.

Our first step is to start Keyman/VSE and entering the properties of your VSE system. This information is needed later for sending created keys and certificates to VSE.

2.1.1 Define the properties of your VSE system

On the main window click on the **VSE host properties** toolbar button.



On the **VSE Host – Properties** dialog box enter the required information for your VSE system. Press the **New** button to create a new VSE host definition.

VSE Host - Properties

Name	SAMPLE	New
IP Address		Add
Port	2893	Delete
VSE User	SYSA	Change
VSE Job Class	A	
VSE Password		
VSE Crypto Library	CRYPTO	KEYRING
Cert. Member Name	TEST01	PRVK / CERT / ROOT
Cert. Mapping Member	BSSDCUID	MAPPING
TCP/IP Library	PRD1	BASE
TCP/IP System ID	00	

OK Cancel Help

Then enter a unique name for your VSE system, its IP address, the port number of the VSE Connector Server, a VSE user ID together with its password and so on.

VSE Host - Properties

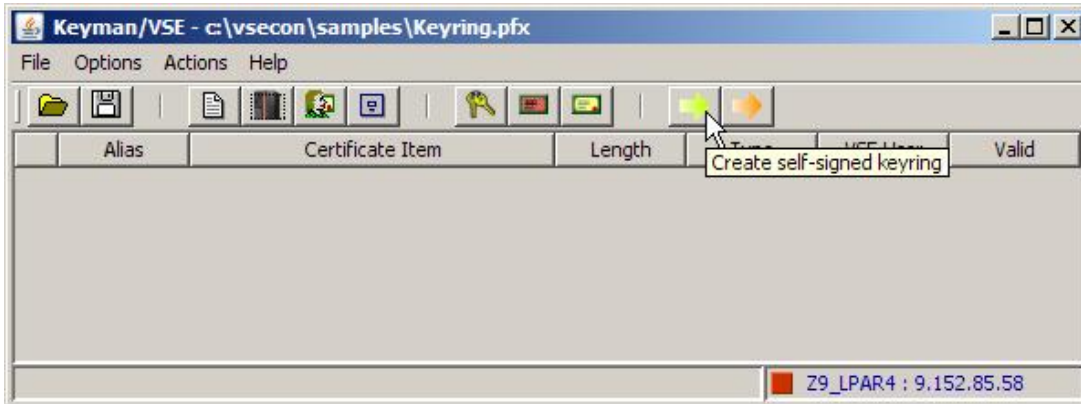
Name	z9_LPAR4	New
IP Address	9.152.85.58	Add
Port	2893	Delete
VSE User	JSCH	Change
VSE Job Class	A	
VSE Password	*****	*****
VSE Crypto Library	CRYPTO	KEYRING
Cert. Member Name	SSL01	PRVK / CERT / ROOT
Cert. Mapping Member	BSSDCUID	MAPPING
TCP/IP Library	PRD1	BASE
TCP/IP System ID	00	

OK Cancel Help

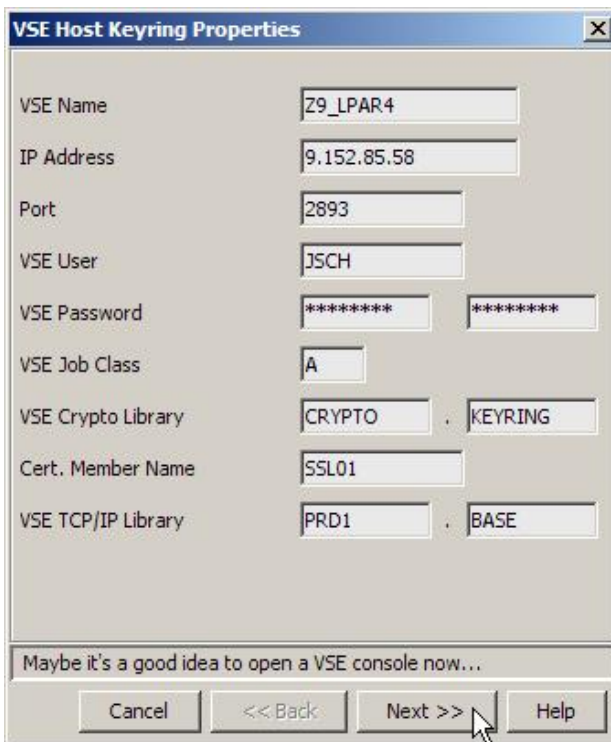
Then press the **Add** button to add the new definition. We are now ready to create the VSE server key and the necessary certificates.

2.1.2 Create the VSE key and certificates

Click on the **Create self-signed keyring** toolbar button.

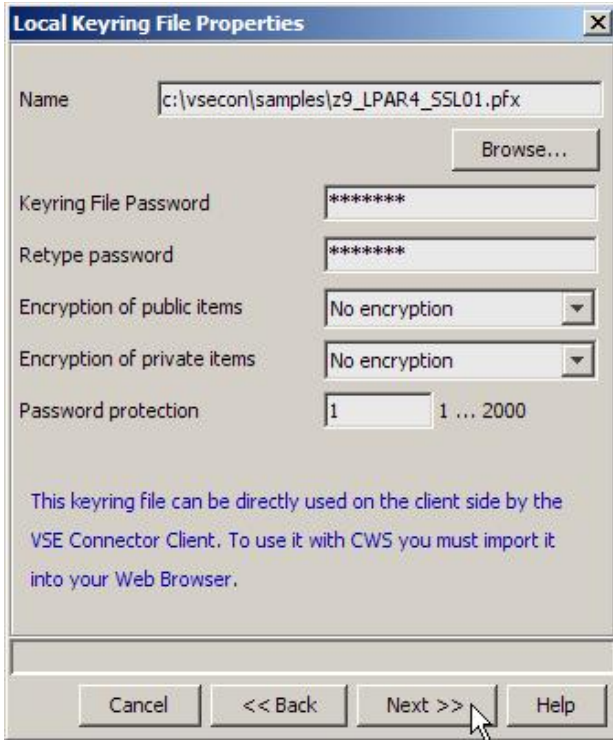


Fill in the required information on the next dialog box



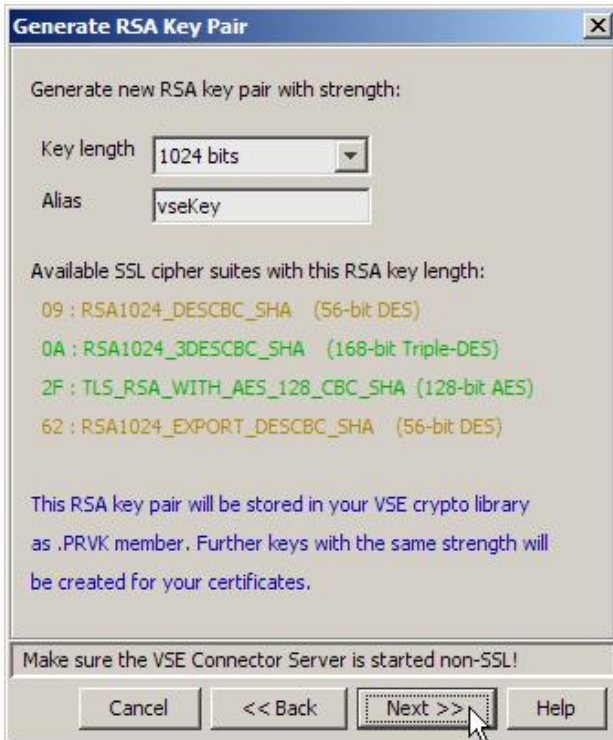
Press **Next**.

On the next dialog specify a password which is used for protecting the local keyring file. You should leave the settings for the encryption of public and private items on **No encryption**. Otherwise there might be problems dependent on your Java runtime when reading the file afterwards.



Press **Next**.

On the next dialog box specify the key length of your server key and a unique alias string to identify the key. The box shows you a list of available cipher suites with the selected RSA key length. This association has been removed with TCP/IP fix ZP15E250; refer to the notes below Table 1 on page 18.



Press **Next**. On the following dialog box specify the personal information for the VSE ROOT certificate.

Personal Information for VSE ROOT Certificate

Common name: VSE ROOT Certificate

Organizational unit: Development

Organization: IBM Germany

City/Location: Boeblingen

State/Province: N/A

Country: DE Germany (DE)

e-mail: zvse@de.ibm.com

Expires: 2009-4-25 1 year

Alias: rootCert

This certificate will be cataloged on VSE as .ROOT member in the VSE keyring library.

New 1024-bit Key generated, elapsed time: 0 second(s).

Cancel << Back Next >> Help

Press **Next**.

On the following dialog box specify the personal information for the VSE server certificate.

Personal Information for VSE Server Certificate

Common name: VSE Server Certificate

Organizational unit: Development

Organization: Your organization

City/Location: Your city/location

State/Province: Your state/province

Country: DE Germany (DE)

e-mail: info@your.company.com

Expires: 2009-4-25 1 year

Alias: vseCert

This certificate will be cataloged on VSE as .CERT member in the VSE keyring library.

New 1024-bit ROOT certificate generated.

Cancel << Back Next >> Help

Press **Next**. A client certificate is only needed for Client Authentication.

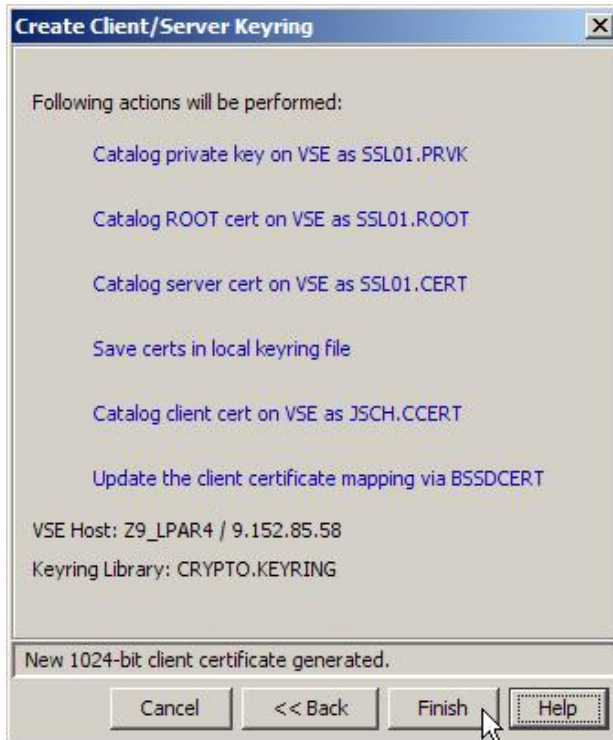


The screenshot shows a dialog box titled "Personal Information for VSE Client Certificate". It contains several input fields for certificate details:

- Common name: VSE Client Certificate
- Organizational unit: Your company
- Organization: Your organization
- City/Location: Your location
- State/Province: Your state/province
- Country: DE (Germany (DE))
- e-mail: vseclient@your.company.com
- Expires: 2009-4-25, 1 year
- Map to VSE User: JSCH (Optional)
- Alias: clientCert

At the bottom, a status bar indicates "New 1024-bit server certificate generated." and there are buttons for "Cancel", "<< Back", "Next >>", and "Help". A mouse cursor is pointing at the "Next >>" button.

Press **Next**.



The screenshot shows a dialog box titled "Create Client/Server Keyring". It lists the actions that will be performed:

- Catalog private key on VSE as SSL01.PRVK
- Catalog ROOT cert on VSE as SSL01.ROOT
- Catalog server cert on VSE as SSL01.CERT
- Save certs in local keyring file
- Catalog client cert on VSE as JSCH.CCERT
- Update the client certificate mapping via BSSDCERT

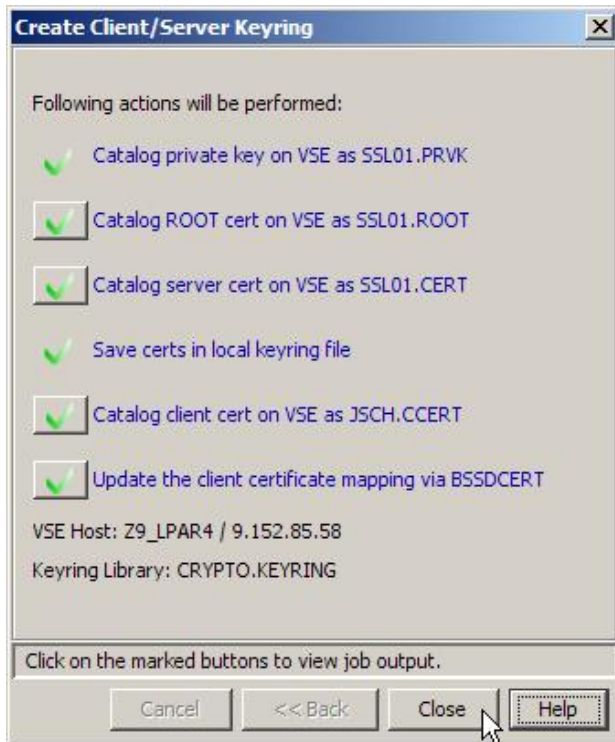
Additional information provided:

- VSE Host: Z9_LPAR4 / 9.152.85.58
- Keyring Library: CRYPTO.KEYRING

At the bottom, a status bar indicates "New 1024-bit client certificate generated." and there are buttons for "Cancel", "<< Back", "Finish", and "Help". A mouse cursor is pointing at the "Finish" button.

Press **Finish**.

Make sure that the VSE Connector Server is started in non-SSL mode on the VSE side.



Press **Close**.

Now you have three VSE library members cataloged into CRYPTO.KEYRING. The PRVK member contains the RSA key pair, the ROOT member contains the self-signed VSE ROOT certificate, and the CERT member contains the VSE server certificate. The two certificates are also saved into the local keyring file.

```
LD SSL01.*
```

MEMBER		CREATION	LAST	BYTES	LIBR	CONT	SVA	A-	R-
NAME	TYPE	DATE	UPDATE	RECORDS	BLKS	STOR	ELIG	MODE	
SSL01	CERT	08-04-25	- -	724 B	1	YES	-	-	-
SSL01	PRVK	08-04-25	- -	2048 B	3	YES	-	-	-
SSL01	ROOT	08-04-25	- -	686 B	1	YES	-	-	-

```
L113I RETURN CODE OF LISTDIR IS 0
L001A ENTER COMMAND OR END
```

You can also close the Keyman/VSE tool now. As we don't need the server key on the client side, the key was not saved to the local file.

2.2 Setting up VSE Connector Server for SSL

On the VSE side, you need following ICCF members, which are provided in ICCF library 59, for setting up SSL for the VSE Connector Server.

- SKVCSCFG
- SKVCSSSL
- SKVCSCAT

2.2.1 Setting up SKVCSCFG

SKVCSCFG is the template for the main configuration member of the VSE Connector Server. Here, we specify whether SSL shall be used or not.

Note: the VSE Connector Server can either be started in SSL mode or in non-SSL mode. If you want to have SSL and non-SSL clients connecting to VSE at the same time, you would start two instances of the VSE Connector Server in two different partitions, one providing SSL, the other one providing non SSL connections.

```

; *****
; TCP/IP - SERVER SPECIFIC CONFIGURATIONS
; - SERVERPORT : THE TCP PORT WHERE THE SERVER IS LISTENING
; - MAXCLIENTS : THE MAXIMUM NUMBER OF CONCURRENT CLIENTS
; - SSLENABLE  : YES/NO - USE SECURE SOCKET LAYER
; *****
SERVERPORT   = 2893
MAXCLIENTS  = 256
SSLENABLE    = YES

```

Change the SSLENABLE parameter to YES.

2.2.2 Setting up SKVCSSSL for server authentication

SKVCSSSL is the template for defining all SSL related parameters. The most important parameter, which changes with each setup, is the name of the VSE keyring, which consists of three VSE library members with file type PRVK (the private key file), ROOT (the root certificate), and CERT (the server certificate).

```

SSLVERSION   = SSL30
KEYRING      = CRYPTO.KEYRING
CERTNAME     = SSL01
SESSIONTIMEOUT = 86440
AUTHENTICATION = SERVER

```

Here, change the CERTNAME parameter to the name we used when uploading the members to VSE. In our first step we allow all available cipher suites to be used.

Parameter AUTHENTICATION is set to SERVER for SSL server authentication. For Client Authentication refer to Restrictions with client authentication on page 18.

```

; *****
; CIPHERSUITES SPECIFIES A LIST OF CIPHER SUITES THAT ARE ALLOWED
; *****
CIPHERSUITES = ; COMMA SEPARATED LIST OF NUMERIC VALUES
              01, ; RSA512_NULL_MD5
              02, ; RSA512_NULL_SHA
              08, ; RSA512_DES40CBC_SHA
              09, ; RSA1024_DESCBC_SHA
              0A, ; RSA1024_3DESCBC_SHA
              62, ; RSA1024_EXPORT_DESCBC_SHA
              2F, ; TLS_RSA_WITH_AES_128_CBC_SHA
              35 ; TLS_RSA_WITH_AES_256_CBC_SHA

```

Now we have to catalog our changes to refresh the VSE library members, which are read by the VSE Connector Server at startup. This is done with job template SKVCSCAT.

```

* $$ JOB JNM=VCSCAT,DISP=D,CLASS=0
// JOB VCSCAT CATALOG VCS CONFIGURATION MEMBERS
// EXEC LIBR,PARM='MSHP'
ACCESS S=PRD2.CONFIG
CATALOG IESVCSRV.Z REPLACE=Y
* $$ SLI ICCF=(SKVCSCFG),LIB=(12)
/+
CATALOG IESSSLCF.Z REPLACE=Y
* $$ SLI ICCF=(SKVCSSSL),LIB=(12)
/+
/*
/&
* $$ EOJ

```

Because we only changed the main configuration member and the SSL configuration member, we only need to catalog these two changes.

After submitting the above job, restart the VSE Connector server.

```

R1 0045 // JOB STARTVCS START UP VSE CONNECTOR SERVER
        DATE 04/25/2008, CLOCK 10/22/07
R1 0045 IESC1001I BEGINNING STARTUP OF VSE CONNECTOR SERVER
R1 0045 IESC1011I USING CONFIG FILE:          DD:PRD2.CONFIG(IESVCSRV.Z)
R1 0045 IESC1012I USING LIBRARIAN CONFIG FILE: DD:PRD2.CONFIG(IESLIBDF.Z)
R1 0045 IESC1013I USING USERS CONFIG FILE:    DD:PRD2.CONFIG(IESUSERS.Z)
R1 0045 IESC1014I USING PLUGIN CONFIG FILE:   DD:PRD2.CONFIG(IESPLGIN.Z)
R1 0045 IESC1060I USING SSL CONFIG FILE:      DD:PRD2.CONFIG(IESSSLCF.Z)
R1 0045 T045: SSL100I IPCRYPTO 01.05 E 10/10/06 11.26 006DC000 806E0000 00620E
R1 0045 IESC1018I LOADING PLUGIN:  IESSAPLG
R1 0045 IESC1018I LOADING PLUGIN:  IESHTOHP
R1 0045 IESC1018I LOADING PLUGIN:  IESCOMPH
R1 0045 IESC1018I LOADING PLUGIN:  IESVSAPL
R1 0045 IESC1018I LOADING PLUGIN:  IESDLIPL
R1 0045 BSD100I IPNRBSDC 01.05 E 10/10/06 07.12 0071D800 05843C00 0583CB80
R1 0045 IESC1002I FINISHED STARTUP OF VSE CONNECTOR SERVER
R1 0045 IESC1003I WAITING FOR CONNECTIONS OF CLIENTS...

```

The SSL100I message tells us that the server is now in SSL mode. To verify this, you can also use the server's STATUS command:

```

msg r1,data=status
AR 0015 1I40I  READY
R1 0045 IESC1029I STATUS COMMAND
R1 0045  SERVER CONFIG FILE      = DD:PRD2.CONFIG(IESVCSRV.Z)
R1 0045  CONFIGURATION INFORMATION:
R1 0045  MAX NUM. OF CLIENTS     = 256
R1 0045  TCP/IP SERVER PORT      = 2893
R1 0045  SSL ENABLED             = YES
R1 0045  SECURITY                = FULL
...

```

The server side is now ready. The following steps are to configure SSL at the client side.

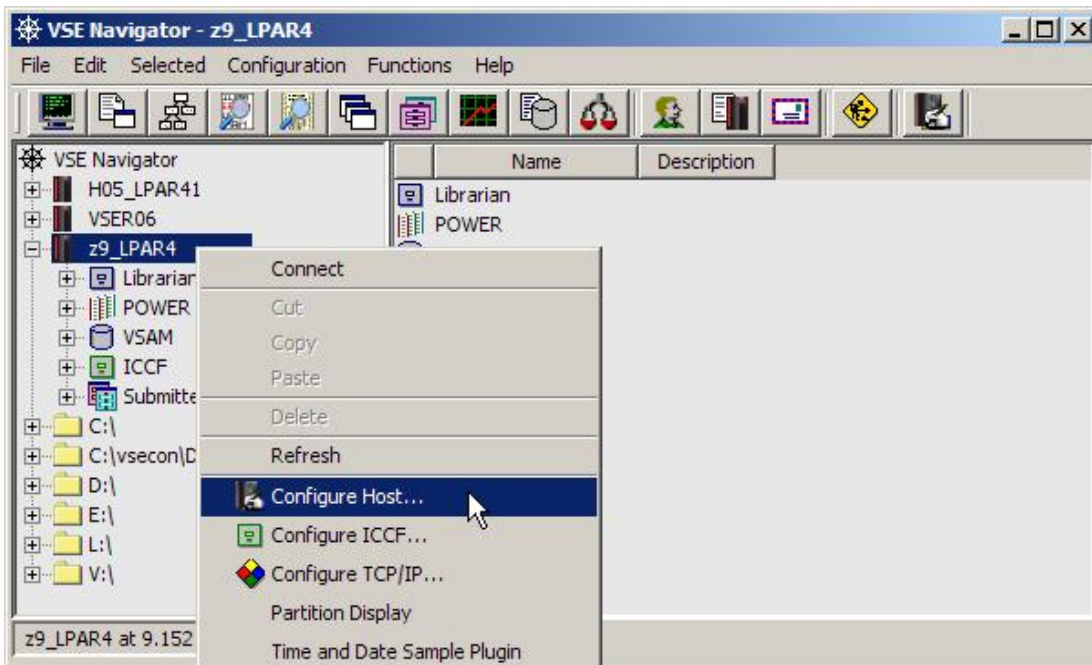
2.3 Setting up VSE Navigator for SSL

In our example we use the VSE Navigator as the SSL client.

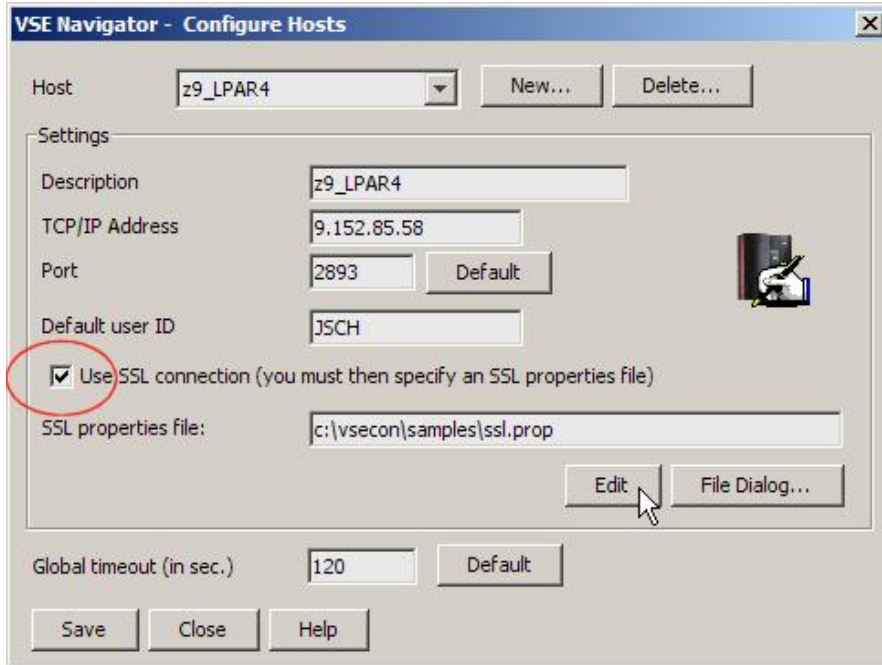
When running the VSE Navigator as SSL client, it has these prerequisites:

- Java 1.4 or higher on the workstation side
- TCP/IP for VSE/ESA 1.5E on the VSE side
- VSE Connector Client installed on the client side
- VSE Connector Server up and running in SSL mode on the VSE side

In the VSE Navigator main window, right-click on your VSE host icon and select **Configure Host**.



On the Configure Hosts dialog box select checkbox **Use SSL connection** and setup your SSL properties file.



Then click on the **Edit** button and verify the settings in your SSL properties file. The SSL properties file is a plain text file, which contains various SSL settings, for example the name and location of your local keyring file, its password, the level of SSL protocol, whether you want to use server or client authentication and so on.

Tip:

- You might want to have a different SSL properties file for each of your VSE systems. Like for naming PFX files, use the following naming convention for SSL properties files:

machine_lpar_keyringname.prop

Example: z9_LPAR4_SSL01.prop

VSE Navigator displays the SSL properties file using the defined editor program.

```
#-----
# SSL properties file for VSE Connector Client
# Keyring file:
#-----
KEYRINGFILE=c:\\vsecon\\samples\\z9_LPAR4_SSL01.pfx

#-----
# Keyring password:
#-----
KEYRINGPWD=ssltest

#-----
# SSL Version:      SSL (SSL 2.0/3.0) or
#                   TLS (TLS 1.0)
#-----
SSLVERSION=SSL
```

```

#-----
# Available cipher suites with VSE:
#-----
CIPHERSUITES=TLS_RSA_WITH_AES_128_CBC_SHA,SSL_RSA_WITH_3DES_EDE_CBC_SHA,SSL_RSA
_WITH_DES_CBC_SHA,SSL_RSA_WITH_NULL_SHA,SSL_RSA_WITH_NULL_MD5,SSL_RSA_EXPORT_WI
TH_DES40_CBC_SHA

#-----
# Logon with the client's certificate: YES
#                                     NO
#-----
LOGONWITHCERT=NO

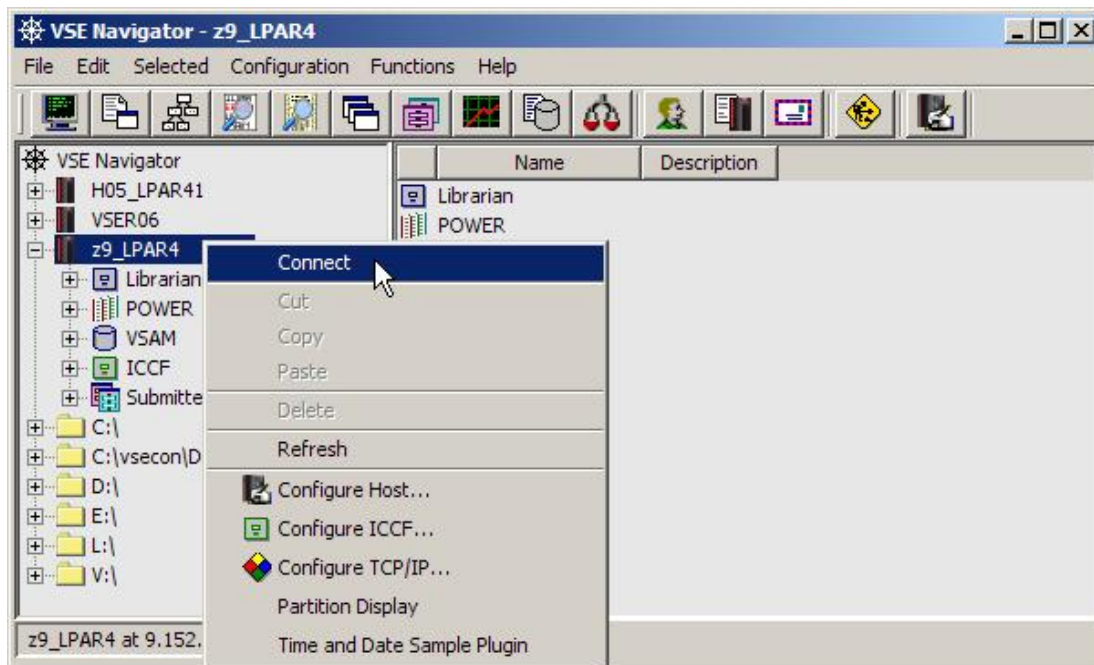
```

In this setup we include all available SSL cipher suites. Note that they have to be written in one single line in the file. No line breaks. As we first do the setup for SSL server authentication, we leave parameter LOGONWITHCERT =NO.

If you omit parameters in the SSL properties file, you will get prompted for any missing parameter when connecting to the VSE Connector Server. This, for example, allows for not specifying the keyring password here.

2.4 Connecting to VSE

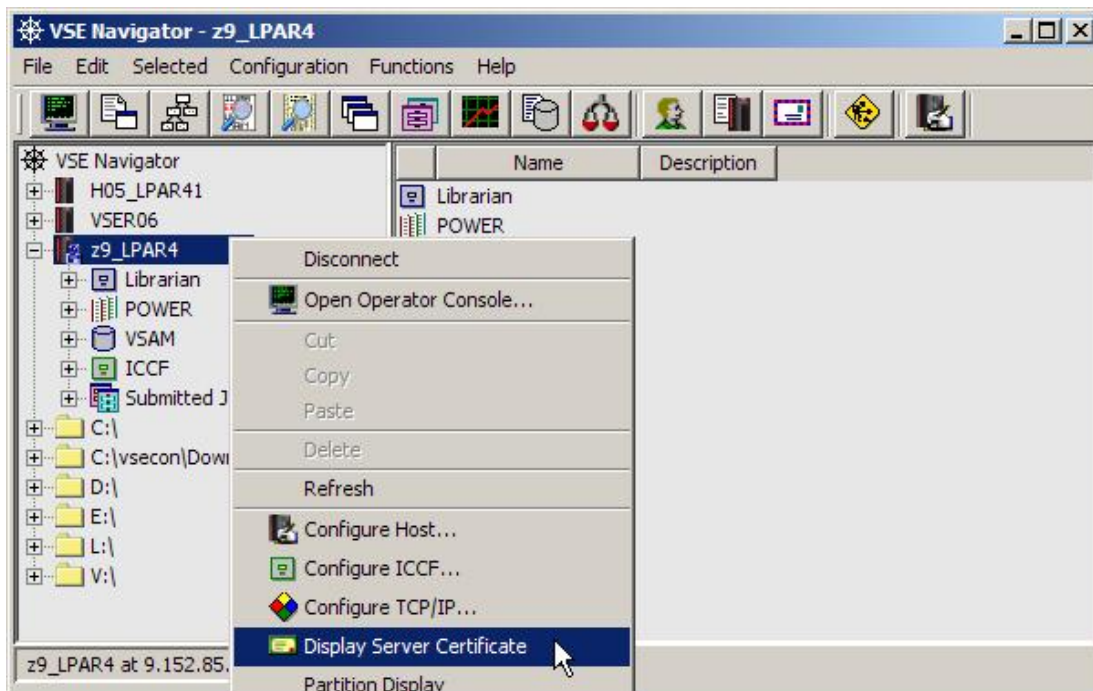
We are now ready for connecting to VSE.



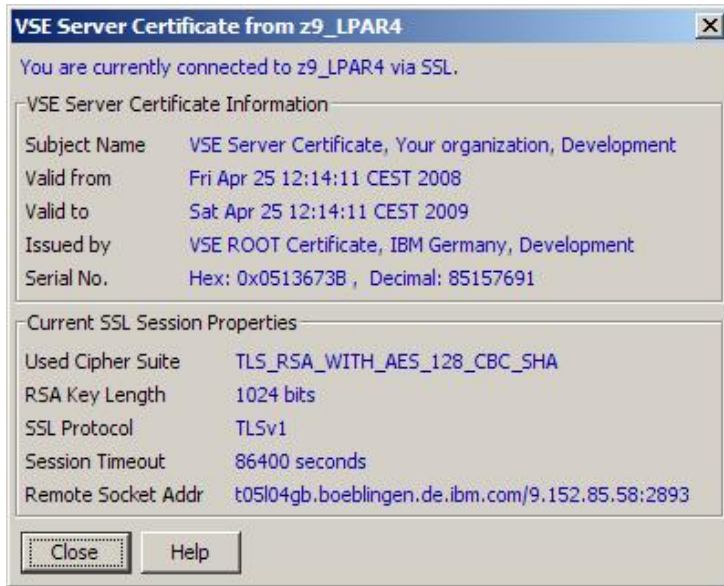
On the logon dialog box enter your VSE user ID and password.



When being connected, you can view some details about the current SSL session via menu choice **Display Server Certificate**.



The **VSE Server Certificate** box shows that we are connected using a 1024-bit RSA key. The used cipher suite indicates that we are using AES-128 for encrypting data that is sent over the line.



Here is the complete list of supported cipher suites and their meaning.

Hex Code	Cipher Suite	Handshaking (*)	Encryption	Min. TCP/IP
01	SSL_RSA_WITH_NULL_MD5	512	None	1.5D
02	SSL_RSA_WITH_NULL_SHA	512	None	1.5D
08	SSL_RSA_EXPORT_WITH_DES40_CBC_SHA	512	40 bits	1.5D
09	SSL_RSA_WITH_DES_CBC_SHA	1024	56 bits	1.5D
0A	SSL_RSA_WITH_3DES_EDE_CBC_SHA	1024	168 bits	1.5D
2F	TLS_RSA_WITH_AES_128_CBC_SHA	1024 / 2048	128 bits	1.5E
35	TLS_RSA_WITH_AES_256_CBC_SHA	1024 / 2048	256 bits	1.5E
62	RSA1024_EXPORT_DES_CBC_SHA	1024	56 bits	1.5D

Table 1: available cipher suites on VSE

Notes:

- When using 2048-bit keys you need a Crypto Express2 or PCI-X Cryptographic Coprocessor card.
- AES support was introduced with TCP/IP fix ZP15E214.
- AES-128 is available as hardware function on IBM System z9 processors and is much faster than the software implementations provided by TCP/IP. It is used transparently by TCP/IP when available.
- (*) TCP/IP fix ZP15E250 removes the restriction of allowing some cipher suites only with a specific RSA key length. If you look at the RFC2240 for TLS you will notice that it does not have a RSA key size associated with the specific cipher suites. Any cipher suite can now be used with any of the RSA key sizes.

2.5 Restrictions with client authentication

Due to restrictions imposed by Java runtime environments, client authentication is very limited with the Java-based connector when using PFX files on the PC side.

In fact, when we initially shipped the VSE Connectors on the basis of Java 1.3 everything worked fine, because we used a separate Java class library from IBM providing SSL functionality. With Java 1.4, SSL

was included into Java, but since then has a problem handling PFX files. Only with a Java 1.4.2 from IBM it was possible in our test environment to get client authentication to work with a local PFX file. The solution for the problem is using Java Key Stores (JKS) instead of PFX files.

While JKS has been invented by Sun Microsystems and is therefore fully supported by Java, PFX is usually the format that can be imported in Web browsers. So in fact we need both, depending on our SSL client.

Since July 2009 Keyman/VSE is enhanced to support JKS keystores. The related fixes in the VSE Connector Client are provided with xxxxxx.

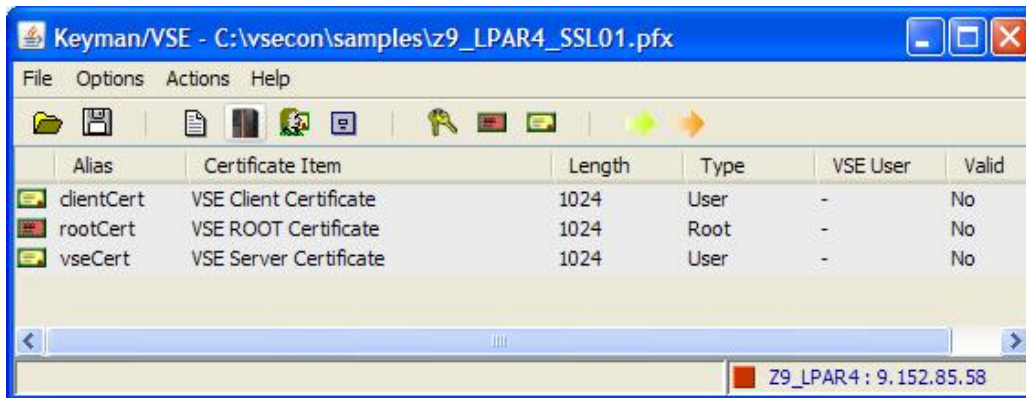
2.6 Setting up for SSL client authentication

You may now change our current setup for SSL client authentication. On the PC side we have to use a Java keystore (JKS) instead of the previously used PFX file.

The steps are:

- Open your previously created keyring file with Keyman/VSE (build July 2009)
- Save its contents as a JKS file
- Change the SKVCSSSL member on the VSE side and restart the connector server

Open your previously created keyring file.

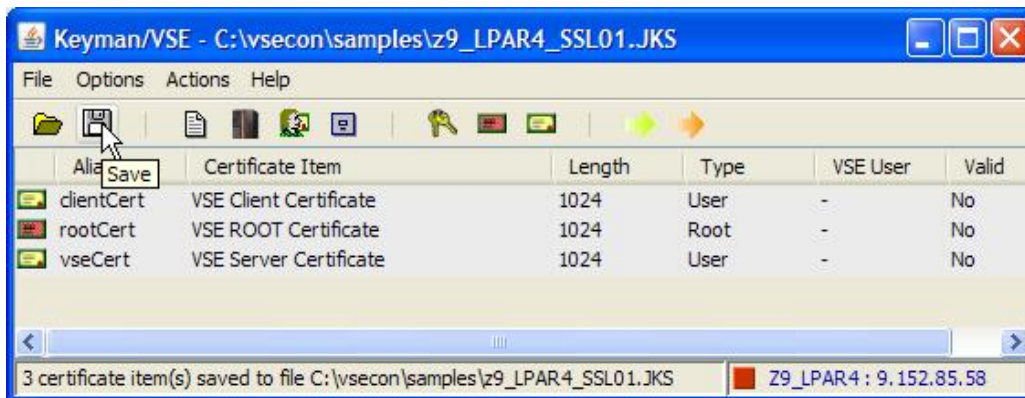


Open the “Local file properties” box and change the file type to JKS. The file extension in the “File Name” text field is automatically changed to JKS.



Press **OK**.

Now save the file contents in the new format.



Now change the SKVCSSSL member on the VSE side.

```

SSLVERSION      = SSL30
KEYRING         = CRYPTO.KEYRING
CERTNAME        = SSL01
SESSIONTIMEOUT  = 86440
AUTHENTICATION   = CLIENT

```

Change parameter **AUTHENTICATION** to **CLIENT**. Then catalog the changed configuration member (see section Setting up SKVCSSSL for server authentication on page 12).

There are no changes necessary in the SSL properties file for the VSE Navigator.

You can now connect to VSE using SSL client authentication.

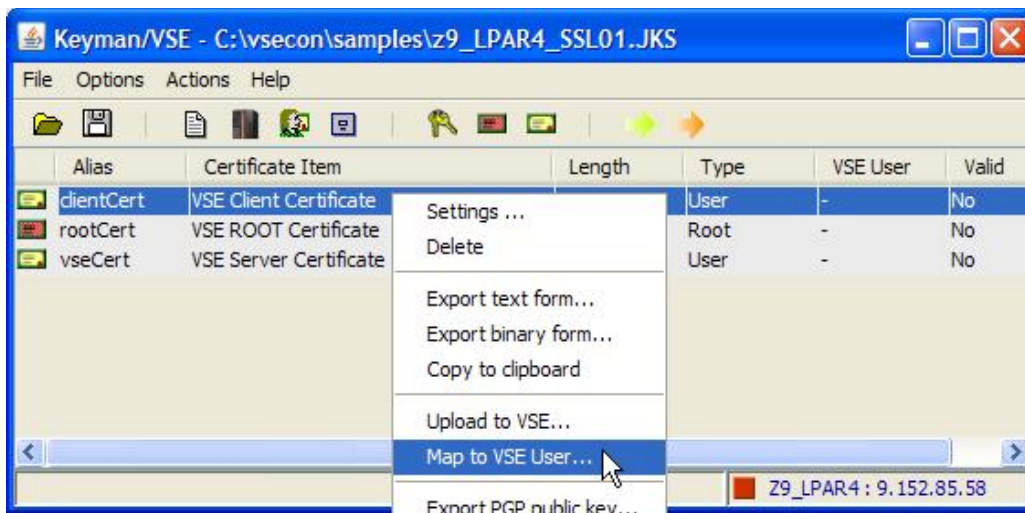
2.7 SSL client authentication with implicit logon

SSL client authentication with implicit logon means that the SSL client certificate is mapped to a VSE user ID. When receiving the client certificate during the SSL handshake, an implicit logon to VSE is performed with this user ID without prompting for the password.

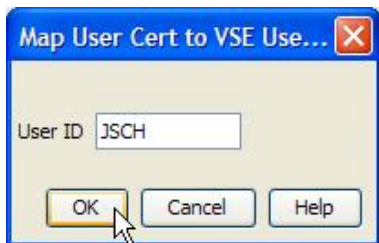
The steps are:

- Map the SSL client certificate to a VSE user ID in Keyman/VSE
- Upload the client certificate to VSE and activate the certificate mapping
- Change the SKVCSSSL member for implicit logon
- Change the SSL properties file for VSE Navigator for implicit logon

First, map the SSL client certificate to a VSE user ID in Keyman/VSE.



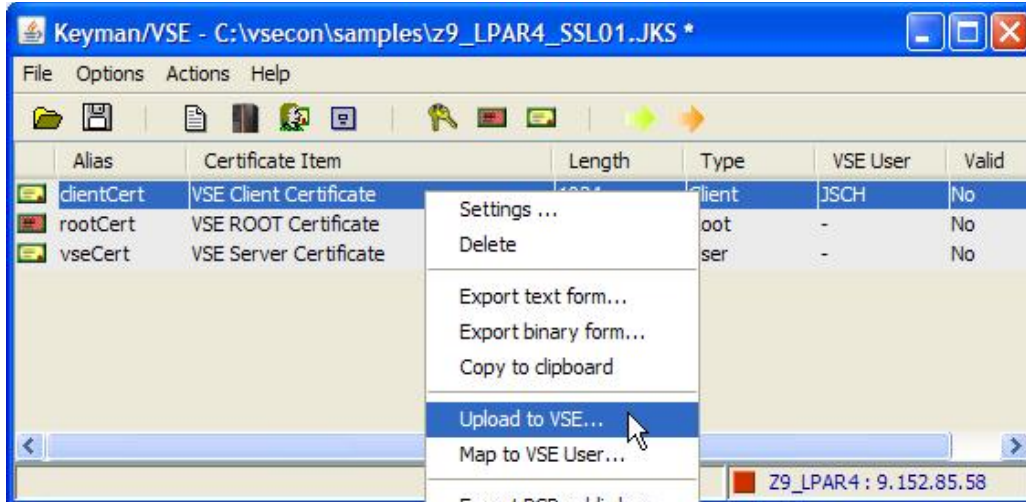
Then right-click the client certificate and select “Map to VSE User”.



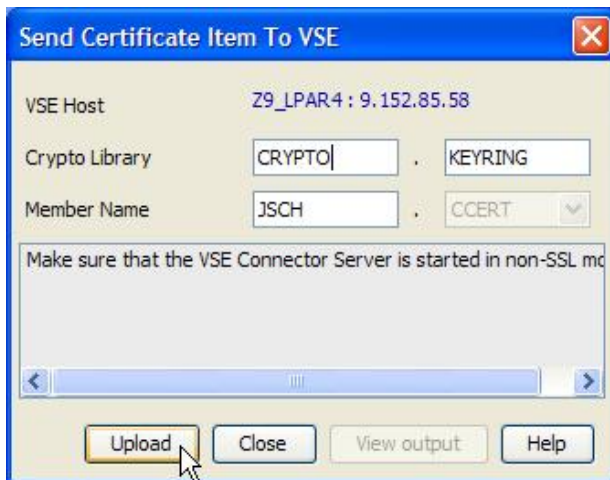
Enter the VSE user ID and press **OK**.

The client certificate now shows the mapped user ID in the Keyman/VSE main window.

Now upload the certificate to VSE.



Right-click the client certificate and select “Upload to VSE”.



Keyman/VSE submits two jobs, one for uploading the client certificate and a second to do the user ID mapping. An output similar to the following should be produced.

```

BG 0001 1Q47I  BG BSSDCERT 11908 FROM (JSCH) , TIME=15:35:40
BG 0000 // JOB BSSDCERT
          DATE 12/02/2009, CLOCK 15/35/40
BG 0000 EOJ BSSDCERT  MAX.RETURN CODE=0000
          DATE 12/02/2009, CLOCK 15/35/41, DURATION  00/00/00
BG 0001 1Q34I  BG WAITING FOR WORK
BG 0001 1Q47I  BG BSSDCERT 11913 FROM (JSCH) , TIME=15:35:42
BG 0000 // JOB BSSDCERT
          DATE 12/02/2009, CLOCK 15/35/42
BG 0000 BSSD01I TABLE OF DIGITAL CERTIFICATES ACTIVATED
BG 0000 EOJ BSSDCERT  MAX.RETURN CODE=0000
          DATE 12/02/2009, CLOCK 15/35/42, DURATION  00/00/00

```

You can view a list of mapped certificates using IUI dialog 2.8.4:

```

TAS$CERS          CLIENT CERTIFICATES - USER IDS

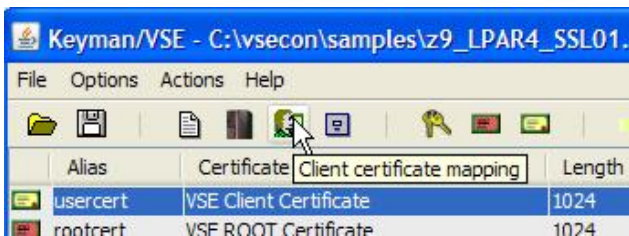
Enter the required data and press ENTER.

OPTIONS: 1 = ADD          2 = CHANGE      5 = DELETE

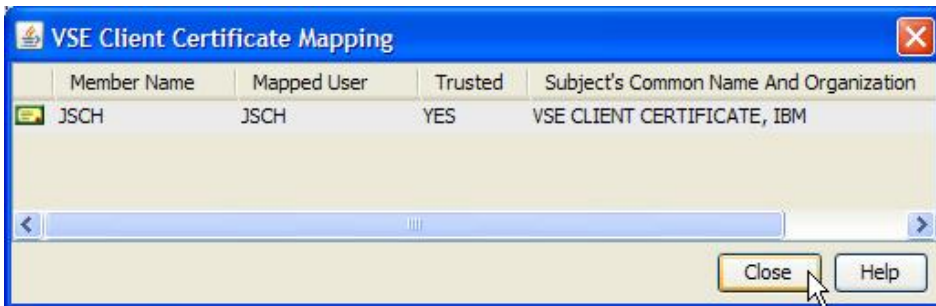
OPT              CERTIFICATE                      USERID  TRUSTED
COMMON NAME      MEMBER NAME
-               VSE Client Certificate, IBM        JSCH     JSCH     X
-
-
-
-
-
-
-
-
-
-
-
-

LOCATE MEMBER NAME == > _____
PF1=HELP          2=REDISPLAY  3=END           5=PROCESS      6=ACTIVATE
    
```

Keyman/VSE can display the same information.



Press toolbar button “Client certificate mapping” to get a list of mapped certificates.



Now change the SKVCSSSL member for implicit logon.

```

SSLVERSION       = SSL30
KEYRING          = CRYPTO.KEYRING
CERTNAME         = SSL01
SESSIONTIMEOUT  = 86440
AUTHENTICATION   = LOGON
    
```

Change parameter AUTHENTICATION to LOGON.

Finally change the VSE Navigator's SSL properties file. Change parameter LOGONWITHCERT to YES.

```
LOGONWITHCERT=YES
```

When now connecting to VSE with VSE Navigator, the logon dialog box is no more shown.

2.8 Using encryption with AES-256

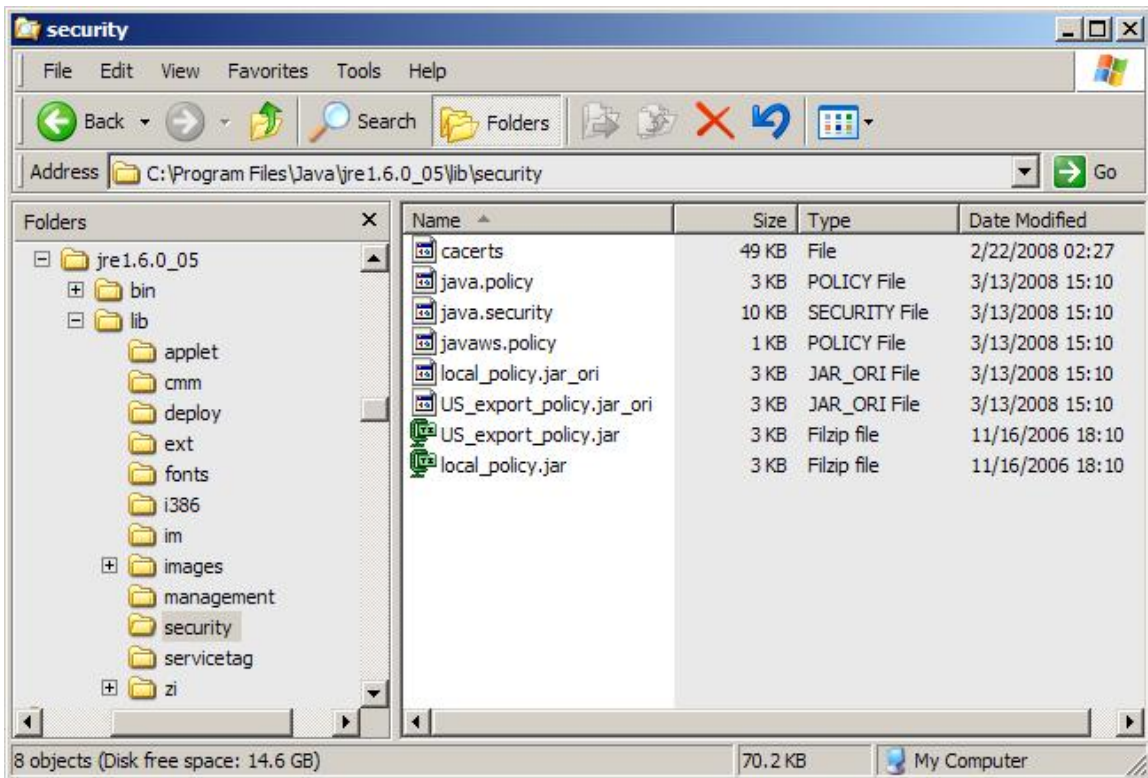
By default, your Java installation does not support AES with key sizes that are greater than 128 bits. However, to use AES-256 you need unlimited strength cryptography.

Due to import-control restrictions imposed by some countries, the jurisdiction policy files shipped with Java only permit strong cryptography to be used. An unlimited strength version of these files (that is, with no restrictions on cryptographic strength) is available for download on this Web page:

<http://java.sun.com/products/jce/javase.html>

To activate unlimited strength cryptography in Java:

- Replace the files *local_policy.jar* and *US_export_policy.jar* in the directory ...*\lib\security* of your Java installation.
- Restart your Java application.



The same files can also be used to activate unlimited strength cryptography for an IBM Java.

When using the VSE Navigator, specify following cipher suite in the SSL properties file:


```
CIPHERSUITES=TLS_RSA_WITH_AES_256_CBC_SHA
```

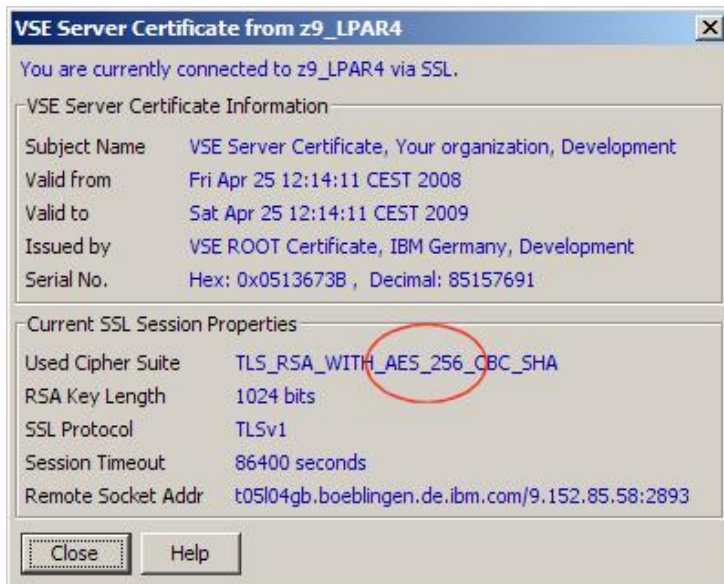
On the host side make sure this cipher suite is contained in the list as specified in the SKVCSSSL member:

```
; *****
; CIPHERSUITES SPECIFIES A LIST OF CIPHER SUITES THAT ARE ALLOWED
; *****
CIPHERSUITES = 35 ; TLS_RSA_WITH_AES_256_CBC_SHA
```

Notes:

- AES-256 requires TCP/IP for VSE/ESA 1.5E with fix number ZP15E214.
- When running on a z9 or below, AES-256 is performed in software
- When running on a z10, AES-256 is provided via the CPU Assist feature (CPACF), which is exploited by TCP/IP transparently.

Now restart the VSE Connector Server.



The connection is now encrypted using AES-256.

The following chapter describes the SSL setup for a Web server environment using the HTTPD and TLSDD daemons.

3 SSL setup for TLSD

Setting up SSL for the HTTP protocol is based on the SSL daemon provided by TCP/IP for VSE/ESA. There are two modes available:

- SSL in *native* mode. Hereby SSL traffic goes directly to the SSL enabled daemon on VSE. Native mode is supported by the HTTPD daemon only. In addition to the HTTPD daemon you have to define a TLSD daemon on the same port. The HTTP daemon will then get all SSL related settings from the related TLSD daemon.
- SSL in *pass through* mode, which for example has to be used for secure Telnet, but may also be used for HTTP. Also here we have to define a TLSD daemon. The difference to native mode is that we use the PASSPORT parameter to route SSL traffic from an unsecured daemon to the SSL daemon.

As the preferred way is native mode, the following section describes the setup for native mode.

3.1 Setup SSL native mode with HTTPD

Here is an example for setting up native mode SSL with a HTTP daemon. The setup uses the VSE keyring members that we created in section Create the VSE key and certificates on page 7.

Add the following definitions to your TCP/IP IPINIT member.

DEFINE TLSD, ID=TLSD1,	ID of this SSL/TLS daemon
PORT=443,	Default HTTPS port
PASSPORT=443,	Native support
CIPHER=2F350A096208,	Allowed cipher suites
CERTLIB=CRYPTO,	Library name
CERTSUB=KEYRING,	Sublibrary name
CERTMEM=SSL01,	Member name
MINVERS=0300,	Minimum version required
TYPE=1,	SSL server authentication
DRIVER=SSLD	Driver phase name

In the above definition, the PORT and the PASSPORT parameter are identical, which indicates native SSL mode. Also the HTTPD daemon below must specify the same port number.

DEFINE HTTPD, ID=HTTPS,	ID of this HTTP daemon
PORT=443,	Default HTTPS port
COUNT=3,	Start 3 sessions
ROOT=PRIMARY.HTTPS	Location of index.html
CONFINE=YES,	Confine to a specific lib
DRIVER=HTTPD	Driver phase name

Note: it is useful to define multiple HTTPD daemons to have multiple parallel sessions available. In the above example we set the COUNT parameter to 3, which starts 3 internal HTTP daemons.

In order to display a simple welcome page, we put an *index.html* file into VSE library PRIMARY.HTTPS, which is the document root of our HTTP server.

```

<HTML>
<head>
<TITLE>Greetings from VSE</TITLE>
</head>
<body>

<h2>Hello world</h2>

<p>
</body>
</html>

```

This HTML file is stored as *index.html* in PRIMARY.HTTPS.

```

LD *.*

DIRECTORY DISPLAY      SUBLIBRARY=PRIMARY.HTTPS      DATE: 2008-04-25
                                                                TIME: 11:24
-----
 M E M B E R      CREATION   LAST      BYTES      LIBR CONT SVA  A- R-
NAME      TYPE      DATE      UPDATE     RECORDS   BLKS STOR ELIG MODE
-----
INDEX     HTML      07-08-17  - -        4 R       1 YES  - - -
L113I RETURN CODE OF LISTDIR IS 0
L001A ENTER COMMAND OR END

```

Before trying to connect, read the next section.

3.2 Considerations on \$WEB user

TCP/IP 1.5E needs a special user ID \$WEB in order to let a Web Browser connect to the HTTPD daemon. This user must either be defined in the IPINIT member, like

```
DEFINE USER, ID=$WEB, PASSWORD=$WEB, WEB=YES
```

or it must be defined as a VSE user ID so that the \$WEB user is known to the Basic Security Manager (BSM). This is described in APAR PQ87041, which is the IBM APAR for TCP/IP service pack 1.5E. If the \$WEB user is not known you will get following error from the Basic Security Manager (BSM):

```
F7 0100 BSST20I INVALID USER ID $WEB      IP ADDRESS = 9.152.216.58
F7 0098 IPN755I Fail File OpenRead $WEB 9.152.216.58 1288 PRIMARY
```

The APAR problem summary is available online at

<http://www.ibm.com/servers/eserver/zseries/zvse/support/>

It states that these special users must be defined in TCP/IP for the different IP protocols.

```

DEFINE USER, ID=$WEB, PASSWORD=$WEB, WEB=YES
DEFINE USER, ID=$LPR, PASSWORD=$LPR, LPR=YES
DEFINE USER, ID=$EVENT, PASSWORD=$EVENT, LPR=YES
DEFINE USER, ID=$LPD, PASSWORD=$LPD, LPD=YES

```

Important: there is a possible security leak.

- When for example \$WEB is defined as VSE type1 user everything works, but the system is now open for anyone who knows about the \$WEB user and its publicly known password. So you have to restrict the access rights of \$WEB to a minimum via the BSM batch security. This requires to IPL your system with SYS SEC=YES. This is my currently recommended option.
- When \$WEB is only defined in the IPINIT member and not known to the BSM, you would have to turn IP security off to get it to work. Otherwise, when being contacted by the HTTPD daemon in order to access the index.html file, BSM would reject the file access, because the user is not known. But turning off IP security just because you want to run a HTTPD is probably not what you want.

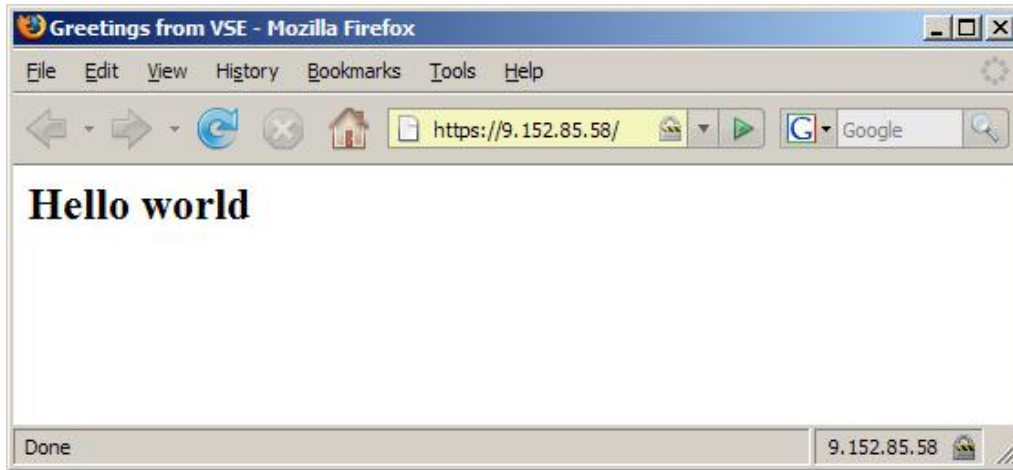
3.3 Connect to the HTTPD using a Web Browser

As we are using the default port 443 for secure HTTPS connections, we can omit the port number in the URL.

When defining our VSE server certificate in section Create the VSE key and certificates on page 7, we specified "VSE Server Certificate" as the Common Name. Web Browsers usually expect the common name in the received server certificate to be the same as the IP address or symbolic name of this server. The following message box appears, because there is a mismatch in our case.



Press **OK** to continue.



The following section shows how to check the available SSL cipher suites in Microsoft Internet Explorer. If you don't get any errors, or you don't want to force any specific cipher suite for your SSL connection, you may skip this section.

3.4 Configure Ciphers in MS Internet Explorer

SSL cipher suites are not visible through any Internet Explorer dialogs. Instead, they are defined in the Windows registry and therefore affect your entire computer. This is described on

<http://support.microsoft.com/>

Search for "SSL cipher suites" and you are directed to the most current knowledge base entry.

3.5 Configure Ciphers in Mozilla Firefox

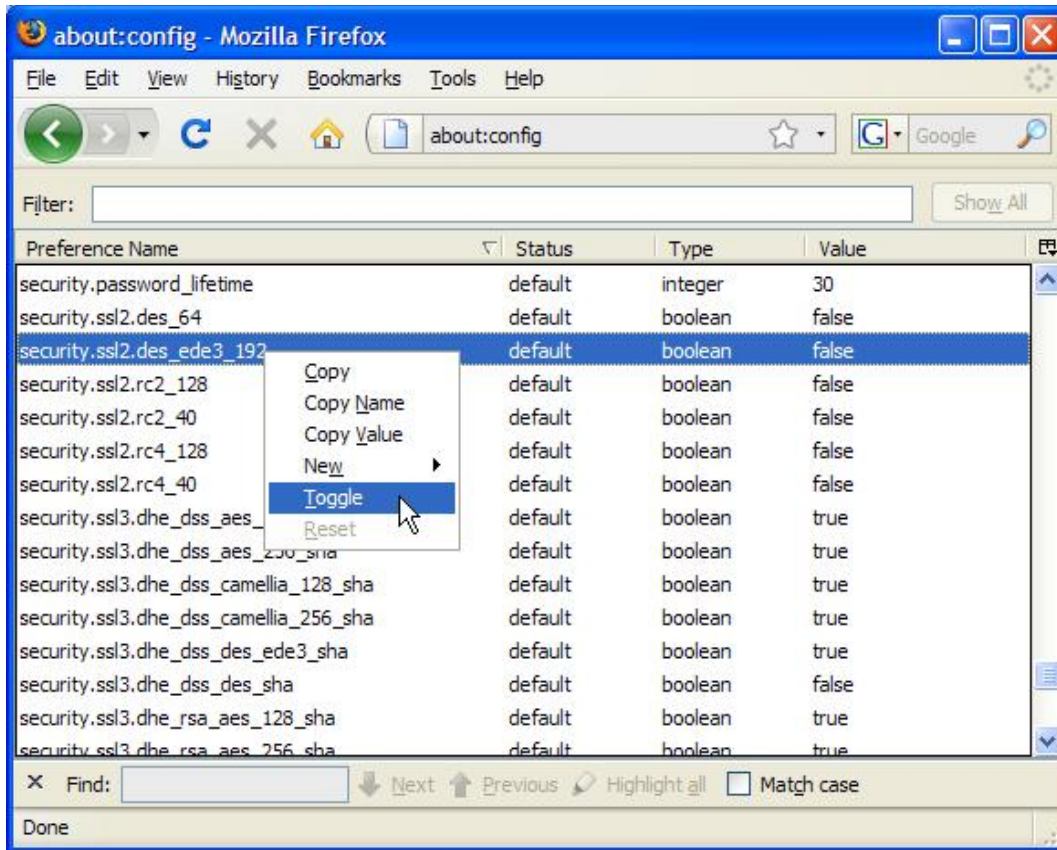
You can find some information of configuring SSL cipher suites in Mozilla Firefox on

https://developer.mozilla.org/en/Security_in_Firefox_2

Basically, you just enter

<about:config>

in the browser's address field. You are then prompted with some warning message. After proceeding with the dialog, you can view and change the entire browser configuration.



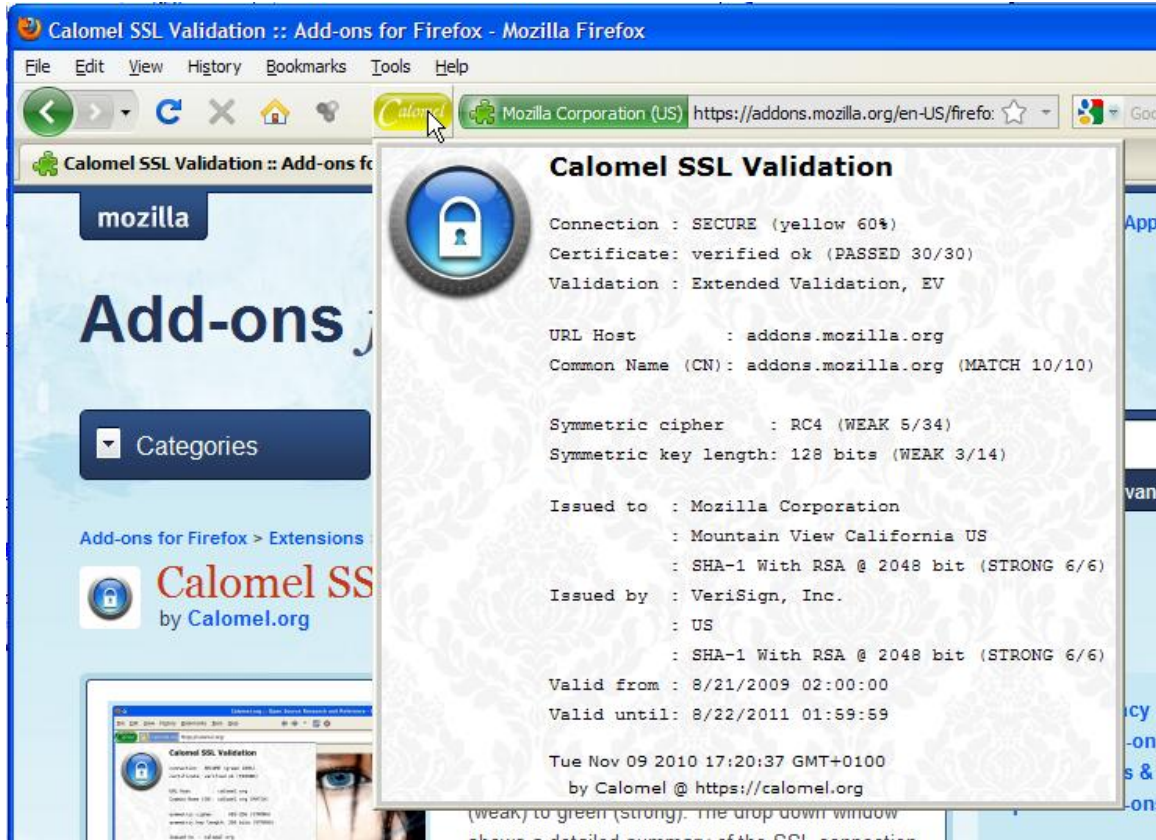
You can change displayed values by right-clicking an entry.

3.6 Displaying SSL properties in Mozilla Firefox

There is a variety of add-ons available for the Mozilla Firefox browser. One of them, called “Calomel”, can display a detailed summary of the SSL connection. You can download the add-on from

<https://addons.mozilla.org/de/firefox/addon/207653/>

When being connected via SSL, the Calomel toolbar button will change its color depending on the strength of encryption from red (weak) to green (strong). The drop down window shows a detailed summary of the SSL connection.



4 Hardware- versus software-based encryption

As described in previous sections, TCP/IP for VSE/ESA uses hardware-based encryption when available. With a \$SOCKOPT phase you can explicitly control whether encryption is performed in hardware or software.

The following jobs can be used to either enforce or disable hardware-based encryption.

4.1 *Suppressing the CPU Assist feature*

This job suppresses the use of the CPU Assist feature.

```

* $$ JOB JNM=SOCKOPT1,CLASS=0,DISP=D
// JOB SOCKOPT
*
* * CREATE A $SOCKOPT.PHASE THAT SUPPRESSES THE
* * USE OF THE KMC INSTRUCTION
*
// OPTION CATAL
// LIBDEF *,SEARCH=PRD1.BASE
// LIBDEF *,CATALOG=PRD1.BASE
// EXEC ASMA90,SIZE=ASMA90,PARM='SZ(MAX-200K,ABOVE) '
      PUNCH      ' PHASE $SOCKOPT,* '
$SOCKOPT CSECT
      SOCKOPT CSECT,SSLFLG2=$OPTSNZA
      END      $SOCKOPT
/*
// EXEC LNKEDT,SIZE=512K
/*
/&
* $$ EOJ

```

4.2 Forcing the CPU assist feature

This job forces the use of the CPU Assist feature.

```

* $$ JOB JNM=SOCKOPT2,CLASS=0,DISP=D
// JOB SOCKOPT2
*
* * THIS JOB WILL CREATE A $SOCKOPT.PHASE THAT FORCES THE
* * USE OF THE KMC INSTRUCTION
*
// OPTION CATAL
// LIBDEF *,SEARCH=PRD1.BASE
// LIBDEF *,CATALOG=PRD1.BASE
// EXEC ASMA90,SIZE=ASMA90,PARM='SZ(MAX-200K,ABOVE) '
      PUNCH      ' PHASE $SOCKOPT,* '
$SOCKOPT CSECT
      SOCKOPT CSECT,SSLFLG2=$OPTSFZA
      END      $SOCKOPT
/*
// EXEC LNKEDT,SIZE=512K
/*
/&
* $$ EOJ

```

4.3 Suppressing the use of crypto cards

This job suppresses the use of crypto cards, like PCICA, PCIXCC, or Crypto Express2. Note that RSA keys with 2048 bits require a PCIXCC or Crypto Express2.


```

* $$ JOB JNM=$$SOCKOPT,CLASS=0,DISP=D
* $$ LST CLASS=A
// JOB $SOCKOPT
// OPTION CATAL
// LIBDEF *,SEARCH=PRD1.BASE
// LIBDEF *,CATALOG=PRD1.BASE
// EXEC ASMA90,SIZE=ASMA90,PARM='SZ(MAX-200K,ABOVE)'
      PUNCH      ' PHASE $SOCKOPT,* '
$SOCKOPT CSECT
      SOCKOPT  CSECT,                Generate a csect          X
              SSLFLG2=$OPTSNHC,      SSL do not use hw-crypto  X
      END      $SOCKOPT
/*
// EXEC LNKEDT,SIZE=512K
/*
/&
* $$ EOJ

```

The default is using crypto cards when available.

4.4 Displaying available crypto hardware

You can display available cryptographic hardware on your VSE system with the following operator console command:

```

msg fb,data=status=cr
AR 0015 1140I  READY
FB 0011 BST223I  CURRENT STATUS OF THE SECURITY TRANSACTION SERVER:
FB 0011 ADJUNCT PROCESSOR CRYPTO SUBTASK STATUS:
FB 0011  AP CRYPTO SUBTASK STARTED ..... : YES
FB 0011  MAX REQUEST QUEUE SIZE ..... : 1
FB 0011  MAX PENDING QUEUE SIZE ..... : 1
FB 0011  TOTAL NO. OF AP REQUESTS ..... : 43
FB 0011  NO. OF POSTED CALLERS ..... : 43
FB 0011  AP CRYPTO POLLING TIME (1/300 SEC).. : 1
FB 0011  AP CRYPTO WAIT ON BUSY (1/300 SEC).. : 75
FB 0011  AP CRYPTO RETRY COUNT ..... : 5
FB 0011  AP CRYPTO TRACE LEVEL ..... : 3
FB 0011  TOTAL NO. OF WAITS ON BUSY ..... : 0
FB 0011  CURRENT REQUEST QUEUE SIZE ..... : 0
FB 0011  CURRENT PENDING QUEUE SIZE ..... : 0
FB 0011  ASSIGNED APS : PCICC / PCICA ..... : 0 / 0
FB 0011                   CEX2C / CEX2A ..... : 1 / 1
FB 0011                   PCIXCC ..... : 0
FB 0011  AP 0 : CEX2A  - ONLINE
FB 0011  AP 1 : CEX2C  - ONLINE
FB 0011  ASSIGNED AP QUEUE (CRYPTO DOMAIN)... : 4
FB 0011 CPU CRYPTOGRAPHIC ASSIST FEATURE:
FB 0011  CPACF AVAILABLE ..... : YES
FB 0011  INSTALLED CPACF FUNCTIONS:
FB 0011  DES, TDES-128, TDES-192
FB 0011  AES-128
FB 0011  SHA-1, SHA-256
FB 0011  END OF CPACF STATUS

```

5 Debugging SSL/TLS connections

This chapter shows how to activate tracing on VSE and on the workstation.

5.1 Tracing on VSE

To activate tracing on VSE just catalog a \$SOCKDBG phase on VSE.

```
* $$ JOB JNM=SOCKDBG,CLASS=A,DISP=D
// JOB $SOCKDBG
// OPTION CATAL
// LIBDEF *,CATALOG=lib.sublib
// EXEC ASMA90,SIZE=ASMA90
      PUNCH      ' PHASE $SOCKDBG,* '
$SOCKDBG CSECT
      SOCKDBG  CSECT,          GENERATE A PHASE                X
              FL01=$DBGWLST,  +DBGWLOG, MESSAGES TO SYSLST AND SYSLOG X
              FL02=$DBGISON,  DEBUG IS ON                     X
              FL03=$DBGNONE,  NONE                             X
              MSGT=$DBGALL,    ISSUE ALL DIAGNOSTIC MESSAGES   X
              DUMP=$DBGNONE,  NO DIAGNOSTIC SDUMPS FOR IPNRBDC X
              SSLD=$DBGSDMP,  YES DIAGNOSTIC SDUMPS FOR IPCRYPTO X
              CIAL=$DBGSDMP,  YES DIAGNOSTIC SDUMPS FOR IPDSCIAL X
              CECZ=$DBGNONE   NO DIAGNOSTIC SDUMPS FOR CIALCECZ
      END      $SOCKDBG
/*
// EXEC LNKEDT,SIZE=512K
/*
/ &
* $$ EOJ
```

Make sure that the SSL server on VSE (e.g. VSE Connector Server) uses option NOSYSDMP so that the trace output, which consists of many small SDUMPs, is written to SYSLST. Also, you should use

```
// UPSI 1
```

5.2 Tracing in Java

To activate SSL/TLS tracing in a Java application, like VSE Navigator, use the Java DEBUG option.

```
java -Djavax.net.debug=all com.ibm.vse.navigator.VSENavigator %*
```

In this example you modify the run.bat file of the VSE Navigator installation.

6 More information

You can find more information in these books:

z/VSE Administration

<http://www.ibm.com/servers/eserver/zseries/zvse/documentation/#vse>

z/VSE e-business Connectors User's Guide

<http://www.ibm.com/servers/eserver/zseries/zvse/documentation/#conn>

Redbook: Security on IBM z/VSE, SG24-7691

<http://www.redbooks.ibm.com/abstracts/sg247691.html?Open>

Debugging SSL/TLS Connections

<http://download.oracle.com/javase/1.5.0/docs/guide/security/jsse/ReadDebug.html>